AuthentiSense: A Scalable Behavioral Biometrics Authentication Scheme using Few-Shot Learning for Mobile Platforms

Hossein Fereidooni*, Jan König†, Phillip Rieger*, Marco Chilese*, Bora Gökbakan‡
Moritz Finke†, Alexandra Dmitrienko†, and Ahmad-Reza Sadeghi*
*Technical University of Darmstadt, Germany
†University of Würzburg, Germany
‡KOBIL, Germany

***Abstract*—Mobile applications are widely used for online services sharing a large amount of personal data online. One-time authentication techniques such as passwords and physiological biometrics (e.g., fingerprint, face, and iris) have their own advantages but also disadvantages since they can be stolen or emulated, and do not prevent access to the underlying device, once it is unlocked. To address these challenges, complementary authentication systems based on behavioural biometrics have emerged. The goal is to continuously profile users based on their interaction with the mobile device. However, existing behavioural authentication schemes are not (i) user-agnostic meaning that they cannot dynamically handle changes in the user-base without model re-training, or (ii) do not scale well to authenticate millions of users.**

In this paper, we present AuthentiSense, a user-agnostic, scalable, and efficient behavioural biometrics authentication system that enables continuous authentication and utilizes only motion patterns (i.e., accelerometer, gyroscope and magnetometer data) while users interact with mobile apps. Our approach requires neither manually engineered features nor a significant amount of data for model training. We leverage a few-shot learning technique, called Siamese network, to authenticate users at a large scale. We perform a systematic measurement study and report the impact of the parameters such as interaction time needed for authentication and n-shot verification (comparison with enrollment samples) at the recognition stage. Remarkably, AuthentiSense achieves high accuracy of up to 97% in terms of F1-score even when evaluated in a few-shot fashion that requires only a few behaviour samples per user (3 shots). Our approach accurately authenticates users only after 1 second of user interaction. For AuthentiSense, we report a FAR and FRR of 0.023 and 0.057, respectively.

Keywords – Behavioural Biometrics, Authentication, Few-shot Learning, and Siamese Networks.

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.23194
www.ndss-symposium.org

I. Introduction

Today, traditional authentication methods such as multi-factor methods (based on SMS, or authenticator apps) are not sufficiently robust to prevent sophisticated attacks [12] leaving a gap for persistent, adaptive and user-friendly authentication schemes. One-time authentication methods, such as passwords or physiological biometrics on mobile platforms require users to explicitly interact with their devices, referred to as explicit authentication, to gain access to the device.

While physiological biometrics promise to create a safer and more convenient alternative to passwords, they are not infallible and suffer from inherent disadvantages related to the nature of physical characteristics. For instance, once the physical characteristics are exposed, they can be reused maliciously multiple times [27], [45].

On the other hand, behavioral biometrics authentication systems aim to address this challenge through an additional layer of security which frequently and unobtrusively monitors the user's interaction with the device [47]. The approach is to identify unique individual regularities in user's behavior and analyze several parameters such as touch gestures, navigation, and motion patterns, during user's online activities, to detect potential irregularities not related to the real user. Behavioral authentication systems promise to provide increased security due to the dynamic authentication and the resilience to circumvention their traits are hard to emulate or copy [60].

An advantage of behavioral biometrics is that they can be collected in non-obtrusively without disturbing the normal service utilization. Moreover, they enable constant user monitoring and ensure that only the authorized user can use the system, even after an initial identity check has been performed. This ensures a frictionless authentication process and prevents identity fraud, account takeover, and automated attacks such as recognizing non-human device activity (bots), Remote Access Trojans and emulators [1]. Behavioral biometrics authentication are rapidly emerging and being deployed by major enterprises, such as Mastercard and Deutsche Bank [5], [3].

Various behavioral authentication approaches for users of mobile devices have been proposed in the literature so far: they

collect user's unique individual features based on, e.g., motion sensors [20], [51], [38], [50], [22], [24], touch gestures [52], [67], [33], [68], or their combinations [29], [16], [21], [18], [70], and detect irregularities during an entire online session. In this paper we focus on behavioral authentication solutions which utilize motion sensor values.

Despite their added value, existing behavioural authentication solutions still face several challenges: i) they often require a large amount of training data to build an accurate model [53], ii) they are not scalable and only work for a limited number of users they were trained for (not user-agnostic) [61], iii) need a model per user to improve model performance which could result in a resource intensive implementation in deployment phase [24], and iv) often require a long interaction time to preciously learn users behaviour [62], [64].

Our Goal and Contributions. We present a framework for continuous user authentication that leverages behavioural biometrics and aims at tackling the aforementioned challenges. Our scheme is (i) efficient and does not require hand-crafted features for model training, (ii) scalable to authenticate millions of users, and (iii) user-agnostic, i.e., does not need to be re-trained when users are dynamically changing (i.e., joining or leaving the system).

For this, we utilize well-established few-shot learning technique [31], [32] in which the model can learn how to perform user authentication with a small amount of data (hence few-shot). More specifically, we use the Siamese neural network [17] which can also be used when even only one user behavioral sample is available (one-shot learning). As the name suggests, our model stems from 'Siamese twins' [55] where two networks share weights and biases with the intention to learn similarities as well as dissimilarities between input data (i.e., users behavior). The Siamese networks have been already utilized for behavioral user authentication [29], [24]. However, they do not solve the scalability problem (cf. Sect. VI).
Our contributions are summarized as follows:

- We present AuthentiSense, a user-agnostic behavioral authentication system that utilizes few-shot learning to authenticate users quickly at large scale without requiring a significant amount of data for model training.
- We develop an end-to-end neural network architecture that can dynamically handle user changes without requiring re-training and feature engineering of input data. It only utilizes motion patterns (i.e., accelerometer, gyroscope and magnetometer data). Our approach can achieve an accuracy of 97% in terms of F1-score for 3-shot verification, and can accurately authenticate users only after 1 second of user interaction.
- We conduct an extensive and systematic measurement study in which we investigate the impact of different parameters such as training strategy (pairwise or triplet), interaction time needed for authentication, and n-shot verification (comparison with enrollment samples) at recognition stage in our system.

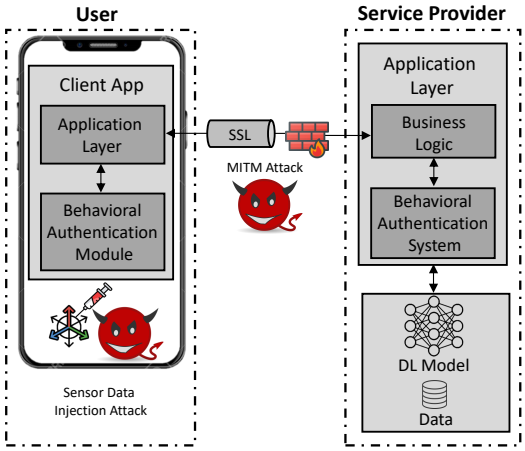


Fig. 1: Threat Model

II. Background

In this section, we provide the background information on Few-shot learning and Siamese neural networks. In App. B we provide further background information on neural networks.

A. Few-shot Learning

Few-shot learning is a machine learning method where models are trained with only a limited number of samples for each class. The common practice for machine learning applications is to feed as much data as the model can take. This is because feeding more data to most machine learning (especially deep learning) applications enables the model to generalize better. However, few-shot learning aims to build precise models with less training data. The fundamental idea is to learn new concepts and generalize tasks from only a few examples. One approach to few-shot learning includes Siamese networks which involve learning an embedding space to compare classes. Few-shot learning has gained traction in areas such as computer vision, natural language processing, audio processing, robotics and medical applications where a large number of classes exist and labeled data is scarce [66].

B. Siamese Networks

Siamese Networks have been utilized for few-shot classification or representation learning problems in which only a limited number of samples are available for each class. They can also be applied to classification problems that suffer from lack of enough data where only one sample per class is available (one-shot learning). A Siamese Network is a tandem of two identical of any given network architectures (i.e., MLP, RNN, or CNN) with the objective of finding the similarity or a relationship between two comparable observations. Both subnetworks share the same parameters (weights w and biases b). The objective of the training is to extract similar features (i.e., embedding vectors) if the two input samples belong to the same class, while extracting differing features if the two input data belong to the different classes where the similarity between vectors is typically quantified by a *p-norm* distance metric, with the Euclidean distance being the most commonly used one.

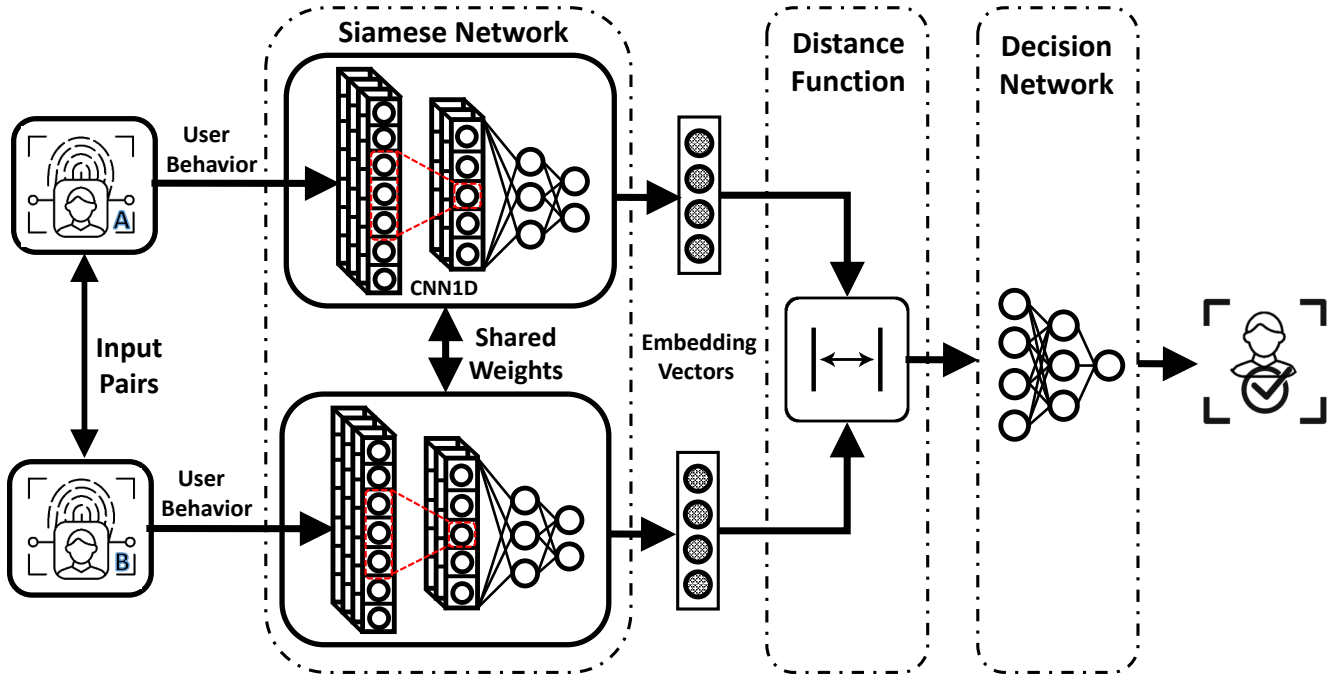


Fig. 2: An abstract view of AuthentiSense at inference time

III. System and Threat Model

Our system model, as shown in Figure 1, includes users who access online services through their mobile apps and a service provider after an authentication process. In this section, we describe the assumptions we make regarding the attacker capabilities and the threat model for our behavioral authentication system. The main objective of the adversary is to bypass the authentication system and impersonate the end-user while authenticating into a sensitive service (e.g., mobile banking). In particular, we make the following assumptions:

- The authentication model is trained and maintained in a data center owned by the service provider. In typical user authentication applications, to complete a successful and secure authentication process, the service provider is trusted. Since the authentication model is trained at the service provider, this will avoid adversarial machine learning such as model poisoning, or privacy attacks.
- The adversary can compromise the user's mobile device, e.g., by installing malware or exploiting vulnerabilities.
- Aligned with existing work [66], we assume that sensor data is trustworthy as its integrity can be protected by hardware security technologies such as ARM TrustZone [52]. The Behavioral Authentication Module (BAM) in the client application, Fig 1., can also run in the secure world of an ARM TrustZone. The motion sensors can be exclusively assigned to the secure world (and the BAM) to avoid any access to sensor readings from outside of the secure world.
  In case the OS is compromised, the adversary can still not access the sensor, since reading from the sensor is only possible from the secure world.
- Side-channel attacks to extract user behavioural data from device sensors are orthogonal to this work and can be mitigated with sensor data obfuscation techniques [28], [54].
- The communication channels are secured through the use of standard secure communication protocols such as SSL/TLS, and hence are assumed not to be affected by the adversary. Also, the adversary cannot tamper with the data that the BAM already reads from the sensor and they can also not conduct MITM or replay attacks on the device. To perform such attacks the attacker needs to sniff the secure communication channel, which requires breaking, e.g., SSL/TLS channel.

To authenticate a new user, AuthentiSense collects a number of enrollment samples (e.g., 3). After the initial setup, AuthentiSense compares the (current) user samples against the enrollment samples. If the samples match (collected samples in the initial setup and current sample belong to the same user), AuthentiSense produces similar embedding vectors, therefore, the Euclidean distance between the computed embedding vectors in the latent space is minimized. AuthentiSense performs a binary classification based on the distance between the embedding vectors. As result of the classification, it outputs the binary label "1" leading to successful user authentication.

AuthentiSense authenticates users with high reliability (see Section V-D); but, in the event of continuous authentication failure, after three unsuccessful attempts, the AuthentiSense automatically falls back to a passive authentication technique (i.e., password) to enforce security.

IV. AuthentiSense

In this section, we first outline the high-level architecture of AuthentiSense and then describe its components in more

detail.

A. High-level Overview

The high level architecture of our system is depicted in Figure 2. The system architecture involves the following components: The Siamese networks (CNN-based), distance function and decision network. At a high-level, the Siamese network functions as a feature extractor generating embedding vectors from input data. Then a Euclidean distance between embedding vectors are computed by the distance function and fed to the decision network for classification.

Siamese network. It consists of two identical sub-networks having the same structure and sharing weights. Each sub-network processes one user behavior and function as a feature extractor learning a meaningful representation of input samples. Two recorded user behavior samples are fed to the sub-networks and transformed into the embedded space learned by the Siamese network.

Distance function. It computes the Euclidean distance between the computed embedding vectors in the latent space. The Euclidean distance here can also be seen as the L_2-norm of the distance vector. The objective of the Siamese network is to maximize the calculated value of this function for negative pairs, i.e., when the input samples belong to different users, and minimize the distance value for positive pairs, i.e., samples from the same user.

Decision network. It consists of fully-connected layers which performs a binary classification based on the distance between the embedding vectors in the latent space. As result of the binary classification, the decision network outputs a binary label, "1" if the captured behavior samples belong to the same user, and "0" otherwise.

B. Design

AuthentiSense is mainly designed for real-world deployment where our focus was on a mobile banking application. We stress that to train the AuthentiSense, diverse user behaviors (i.e., different genders, ages, and occupations) were captured on a real mobile application from a bank where the users provided their signed consent before data collection. Moreover, to demonstrate the practicality of AuthentiSense, the most frequently used functions in the mobile application (according to an analysis conducted on the usage patterns of the bank customers [40]) were identified and used to simulate the user behavior. AuthentiSense consists of three components, each of which has a precise task. In the following, we elaborate on each component in more detail.

Siamese Network. We utilize Siamese neural network to extract highly discriminative features for input data that can distinguish the behavior between genuine and impostor users. Particularly, the Siamese network aims to learn information-rich transformation of the input data into an embedding (latent) space that can preserve distance relation between input data. Suppose we are given a pair of recorded behavior samples as input data; the objective is to map them to an embedding space where the embeddings of two input samples from the same user are closer together and two input data from different users are far apart. The Siamese network architecture, as depicted in

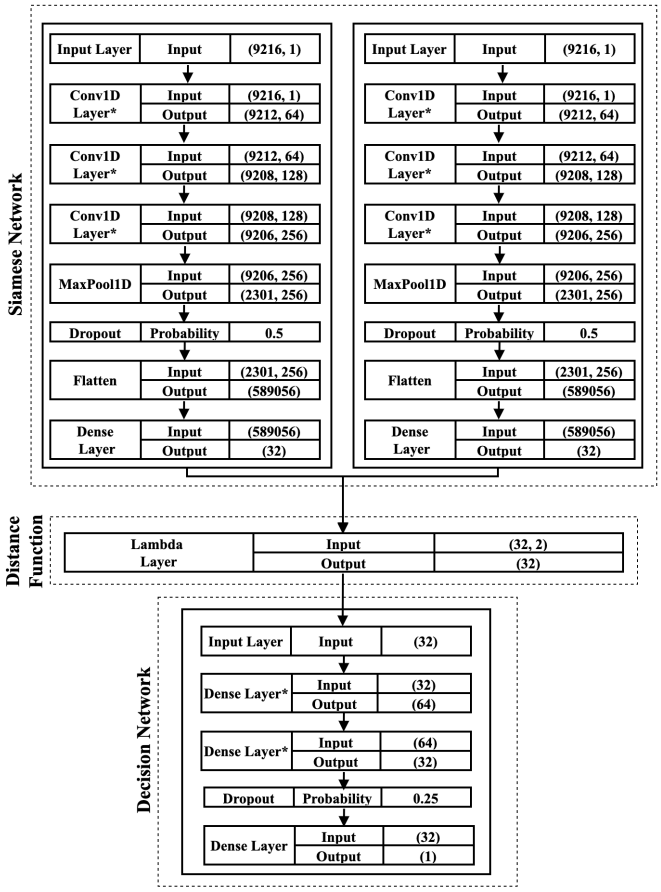


Fig. 3: Model Architecture: Siamese Neural Network followed by the Distance Function and the Decision Network. The layers with $*$ are followed by a Batch Normalization layer.

Figure 3, is made of two identical sub-networks (CNN1D, cf. App. C) which share weights and biases. Each sub-network processes one of the input behavior samples and works as a feature extractor and outputs an encoding of recorded behavior (embedding) using the same filters as the other. We perform L_2-normalization to normalize the embedding vectors and map them to the surface of n-dimensional hyper-sphere of radius 1. This allows to compare the similarity between different inputs by distance between two embedding vectors. As all the embeddings will reside on the surface, this prevents that embeddings with a high L_2-norm distract the decision network.

Distance function. In order to train the Siamese network, we define a triplet loss function (cf. Sect. IV-C) which, for a given pair of input data, takes the distance between the two output embedding vectors from the two sub-networks and regulates large or small distances. The generated embedding vectors are subjected to a distance function (created using a Lambda layer [4]) which computes the L_2-distance. Basically, this distance between embedding vectors should be large enough when the input samples belong to different users but small when they belong to the same user. Among several distance functions, we opted for the L_2-distance. We evaluated other distance metrics (i.e., L_1-distance, and cosine distance), but L_2-distance showed the best results. Utilizing the L_2-distance, we fine-tuned the network parameters (i.e., weights) using back propagation [58].

Decision Network. A fully-connected decision network followed by the distance function makes the classification decision based on the distance between the embedding vectors in the embedding space. The role of the decision network is to solve a binary classification problem and it outputs "1" if the behavior samples belong to the same user, and "0" otherwise. We design a feed forward neural network with three subsequent dense layers with decreasing width in terms of neurons, and ReLU activation function along with a L_2-norm as kernel regulizer. The first-two layers are then followed by a batch normalization layer for regularizing the activations of the prior layer, so that they have mean close to 0 and standard deviation close to 1. The last dense layer, followed by a dropout layer with probability of 0.25, is made of a single unit with sigmoid activation function for being consistent with a Bernoulli output distribution.

As an input, the decision network receives the distance vector of two embedding vectors and produces in output a probability value (i.e., in range $[0, 1]$), which indicates how likely the input data belongs to the same user. Since the point of Equal Error Rare (EER) represents the best classification threshold (because it is the point where False Acceptance and False Rejection rates meet, cf. Sect. V-B), we set the threshold for classification to the point of EER for a set of validation data that were not used for training the neural networks (cf. Fig. 5a), to decide if we will consider the prediction of the decision network as correct.

C. Implementation

In the following, we explain sample preparation to train the Siamese and decision networks. We then elaborate on model building and describe our training strategy and model hyperparameter search and tuning in more detail.

Sample preparation. The Siamese network learning process is based on comparing pairs of behavior samples, namely positive and negative pairs. In positive pairs, two samples belong to the same user, while in negative pairs two samples are from different users. During the model training phase multiple samples of these pairs are fed to the model to minimize the L_2-distance between the embeddings of positive pairs, and to maximize the L_2-distance for negative pairs. One way of selecting the positive and negative pairs is a random selection. This naïve approach has been shown to be good enough for the positive pair selection. However, for negative pairs this naïve approach is problematic as many samples from different users differ significantly, while it is more efficient for training to focus on negative pairs that the network fails to distinguish. Sophisticated techniques such as triplet loss can be utilized for sample preparation, in which three samples simultaneously are used to optimize every training step. In triplet sampling, the first two samples are positive and the last one corresponds to a negative sample. The goal is to minimize the L_2-distance in the embedding space between positive samples and maximize the L_2-distance between positive and negative samples.

Model building. The deep learning part of AuthentiSense uses a Siamese Neural Network (NN) and a decision network. The Siamese NN consists of 2 sub-networks, arranged in the Siamese architecture by assembling layers (in this case, Conv1D, Max Pooling1D, Flatten and Dense layers). Afterwards, the outputs of the two sub-networks are piped through our custom distance function (using a Lambda layer), as shown in Figure 3. In the end, we append the decision network to the Siamese network and build our classifier by stacking fully connected Dense layers.

We use three subsequent convolutional layers with an increasing number of filters (from 64 to 256) in which data is expanded in depth, and with a kernel size that is reduced in the last layer (from 5 to 3) for a more fine-grained computation. Since we are dealing with non-normalized data, after each convolutional layer we add a batch normalization layer to normalize the inputs to a layer for each mini-batch of data. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train the Siamese network [43]. Furthermore, each convolutional layer includes a kernel regularizer (L_2-norm) with a value of 10^{-3}.

After the convolutional layers, we use a 1-dimensional Max Pooling layer for down-sampling the input representation by taking the maximum value over a spatial window of a specified size (in our case 4). Then, we flatten the output of the convolutional block and feed it to the last fully connected (i.e., dense) layer with no activation which is responsible for producing the embedding representation of the input as an array of fixed size (length of 32×1).

Training Strategy. The training procedure depends on the sample generation strategy (i.e., pairwise or triplet). For the pairwise training a contrastive [36] loss is used (cf. App. A) while triplet loss [8] is utilized for triplet training. Triplet loss learns both positive and negative distances simultaneously and focuses more on negative samples that are hard to distinguish reducing the number of easy-distinguishable negative samples. This helps to reduce the risk of overfitting compared to pairwise training. We opt for triplet training, therefore, we train our Siamese network with triplets loss. The objective of triplet loss is to ensure that two samples with the same label have their embeddings close together in the embedding space, while embeddings of two samples with different labels are far away from each other.

To train the network, we sample triplets of desired batch size. As the name implies, three input samples are needed, which are called: i) *Anchor* which is a sample input data, ii) *Positive* that is just another variation of the anchor, and iii) *Negative* which is a different sample from above two similar samples. This helps our model learn dissimilarities with the anchor sample. Positive and negative samples are passed individually to the Siamese network, as triplets are mined online [59]. As depicted in Fig. 4 the network learns to decrease the distance between the anchor and positive, while increase the distance between anchor and negative such that the difference of the two distances would reach to Alpha α which is a pre-defined hyper-parameter (Eq. 1). The alpha parameter in Eq. 1 aims to ensure that the difference between the anchor-positive distance (d(a,p)) and the anchor-negative distance (d(a,n)) is at least as big as a margin equal to alpha to discourage the model from collapsing to trivial solutions where f(a) = f(p) = f(n) which would satisfy Eq. 1. The triplet loss is defined as follows:

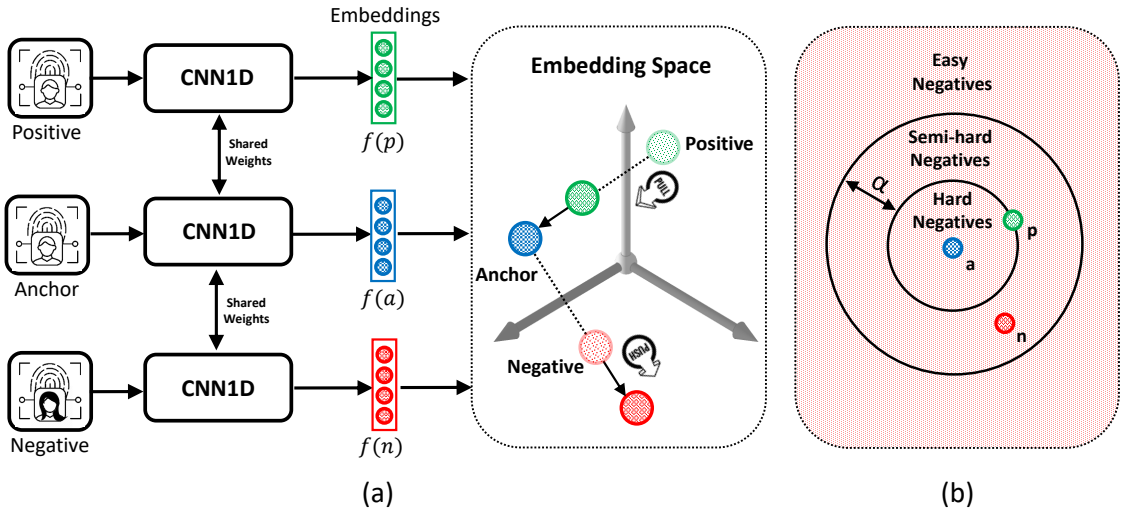


Fig. 4: Training using triplet loss: figure (a) demonstrates how triplet loss pushes positive and anchor samples toward each other and pulls negative sample from anchor. Figure (b) shows Semi-hard triplet selection scheme.

$$\mathscr{L}(a,p,n) = max(||f(a) - f(p)||^2 - ||f(a) - f(n)||^2 + \alpha, 0) \quad (1)$$

In the triplet loss function, the term $||f(a) - f(p)||$ is the distance between the anchor and positive and the term $||f(a) - f(n)||$ is the distance between the anchor and negative. The value of the first term is learned to be smaller while the second term to be bigger. If their subtraction is smaller than minus alpha, the loss would become zero and the network parameters would not be updated at all. During the training, we minimize this loss, which pushes $||f(a) - f(p)||$ to 0 and $||f(a) - f(n)||$ to be greater than $||f(a) - f(p)||$ + α. Based on the definition of the triplet loss, there are three categories of triplets:

- Easy triplets which have a loss of 0, because $||f(a) - f(p)|| + \alpha \leq ||f(a) - f(n)||$
- Hard triplets where the negative is closer to the anchor than the positive, i.e., $||f(a) - f(n)|| \leq ||f(a) - f(p)||$
- Semi-hard triplets where the negative is not closer to the anchor than the positive, but which still have positive loss: $||f(a) - f(p)|| \leq ||f(a) - f(n)|| \leq ||f(a) - f(p)|| + \alpha$

Each of these triplet definitions depends on where the negative sample is located, relatively to the anchor and positive. The categorization (Easy, Hard, and Semi-hard triplets) can therefore be applied to the negative pairs analogously: Hard negatives, Semi-hard negatives or Easy negatives, as shown in Fig. 4. So theoretically, in order to ensure the best effect of network training, we need to choose Hard triplets or Semi-hard triplets. The Hard negatives deliver better losses for model optimization and lead to a strong convergence, but in practice they might be too aggressive and collapse the loss function. Therefore, we opt to use a semi-hard mining strategy in model training.

After defining the loss on triplets of embeddings and observing that some triplets are more useful than others, meaning their loss values help network for a better weight optimization, we select online triplet mining [59] approach to mine triplets. Unlike offline triplet mining where the data is fed as a triplet-form input to the network, in online triplet mining triplets are computed during training within each batch of data. The idea behind online mining is to dynamically compute useful triplets on the fly, for each batch of data. Given a batch of B samples, it computes the B embeddings and then can find a maximum of B^3 triplets[1] and selects the triplets for training that are semi-hard. As already described, we use semi-hard triplet mining to train the network, as semi-hard triplets are not as difficult as hard triplets to learn, but still provide useful information. The mining strategy that we adopt computes triplets online from a batch of randomly drawn samples, therefore, it is impossible to pre-determine the number of triplets.

We also stress that Figure 2 represents an abstract view of the system at inference time, while Figure 4 illustrates the AuthentiSense at training time. We utilized online triplet mining [60] to train our network and unfolded the network structure, in online mining, to explain the inner workings of our system.

End to end model parameters optimization. Having trained the Siamese network with triplet loss, we perform an end to end parameter (i.e., weights) optimization of the decision

[1]However, since each triplet must consist of an anchor, a sample from the same user of the anchor (positive pair), and a sample from a different user (negative pair), many of these naïve combinations are invalid, i.e., do not contain a negative and a positive pair, s.t., the actual number of valid triplets is smaller than B^3 and depends on the batch.

Variable	Setting
Optimizer	Adam, SGD
Learning Rate	$0.1, 0.05, 0.01, 0.005, 0.001, 0.0005$
Batch Size	64, 128, 256, 1024
Margin (Alpha)	$1, 0.5, 0.3, 0.1, 0.05, 0.03, 0.01$

TABLE I: Hyperparameter search for the Siamese Network

Variable	Setting
Optimizer	Adam, SGD
Learning Rate	$0.1, 0.05, 0.01, 0.005$
Batch Size	32, 64, 128

TABLE II: Hyperparameter search for the Decision Network

network using standard backpropagation on the predictions of the entire network including both the Siamese and decision networks. This includes the following steps: First, the weights of the Siamese network are frozen (it is used as a feature extractor), then fully connected decision network is appended to the Siamese network and trained using a binary cross-entropy loss function to optimize the network's weights.

Hyperparameter tuning and network configuration. After constructing the model architecture, we take a Grid Search [6] approach and loop through pre-defined hyperparameters, as shown in Tables I, and II, to choose a set of optimal hyperparameters for the learning algorithms that maximize the model performance. In particular, we investigate multiple options concerning the choice of the optimizer, learning rate, batch size, and the margin value (Alpha) for the Semi-Hard Triplet Loss function. At the end, we select the best performing parameters that maximize the performance of the models, as shown in Tables III and IV and use them for training our networks.

V. Evaluation

In the following we describe the evaluation and also show different alternatives for the design of AuthentiSense.

A. Dataset

To conduct our experiments, we use the DAKOTA Dataset [40], more specifically, the recorded motion sensor values (i.e. accelerometer, gyroscope and magnetometer) for 45 users while using a mobile banking smartphone app. Out of the 45 users, we randomly selected 35 users for training, 3 users as validation data to determine the classification threshold, and 7 users for testing. Data was recorded for every user in 5 sessions for each of the postures *sitting*, *standing* and *phone on table*. Therefore, each user in the training set has 15 sessions, each 90 seconds in length. So it takes only a few volunteer users to train the feature extractor. Furthermore, we have mainly focused on having a few-shot solution for new users joining after the system is deployed. We could have asked the volunteers to provide as much data as we wanted. However, for real-world users, we can only expect to collect a few samples (Table V). As the raw data was sampled with a non-constant frequency and non-uniform starting and ending timestamps for each sensor, we resampled the sensor data for every session at a rate of 5 ms, taking the mean in areas where the data was downsampled and linearly interpolating in areas where the data was upsampled. To obtain a pool of samples, we then ran a sliding window of fixed length (1, 3, 5, 10 and 15 seconds) with step size equal to 1/10th of its length over the data of every recorded session and labeled it with the corresponding user. For each window, the values of every axis of each motion sensor are concatenated to shape a one-dimensional array that is later used to construct positive and negative examples. The pool of windows was then shuffled and split up into batches for the training and testing sets to be used for the Siamese network. For the decision network, an equal number of window pairs by the same user (positive example) and different users (negative example) was randomly sampled to generate 50,000 and 10,000 pairs which were then batched for training and testing, respectively.

Hyperparameter	Setting
Optimizer	Adam
Learning Rate	10^{-3}
Batch Size	$64, 128, 256$
Loss Function	Semi-Hard Triplet Loss
Margin (Alpha)	0.03

TABLE III: Siamese Network hyperparameter settings

Hyperparameter	Setting
Optimizer	Adam
Learning Rate	10^{-4}
Batch Size	64
Loss Function	Binary Cross-entropy

TABLE IV: Decision Network hyperparameter settings

B. Evaluation Metrics

The performance evaluation of AuthentiSense is performed using common metrics. Each metric is based on the number of correctly (TP) and incorrectly (FN) classified benign authentication events as well as the number of correctly detected (TN) and unrecognized (FP) attack attempts. We use the following metrics to evaluate the effectiveness of AuthentiSense.

False Acceptance Rate (FAR) represents the risk to accept attack attempts and is defined as follows:

$$\text{FAR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2}$$

False Rejection Rate (FRR) analogously represents the risk to mistakenly decline benign attempts. It is defined as:

$$\text{FRR} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{3}$$

F1-Score: The F1-Score is harmonic mean of Precision and Recall. The Precision calculates the ratio between the number of positive samples classified correctly, and the total number of samples classified as positive (Eq. 4). Recall defines how many positive samples have been identified correctly overall (Eq. 5).

$$\text{Pr} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4}$$

$$\text{Re} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

The F1-Score is then defined as:

$$\text{F1-Score} = 2 \cdot \frac{\text{Pr} \cdot \text{Re}}{\text{Pr} + \text{Re}} \tag{6}$$

Area Under ROC Curve (AUC): The above-mentioned metrics depend on a threshold that converts the predicted probabilities into binary predictions (accept and reject). The Receiver Operating Characteristic (ROC) curve plots the model's True Positive Rate TPR $= 1 -$ FRR against FAR for different threshold values, as illustrated in Fig. 6. The resulting area between such curve and the X-axis at an interval of $[0, 1]$ is termed AUC and expresses the model's capability of correct classification.

Equal Error Rate (EER): The EER is the value of FAR (and equally FRR) at a threshold value where both FAR and FRR embody equal values.

C. Experimental Setup

All the experiments were conducted on a server running Debian 10, with 1 TB of memory, 64 physical cores/128 threads, provided by an AMD EPYC 7742 processor, and 4 NVIDIA Quadro RTX 8000. We leveraged TensorFlow 2.4.0 [7] to implement the neural networks. We trained our Siamese network using Semi-Hard triplet loss (cf. Sect. IV). We used Contrastive loss as a baseline to compare the results of this loss function with results obtained from the triplet training in AuthentiSense. For all the loss functions, we used the implementation provided by the Tensorflow Addons (TFA) library [8].

D. Evaluation Results

To evaluate the effectiveness of AuthentiSense, we first discuss the performance of AuthentiSense, and then we compare it against the baseline to see how the choice of loss function for training the Siamese network can influence the performance. We also conduct a performance comparison with individual sensor modalities and fusion of modalities.

1) Performance of AuthentiSense: To verify the capabilities and performance of AuthentiSense, we conduct a number of experiments with different setups in the training and recognition phases. We perform a systematic evaluation with different variables such as authentication window time (interaction time required for authentication purpose) and the number of known user samples (from previous history) to compare with (n-shot).

As shown in Tab. V, AuthentiSense achieves a F1-Score of 0.97 in 3-shot verification just in *1 second* of interaction with the user, meaning that for user authentication it only needs 3 enrollment samples. AuthentiSense can also verify users in *1-shot* after 1 second of user interaction at the cost of 0.02 drop in the model performance (F1-Score =0.95). The table also shows that the results vary slightly for an authentication window length of 1s for 4- and 5-shot verification (F1-Score =0.96).

Moreover, Tab. V shows that for 5-shot verification, AuthentiSense effectively authenticates the users for all authentication window lengths. This demonstrates the advantage of the few-shot learning approach that allows AuthentiSense to obtain a satisfactory description of user behavior from the collected information (i.e., the motion patterns), resulting in reliable authentication results, even for corner cases with small number of (n) shots and short interaction time.

		Authentication window length (Sec.)				
		1	3	5	10	15
n-shot	1	0.95	0.88	0.91	0.85	0.85
	2	0.96	0.90	0.92	0.90	0.88
	3	0.97	0.91	0.94	0.92	0.82
	4	0.96	0.92	0.92	0.94	0.95
	5	0.96	0.93	0.94	0.94	0.95

TABLE V: F1-Score for triplet training on test set.

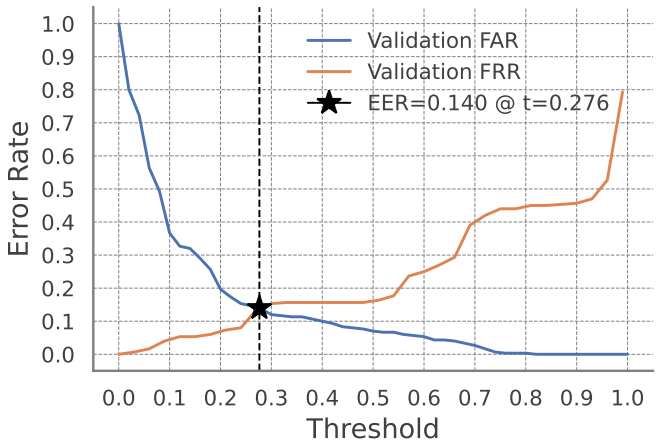
Especially, the shorter authentication window time allows AuthentiSense to perform verification very quickly at recognition stage with only limited amount of data collected during user interaction (t=1s). This confirms the usability of AuthentiSense in practice. The users are not required to interact for a long time with their devices to collect enrollment samples needed for authentication process.

Figure 5b depicts the FAR and FRR for different classification thresholds. For AuthentiSense, the point of EER for the validation data, as shown in Fig. 5a, represents the best threshold (t=0.276) to choose, because it is the point where FAR and FRR are equal. Having applied this threshold on the unseen test data, AuthentiSense achieves a FAR=0.023 and an FRR=0.057. It should be noted that FAR is significantly lower than FRR. Since a false-reject results in an additional request for manual authentication while a false-accept leads to an unauthorized access, thus, having as low FAR as possible is more important for AuthentiSense.

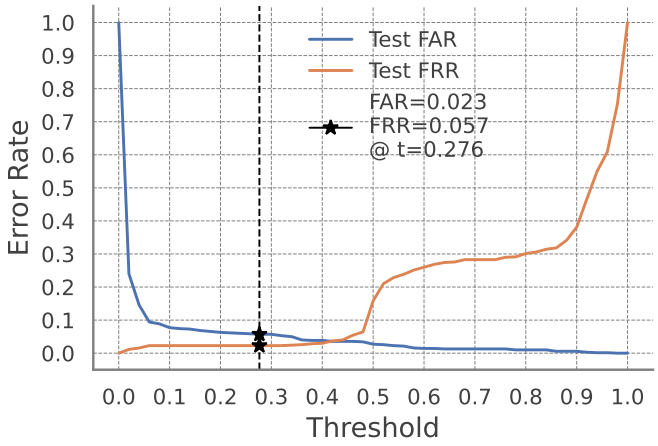
Moreover, Fig. 6 illustrates the Receiver Operating Characteristic (ROC) curve for test samples. The ROC curve shows the dependency between the FAR, FRR and the system's detection threshold. The Area Under Curve (AUC), ranges from 0.5 (random guessing) to 1 (perfect classification), aggregates the system's performance at all threshold settings and acts as an indicator of the model performance. As it can be seen in the figure, AuthentiSense obtains AUC=0.987 for unseen test data.

2) Comparison against baseline: We perform an experiment where we set the contrastive loss as a baseline to compare the model performance in terms of F1-Score against triplet training. Table VI shows the performance of AuthentiSense for contrastive loss function while training its Siamese Network. As shown in the table, the triplet loss outperforms contrastive loss function, for almost all numbers of shots (n-shot) and all authentication window times. Also the best F1-Score for the triplet loss (0.97), as shown in Tab. VI, is higher than the best value of the contrastive loss (0.93). The better efficiency becomes especially visible for challenging authentication scenarios. For example, for an interaction time of 1s and n-shot = 3, the contrastive loss baseline achieves an F1-Score of only 0.92, while AuthentiSense achieves a F1-Score of 0.97. This shows the advantage of the triplet-loss function of AuthentiSense, as here the triplet loss provides the training process of the Siamese network with better choices for negative samples, leading to a more powerful embedding extraction.

3) Individual modality performance: AuthentiSense performs an implicit fusion of the individual sensors' values, i.e., the raw data of multiple sensors (modalities) is consolidated by the CNN before the model extracts any information. This stands in contrast to classical fusion where the data of each sensor is processed separately (i.e., by separate NNs) and the



(a) Calculation of the Equal Error Rate (EER).



(b) Calculation of FAR and FRR.

Fig. 5: Calculation of the EER=0.140 and threshold (t = 0.276) on validation data (a). Computation of FAR=0.023 and FRR=0.057 at the threshold point (t=0.276) on testing data (b).

		Authentication window length (Sec.)				
		1	3	5	10	15
n-shot	1	0.92	0.91	0.90	0.91	0.89
	2	0.92	0.91	0.90	0.91	0.88
	3	0.92	0.91	0.89	0.91	0.86
	4	0.92	0.90	0.92	0.90	0.86
	5	0.93	0.90	0.91	0.90	0.86

TABLE VI: F1-Score for pair training with the contrastive loss.

results of the complete processing pipeline are combined at the end. In Sect. V-D4 we compare the implicit fusion of AuthentiSense against different standard fusion techniques, e.g., summing up or using a Multi-Layer-Perceptron (MLP). In order to assess the performance of each modality individually, in the following first we train AuthentiSense using the data of every sensor. Therefore, we obtain three models trained on three modalities (i.e., accelerometer, gyroscope, and magnetometer). For this assessment, the Siamese Network is

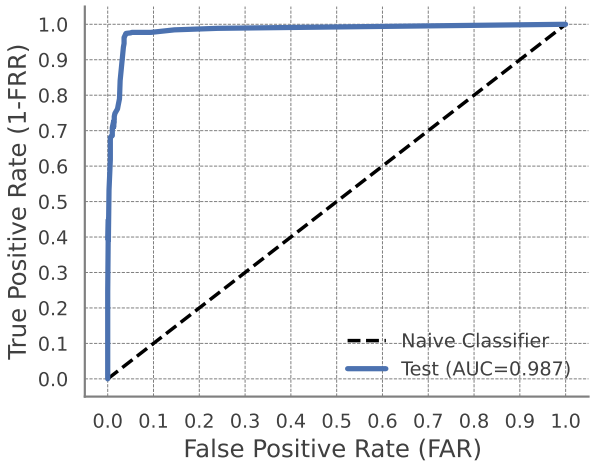


Fig. 6: ROC curve: The area under curve for test samples which serves as a measurement for the model performance.

trained with 100 epochs and a batch size of 1024 while the corresponding decision network is trained with 10 epochs and a batch size of 64.

The results are shown in Fig. 7 and Tab. VII where they are also compared with the performance of AuthentiSense. Both accelerometer- and magnetometer-trained models, although each performing good on the train samples, suffer from a sharp performance decline when evaluated on the test samples. In both cases, the precision remains mostly unchanged above 0.90, while the recall drops by twenty to thirty percentage points (from 0.89 to 0.49 in the case of the magnetometer). The gyroscope-trained model, on the other hand, offers the best performance (F1-Score: 0.80) with relatively stable precision and recall values (0.76 and 0.83 for test samples). Nevertheless, even the gyroscope-trained model is not comparable with the performance of AuthentiSense (F1-Score: 0.97), as shown in Fig. 7.

It must be noted that an optimal F1-Score does not imply an optimal EER (where FAR=FRR), since the threshold for the highest score is sometimes located at a position, where FAR≠FRR. This is also the case in this experiment, as found in Tab. VII. Here, it is the accelerometer-trained model that achieves the best results in regard to EER. Still, the achieved EER of 0.1706 is more than four times worse than the corresponding EER of AuthentiSense. Overall, none of the single-modality-trained models is able to compete with AuthentiSense, affirming the benefits of an architecture that feeds the consolidated data of all motion sensors into a single neural network.

	Modality			AuthentiSense
Samples	Acc.	Gyr.	Mag.	
Test	0.1706	0.2308	0.1902	0.0371

TABLE VII: EERs of the test set of each single-modality trained model and of AuthentiSense.

4) Fusion of modalities: Single-modality-trained models do not perform well on their own, as found in Section V-D3. With the fusion of their matching scores, an additional performance improvement can be attempted. In the following, we

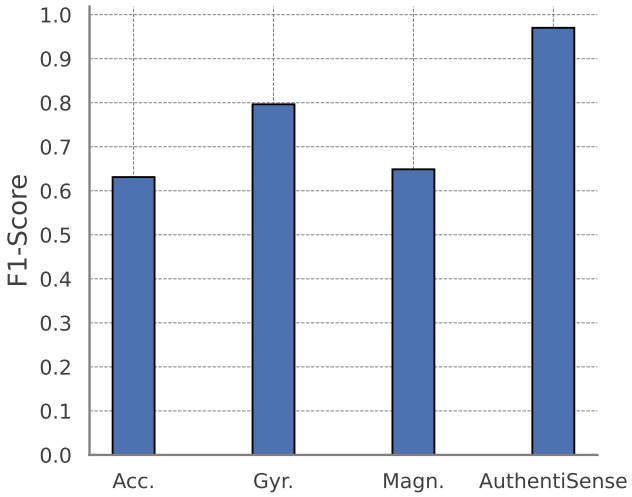


Fig. 7: The F1-Scores of single-modality-trained models and of AuthentiSense.

evaluate the effectiveness of such score level fusions. The performance of fusion is compared both with the best performing single-modality-trained models (described in Sect. V-D3) and with the best performing configuration of AuthentiSense.

We evaluate multiple fusion methods. For each method, different matching score normalization techniques are applied. The individual methods are described in App. D in detail. For training the fusion and normalization methods, we use the scores that were predicted for the 50,000 training samples in Sect. V-D3. For evaluation, the trained fusion methods are applied to the matching scores of the corresponding 10,000 test samples. The results of all utilized fusion techniques in regard to their best performing normalization method are described in Tab. VIII.

Measured by EER, the best results are obtained with EER-weighted sum fusion with min-max normalization (EER: 0.1419). Compared with the best single-modality-trained model (EER: 0.1706; cf. Tab. VII), this experiment shows that score level fusion can offer a performance boost over single-modality based systems in regard to the EER. Such boost, however, is not achieved for the F1-Score. The best results, measured by F1-Score, are obtained with Z-Score normalized MLP Fusion. This technique's F1-Score surpasses the scores of the accelerometer and magnetometer trained models. However, it is slightly below the score of the gyroscope trained model (cf. Fig. 7). In general, all fusion methods reveal high precision and low recall values, as very similarly experienced with the accelerometer and magnetometer trained single-modality models in Sect. V-D3. Overall, the improvement of score level fusion over single-modality-trained models is only marginal. The performance of AuthentiSense (F1-Score: 0.97; EER: 0.0371) is not reached by any of the single-modality-trained models (best F1-Score: 0.80; best EER: 0.1706). With only marginal improvement of score level fusion over the latter (best F1-Score: 0.76; best EER: 0.1419), score level fusion is equally incapable of achieving significantly better results. It must therefore be acknowledged that, for the use cases discussed in this work, our proposed sensor level fusion approach of AuthentiSense constitutes a substantial advantage over score level fusion.

Fusion	Normalization	F1-Score	EER
Linear Reg.	Not normalized	0.69	0.1454
Logistic Reg.	Not normalized	0.69	0.1453
MLP	Z-Score	0.76	0.1451
Sum	Min-Max	0.75	0.1532
EER-W. Sum	Min-Max	0.67	0.1419
AuthentiSense		0.97	0.0371

TABLE VIII: Performance of all utilized fusion methods. For each fusion method, only the best performing normalization method is listed.

VI. Related Work

In the past, different approaches for authenticating the users of mobile devices have been developed. These approaches use either statistical features extracted on touch gestures [52], [67], [33], [69], [68], [44], [26], values that are captured by the sensors for identifying the user based on its motions [51], [38], [18], [22], [14], [70], [50], [37], [15], [30], [49], [65], [25], or correlate the touch gestures and motion sensors [29], [16], [20], [21]. However, these existing approaches are 1) restricted to identifying a user among a set of known users, thus can only detect intruders that were part of the training data [30], [33], [52], [65], [67], 2) require training the model when a new user joins the system limiting their scalability [15], [24], [70], or 3) work only in certain situations and not in a continuous way, e.g., when picking up the phone [19], [20], [21]. Compared to the existing works, by combining different data sources (accelerometer, gyroscope, magnetometer) with a few-shot learning approach, AuthentiSense can authenticate the user reliably, while requiring only a minimum of user interaction. In the following, we discuss the existing approaches in more detail and compare them with AuthentiSense.

A. Touch and Typing Behavior

Lu *et al.* [52], Xu *et al.* [67], and Frank *et al.* [33] collect touch-event features such as the position (X and Y coordinates), pressure and gesture velocity for training a Support-Vector-Machine (SVM). Lu *et al.* [52] and Xu *et al.* [67] use training data from the benign user and another user that they define as the attacker for training the SVM to distinguish them. The SVM Frank *et al.* is trained for recognizing the user out of a set of known users [33]. However, these approaches are not practical, since they cannot detect unknown attackers but only distinguish between known persons. Therefore, they cannot reliably prevent unknown users from accessing the system as the respective scheme will recognize unknown intruders as one of the users from training, whose behavior is most similar to the intruders' behavior. AuthentiSense addresses this issue by using a few-shot learning approach that detects arbitrary intruders even if no samples of the benign user were part of the training data.

Zhao *et al.* convert touch features (position, pressure etc.) into an image, called Graphic Touch Gesture Feature (GTGF) [68]. An extension uses a statistical feature model on the features to make the system more robust against variations in the benign behavior [69]. However, these approaches are limited since they require the user to perform specific gestures on the phone for authentication, e.g., swiping from

the top-right corner to the bottom-left corner. In contrast, AuthentiSense continuously analyses the sensor values for identifying the user without requesting any specific gesture patterns.

Chen *et al.* combine features that are extracted from the touch gestures with acoustical sensor data [26]. However, their similarity technique requires a comprehensive set of comparison samples, while for AuthentiSense less than 5 shots are sufficient. Similarly, also the approach of Karanikiotis *et al.* requires a high amount of training data for each user to extract standard features from the touch gestures, like the swiping duration [46] and apply a SVM.

The solution by Islam *et al.* requires the users to solve a challenge, e.g., draw a circle [44]. However, with this explicit authentication action, there is no advantage compared to standard explicit authentication approaches, e.g., scanning the fingerprint. If this is performed continuously, such a scheme is likely to disturb the user, causing the user to turn-off the authentication scheme. In comparison, AuthentiSense can run unobtrusively in the background and becomes only visible, when the user is not recognized and the device is looked.

Further approaches analyze the mobile keystroke dynamics [42], [35], [23], [10], [57], [63]. However, keystroke dynamics are not well-suitable for continuous authentication on mobile platforms, as mobile apps do not frequently involve keyboard-based user interactions. In contrast, AuthentiSense utilizes only motion patterns (sensor values) and can frequently and reliably authenticate users.

B. Motion-based Approaches

In order to obtain a more augmented set of machine learning features, the CNN-based approach SCANet [51] and its predecessor CNNAuth [38] perform a transformation of the temporal sensor data into the frequency domain and utilize a CNN that is trained to recognize the legitimate user. However, since both approaches train CNNs for each user individually, they require a large amount of training data during the system enrollment. In comparison, the few-shot approach of AuthentiSense allows an effective authentication of the user and requires only very few user samples, e.g., just a single sample of the current user for comparing the input without the need for subsequent training which would require hundreds of samples.

Buriro *et al.* proposed an approach that uses the sensor data of the mobile phone for authenticating the user after unlocking the phone, assuming that the movements in this scenario are always similar for the same user [22]. In comparison to AuthentiSense that performs continuous authentication while the phone is being used, the approach of Buriro *et al.* only authenticates the user once, i.e., right after the phone is unlocked, s.t. their approach is orthogonal to AuthentiSense.

ActiveAuth uses a Gaussian Data Description verifier for identifying the user [18]. Analogously to the previous approach, also ActiveAuth does not perform continuous authentication but authenticates the user only in certain situations, e.g., when uninstalling an application.

DeepAuth applies a user-specific RNN on data from the accelerometer and gyroscope [14]. By using Long-Short-Term-Memories (LSTMs), the system can effectively process time-series data [14]. However, for training the user-dependent RNN, DeepAuth first needs to collect a sufficient amount of data, while AuthentiSense just requires few comparison samples, making AuthentiSense more practical.

Zhu *et al.* use an n-gram Markov model based on data from accelerometer, gyroscope and magnetometer to authenticate the user [70]. In contrast to the user-agnostic approach of AuthentiSense, their model is user-specific and requires training efforts for each new user.

Lee *et al.* proposed an approach that identifies users based on the motion that occurs while their phone is being picked up from a surface [50]. A similar system is also introduced by Haring *et al.* [37]. Compared to AuthentiSense, their approach is is only effective in a single use-case, i.e., when a user picks up the phone. Therefore, it cannot provide continuous nor passive authentication while AuthentiSense provides both authentication types.

Discrete motion is also generated when a user taps the phone's touchscreen. The authors of [15] capture the motion sensor data of such events and feed a CNN model with it. An SVM is trained on the generated CNN features as binary classifier using data from the legitimate user and a non-legitimate user [15]. However, because the SVM is trained only to distinguish the legitimate user and the data from other people that was used during training, it cannot detect an intruder whose behavior was not covered by the training data and is, e.g., more similar to the benign user than to the non-legitimate user that was used during training.

The authors of [49] propose an authentication system that enriches the motion data of phones with additional motion information obtained from user-owned wearable devices such as Smartwatches. While this system does not only require the presence of a wearable device (the performance is considerably degraded without such auxiliary information), it also requires an available cloud to which the user's motion data must be sent for training. In contrast, AuthentiSense requires neither subsequent training nor additional devices or cloud services.

In order to authenticate users continuously over a longer period, Ehatisham-ul-Haq *et al.* utilize a phone's motion sensors to detect different types of human activity and identify users based on their everyday activity patterns using an SVM [30]. Wang *et al.* use an accelerometer to identify the user out of a set of known users, also using a Siamese network [65]. However, their approach is not suitable for authentication as it cannot distinguish between the benign user and unknown users which were not present in the training data. Centeno *et al.* train a One-Class SVM to identify the benign user after extracting features from sensor data using a Siamese network [25]. However, their approach does not scale, because for training the One-class SVM, a high number of training samples for each user is needed, while AuthentiSense requires only few samples for comparison.

C. Touch-Motion Behavior

Deb *et al.* [29] combine raw horizontal and vertical scrolling logs for touch behaviour, and Fourier-transformed accelerometer, gyroscope or magnetometer data. For each

modality, they train a separate LSTM network using contrastive loss, and modalities are combined with score level fusion. However, they only train a distance threshold, which, given the high variance of behavior data, does not solve the scalability problem.

Buriro *et al.* proposed a sensor-augmented authentication model during for PIN entry under different user positions, using summary statistic representations for sensor data and inter-key latency for fixed-length PIN entries, fed into binary classifiers [21]. However, since they analyze the user-behavior during the PIN insertion their approach focuses on improving the security of classical authentication methods. In comparison, AuthentiSense performs a continuous authentication allowing to detect intruders also after the regular authentication. Therefore, the approach of Buriro *et al.* is orthogonal to AuthentiSense.

In a follow up work, Buriro *et al.* [20] used a one-class Multilayer Perceptron on a behavioural dataset of 30 users signing on their touch screens along with sensor data. For each user, a Multilayer Perceptron was trained with owner data only, without any impostor samples. However, analogously to the previous work, also this approach cannot be applied continuously. AnswerAuth analyses the phones' sensor data while the user is performing certain actions, e.g., like sliding or lifting the phone. AnswerAuth applies a Random-forest classifier to identify the current user out of a set of known users [19]. However, since AnswerAuth can only identify a person from a set of known users, for which training samples were recorded, it cannot identify intruders.

Multiple approaches train models for each user individually for authenticating them based on touch-gestures and sensor data. Incel *et al.* use sensor and touch data to authenticate users in a banking app. They trained binary SVMs with RBF kernels to authenticate users [41]. Humayoun *et al.* train a DNN to combine features that are extracted from touch-gestures with sensor data [39]. Abuhamad *et al.* [9] utilize LSTM Networks to authenticate users with short authentication windows at a high frequency. Acien *et al.* use touch, accelerometer, gyroscope, keystroke, WiFi, GPS, and App Usage data to authenticate user. They profiled users by training separate RBF-SVM classifiers for each user and each modality, ending up with seven models for every single user, combined with score-level fusion [11]. However, since these approaches train separate models for each user, they need to collect a high number of training samples during the enrollment phase. Since normal users are unlikely to perform certain gestures for many times in during the setup, these approaches are not scalable.

To summarize, AuthentiSense utilizes only standard built-in sensors (i.e., accelerometer, gyroscope, magnetometer) for authentication purpose. These sensors are typically available on mobile devices, therefore, AuthentiSense can work in most settings. Furthermore, AuthentiSense is trained in a user-agnostic fashion and even works for users it is not trained for. It is scalable and allows users to be on-boarded with only a small number of enrollment samples without any further training. The use of a feature extractor network also makes it possible to enroll users by only storing behavioral feature vectors rather than raw behavior data, which makes it less privacy invasive and reduces the total authentication overhead.

VII. Discussion

Behavior biometrics systems are becoming more widespread and effective as technology advances, they are not a bullet-proof solution for authentication or identification. In the following, we outline some of the limitations of behavioral biometrics authentication systems including AuthentiSense.

Behavioral biometrics authentication systems may suffer from failure to enrol. This happens when a reference sample for biometrics cannot be successfully created at the time of enrolment due to a number of factors, such as low-quality sensors, poor environmental conditions, physical or medical conditions of the individuals [2]. Ensuring effective enrolment is crucial to the successful operation of a biometrics authentication system.

The use of behavioral biometrics, as with other security measures, has vulnerabilities and can be compromised. The sensor data can be spoofed or retrieved through side-channel attacks. However, behavioral biometrics spoofing can be mitigated by hardware security extension technologies or through sensor data obfuscation.

Another limitation of behavioral biometrics systems is that unlike traditional authentication methods (i.e., passwords), behavioral biometric characteristics cannot be reissued or cancelled. In case of being compromised, if not impossible it can be extremely difficult to change the characteristics. This makes it problematic when using those behavioral biometric characteristic for future authentication.

The matching of an individual with a reference information stored in the system is a probabilistic calculation. There are errors (i.e., false acceptance and rejection rates) that may be influenced by a range of factors. The user interaction with a sensor may differ between the enrollment and recognition phases or, in rare cases, individuals may share similar behavioral biometric characteristics. In addition, factors such as aging, or medical conditions can also affect individuals' behavioral biometric characteristic between the enrollment and recognition stages.

Behavioral biometrics like many other technologies can pose challenges to privacy. Since in behavioral biometrics authentication systems the information collection is covert or passive, individuals may be unable to provide consent or control over what information is collected or how it is used. Another privacy risk is depending on the characteristics, some behavioral biometrics (i.e., motion patterns) could potentially reveal secondary information (i.e., health-related issues) about an individual who may not want to provide that information.

Furthermore, during training of AuthentiSense, our loss function does not take the context into account, and triplets are formed using only "subject" information, where it is possible to encounter a positive sample from a very different context than the anchor. For instance, in the anchor, the user could be on one device and scrolling, and in the positive sample, the user could be typing on a different device. In this case, it would be difficult for the network to learn a function that can recognize the similarities between the two samples.

In future work, a triplet mining procedure can be implemented to consider more than label information when selecting

positive and negative samples, such as posture, device model, or any other relevant *contextual* information which could affect the behavior samples collected. We hypothesize that a feature extractor model may create more compact clusters for different devices or postures, if positive samples are only drawn for the same context, rather than one big loose cluster for all of a given user's behavior samples.

VIII. Conclusion

We propose AuthentiSense, a user-agnostic behavioral biometrics authentication system that continuously uses motion patterns while interacting with mobile apps and regularly validates the authenticity of a user after the user has logged in. The passive nature of AuthentiSense makes it non-intrusive to the users' experience and takes the burden off the users, offering a frictionless and quick authentication method. We utilize a few-shot learning-based model called Siamese network and train an efficient model that is not user-specific. Our proposed approach is highly scalable and fast. It does not require handcrafted features for model training and does not need to be re-trained when users are dynamically changing (i.e., joining or leaving the system). When evaluated in a Few-shot fashion, our system needs only a few behavior samples per user. We conduct an extensive and systematic measurement study and analyze the impact of different parameters such as choice of loss functions, size of the authentication window time, and n-shots. Our evaluation results demonstrate that AuthentiSense can achieve an accuracy of 97% in terms of F1-score for 3-shot verification and can accurately authenticate users already after 1 second of user interaction.

Acknowledgment

We would like to thank Intel Private AI center and BMBF for their support of this research.

References

[1] Behavioral biometrics, 2022. https://finance.arvato.com/en/industries/digital-business/behavioral-biometrics/.

[2] Biometrics and privacy - issues and challenges, 2022. https://ovic.vic.gov.au/privacy/biometrics-and-privacy-issues-and-challenges/.

[3] Dautsche bank, 2022. https://internationaldirector.com/finance/how-banks-can-achieve-protective-yet-intuitive-security/.

[4] Lambda layer, 2022. https://keras.io/api/layers/core_layers/lambda/.

[5] Mastercard, 2022. https://www.mastercard.com/news/perspectives/2021/behavioral-biometrics-explained/.

[6] Scikit-learn: Machine learning in Python, 2022. https://scikit-learn.org/stable/.

[7] Tensorflow, 2022. https://tensorflow.org.

[8] Tensorflow addons, 2022. https://www.tensorflow.org/addons.

[9] Abuhamad, M., Abuhmed, T., Mohaisen, D., and Nyang, D. Autosen: Deep-learning-based implicit continuous authentication using smartphone sensors. *IEEE Internet of Things Journal 7*, 6 (2020), 5008–5020.

[10] Acien, A., Morales, A., Vera-Rodriguez, R., and Fierrez, J. Keystroke mobile authentication: Performance of long-term approaches and fusion with behavioral profiling. In *Iberian Conference on Pattern Recognition and Image Analysis* (2019).

[11] Acien, A., Morales, A., Vera-Rodríguez, R., and Fiérrez, J. Multilock: Mobile active authentication based on multiple biometric and behavioral patterns. *CoRR abs/1901.10312* (2019).

[12] Akhtar, Z., Micheloni, C., Piciarelli, C., and Foresti, G. L. Mobio_livdet: Mobile biometric liveness detection. In *IEEE International Conference on Advanced Video and Signal Based* (2014).

[13] Alsaade, F. *Score-level fusion for multimodal biometrics*. PhD thesis, University of Hertfordshire, 2008.

[14] Amini, S., Noroozi, V., Pande, A., Gupte, S., Yu, P. S., and Kanich, C. Deepauth: A framework for continuous user re-authentication in mobile apps. In *ACM International Conference on Information and Knowledge Management* (2018).

[15] Benegui, C., and Ionescu, R. T. Convolutional neural networks for user identification based on motion sensors represented as images. In *IEEE Access* (2020).

[16] Bo, C., Zhang, L., Li, X.-Y., Huang, Q., and Wang, Y. Silentsense: silent user identification via touch and movement behavioral biometrics. In *Annual international conference on Mobile computing & networking* (2013).

[17] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems* (1994).

[18] Buriro, A. *Behavioral biometrics for smartphone user authentication*. PhD thesis, University of Trento, 2017.

[19] Buriro, A., Crispo, B., and Conti, M. Answerauth: A bimodal behavioral biometric-based user authentication scheme for smartphones. *Journal of information security and applications 44* (2019), 89–103.

[20] Buriro, A., Crispo, B., Delfrari, F., and Wrona, K. Hold and sign: A novel behavioral biometrics for smartphone user authentication. In *IEEE security and privacy workshops (SPW)* (2016).

[21] Buriro, A., Crispo, B., Frari, F. D., and Wrona, K. Touchstroke: Smartphone user authentication based on touch-typing biometrics. In *International Conference on Image Analysis and Processing* (2015).

[22] Buriro, A., Crispo, B., and Zhauniarovich, Y. Please hold on: Unobtrusive user authentication using smartphone's built-in sensors. In *IEEE International Conference on Identity, Security and Behavior Analysis (ISBA)* (2017).

[23] Buschek, D., Bisinger, B., and Alt, F. Researchime: A mobile keyboard application for studying free typing behaviour in the wild. In *CHI Conference on Human Factors in Computing Systems* (2018).

[24] Centeno, M. P., Guan, Y., and van Moorsel, A. Mobile based continuous authentication using deep features. In *International Workshop on Embedded and Mobile Deep Learning (EMDL)* (2018).

[25] Centeno, M. P., Guan, Y., and van Moorsel, A. Mobile based continuous authentication using deep features. In *2nd International Workshop on Embedded and Mobile Deep Learning* (2018).

[26] Chen, H., Li, F., Du, W., Yang, S., Conn, M., and Wang, Y. Listen to your fingers: User authentication based on geometry biometrics of touch gesture. In *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2020), vol. 4.

[27] Czajka, A., and Pacut, A. Replay attack prevention for iris biometrics. In *42nd Annual IEEE International Carnahan Conference on Security Technology* (2008).

[28] Das, A., Borisov, N., and Caesar, M. Tracking mobile web users through motion sensors: Attacks and defenses. In *NDSS* (2016).

[29] Deb, D., Ross, A., Jain, A. K., Prakah-Asante, K., and Prasad, K. V. Actions speak louder than (pass) words: Passive authentication of smartphone* users via deep temporal features. In *international conference on biometrics (ICB)* (2019).

[30] Ehatisham-ul Haq, M., Azam, M. A., Naeem, U., Amin, Y., and Loo, J. Continuous authentication of smartphone users based on activity pattern recognition using passive mobile sensing. In *Journal of Network and Computer Applications* (2018).

[31] Fei-Fei, L., Fergus, R., and Perona, P. One-shot learning of object categories. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2006).

[32] Fink, M. Object classification from a single example utilizing class relevance metrics. In *Advances in Neural Information Processing Systems* (2004), L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17, MIT Press.

[33] Frank, M., Biedert, R., Ma, E., Martinovic, I., and Song, D. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. In *IEEE transactions on information forensics and security* (2012), vol. 8.

[34] GOODFELLOW, I., BENGIO, Y., AND COURVILLE, A. *Deep learning*. MIT press, 2016.

[35] GURARY, J., ZHU, Y., ALNAHASH, N., AND FU, H. Implicit authentication for mobile devices using typing behavior. In *International Conference on Human Aspects of Information Security, Privacy, and Trust* (2016).

[36] HADSELL, R., CHOPRA, S., AND LECUN, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (2006).

[37] HARING, M., REINHARDT, D., AND OMLOR, Y. Pick me up and i will tell you who you are: Analyzing pick-up motions to authenticate users. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (2018).

[38] HU, H., LI, Y., ZHU, Z., AND ZHOU, G. Cnnauth: continuous authentication via two-stream convolutional neural networks. In *2018 IEEE international conference on networking, architecture and storage (NAS)* (2018).

[39] HUMAYOUN, S. R., ABBAS, G., AND AL-TARAWNEH, R. Touch-behavioral authentication on smartphones using machine learning. In *27th International Conference on Intelligent User Interfaces* (2022), pp. 105–108.

[40] INCEL, Ö. D., GÜNAY, S., AKAN, Y., BARLAS, Y., BASAR, O. E., ALPTEKIN, G. I., AND ISBILEN, M. Dakota: Sensor and touch screen-based continuous authentication on a mobile banking application. In *IEEE Access* (2021), vol. 9.

[41] INCEL, O. D., GÜNAY, S., AKAN, Y., BARLAS, Y., BASAR, O. E., ALPTEKIN, G. I., AND ISBILEN, M. Dakota: Sensor and touch screen-based continuous authentication on a mobile banking application. *IEEE Access 9* (2021), 38943–38960.

[42] INGUANEZ, F., AND AHMADI, S. Securing smartphones via typing heat maps. In *International Conference on Consumer Electronics-Berlin (ICCE-Berlin)* (2016).

[43] IOFFE, S., AND SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (2015).

[44] ISLAM, M. M., SAFAVI-NAINI, R., AND KNEPPERS, M. Scalable behavioral authentication. In *IEEE Access* (2021), vol. 9.

[45] JOSEPH, S., KELVIN, B., SHELDON, A., LASANIO, S., JOSHUA, A., DERRICK, L., ANIESHA, A., KARL, R., AND GERRY, D. Genetic & evolutionary biometric security: Disposable feature extractors for mitigating biometric replay attacks. In *Procedia Computer Science* (2012).

[46] KARANIKIOTIS, T., PAPAMICHAIL, M. D., CHATZIDIMITRIOU, K. C., OIKONOMOU, N.-C. I., SYMEONIDIS, A. L., AND SARIPALLE, S. K. Continuous implicit authentication through touch traces modelling. In *International Conference on Software Quality, Reliability and Security (QRS)* (2020).

[47] KENNETH, R. Behavioral biometrics. John Wiley & Sons, Ltd.

[48] KIRANYAZ, S., AVCI, O., ABDELJABER, O., INCE, T., GABBOUJ, M., AND INMAN, D. J. 1d convolutional neural networks and applications: A survey. In *Mechanical Systems and Signal Processing* (2021), vol. 151.

[49] LEE, W.-H., AND LEE, R. B. Implicit smartphone user authentication with sensors and contextual machine learning. In *Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2017).

[50] LEE, W.-H., LIU, X., SHEN, Y., JIN, H., AND LEE, R. B. Secure pick up: Implicit authentication when you start using the smartphone. In *ACM Symposium on access control models and technologies* (2017).

[51] LI, Y., HU, H., ZHU, Z., AND ZHOU, G. Scanet: sensor-based continuous authentication with two-stream convolutional neural networks. In *ACM Transactions on Sensor Networks (TOSN)* (2020), vol. 16.

[52] LU, L., AND LIU, Y. Safeguard: User reauthentication on smartphones via behavioral biometrics. In *IEEE Transactions on Computational Social Systems* (2015), vol. 2.

[53] NEVEROVA, N., WOLF, C., LACEY, G., FRIDMAN, L., CHANDRA, D., BARBELLO, B., AND TAYLOR, G. Learning human identity from motion patterns. In *IEEE Access* (2016).

[54] P. SHRESTHA, M. M., AND SAXENA, N. Slogger:smashing motion-based touchstroke logging with transparent system noise. In *ACM WiSec* (2016).

[55] QUIGLEY, C. Conjoined twins. In *Encyclopedia of Applied Ethics (Second Edition)* (2012), Academic Press.

[56] ROSS, A. A., NANDAKUMAR, K., AND JAIN, A. K. *Handbook of multibiometrics*, vol. 6. Springer Science & Business Media, 2006.

[57] ROY, S., SINHA, D., AND ROY, U. User authentication: Keystroke dynamics with soft biometric features. In *Internet of Things (IoT): Technologies, Applications, Challenges and Solutions* (2017).

[58] RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. Learning representations by back-propagating errors. In *nature* (1986), vol. 323, Nature Publishing Group.

[59] SCHROFF, F., KALENICHENKO, D., AND PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In *IEEE conference on computer vision and pattern recognition* (2015).

[60] SHARMA, M., AND ELMILIGI, H. Behavioral biometrics: Past, present and future. In *Recent Advances in Biometrics* (2022), IntechOpen.

[61] SHEN, C., YU, T., YUAN, S., LI, Y., AND GUAN, X. Performance analysis of motion-sensor behavior for user authentication on smartphones. In *Sensors* (2016).

[62] SITOVÁ, Z., ŠEDĚNKA, J., YANG, Q., PENG, G., ZHOU, G., GASTI, P., AND BALAGANI, K. S. Hmog: New behavioral biometric features for continuous authentication of smartphone users. In *IEEE Transactions on Information Forensics and Security* (2016).

[63] STANCIU, V.-D., SPOLAOR, R., MAURO, C., AND CRISTIANO, G. On the effectiveness of sensor-enhanced keystroke dynamics against statistical attacks. In *6th ACM conference on data and application security and privacy* (2016).

[64] VOLAKA, H. C., ALPTEKIN, G., BASAR, O. E., ISBILEN, M., AND INCEL, O. D. Towards continuous authentication on mobile phones using deep learning models. In *Procedia Computer Science* (2019).

[65] WANG, C., XIAO, Y., GAO, X., LI, L., AND WANG, J. A framework for behavioral biometric authentication using deep metric learning on mobile devices. In *IEEE Transactions on Mobile Computing* (2021).

[66] WANG, Y., YAO, Q., KWOK, J. T., AND NI, L. M. Generalizing from a few examples: A survey on few-shot learning. In *ACM Comput. Surv.* (2020), vol. 53, Association for Computing Machinery.

[67] XU, H., ZHOU, Y., AND LYU, M. R. Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones. In *10th Symposium On Usable Privacy and Security (SOUPS 2014)* (2014).

[68] ZHAO, X., FENG, T., AND SHI, W. Continuous mobile authentication using a novel graphic touch gesture feature. In *IEEE sixth international conference on biometrics: theory, applications and systems (BTAS)* (2013).

[69] ZHAO, X., FENG, T., SHI, W., AND KAKADIARIS, I. A. Mobile user authentication using statistical touch dynamics images. In *IEEE Transactions on Information Forensics and Security* (2014), vol. 9.

[70] ZHU, J., WU, P., WANG, X., AND ZHANG, J. Sensec: Mobile security through passive sensing. In *International conference on computing, networking and communications (ICNC)* (2013).

APPENDIX

A. Contrastive loss

In contrastive loss, two data samples are fed into the Siamese networks one after the other to get embedding vectors. Then in the latent embedding space, the distance D between the two embedding vectors are computed. Finally, the calculated distance D is substituted into the loss function (Eq. 7) and the Siamese network is trained via backpropagation for better latent vector embedding. The loss function is defined as below:

$$\mathscr{L}(Y,D) = (Y)\cdot(D^2)+(1-Y)\cdot max(margin - D, 0)^2 \quad (7)$$

Where the Y value is ground truth label. $Y = 1$ if the input pairs are of the same user, and $Y = 0$ otherwise. The max function takes the largest value of 0 and the margin (a pre-defined hyper-parameter) minus the distance.

B. Neural Network

Neural Networks use a so-called training dataset to learn a specific task. They take samples from a domain $\mathscr{D}$, e.g., images or sensor values, as input and predict an output vector l indicating e.g., the recognized category, from a target set $\mathscr{L}$. The parameters of a NN, therefore, realize a function $f : \mathscr{D} \mapsto \mathscr{L}$. It should be noted that l can also be a binary label, then $\mathscr{L} = \{0, 1\}$.

NNs consist of individual neurons, which are grouped into layers. The input of the NN is given to first layer, that determines for each neuron an activation status. The activation status of each neuron of the current layer is then given to the neurons of the next layer. The activation $n_{l,i}$ of a (linear) neuron i in layer l, which uses the activations of K neurons from the previous layer as input is then given by:

$$n_{l,i} = g(b_{l,i} + \sum_{k=0}^{K} w_{l,i,k} \cdot n_{l-1,k}) \tag{8}$$

where $b_{l,i}$ and $w_{l,i,k}$ are parameters of the NN that are optimized during the training phase for the respective task. g is the so-called activation function, a non-linear function as $\text{relu}(x) = \max(0, x)$ that is used to determine activation of the current neuron and allows the NN to recognize also non-linear patterns [34].

C. 1-dimensional Convolutional Neural Network.

In Convolutional Neural Networks (CNNs) each neuron only uses a small window of neighboring inputs to determine its activation status, e.g., neighboring time steps for sequential data [34]. CNNs are taking their name from the mathematical tool called convolution, which is a linear operator used for feature extraction where a small filter or a kernel, is slided across an input sequence. Iterating on the input piece by piece allows the model to extract small features producing in output a feature map, that depends on both the input values and the kernel weights. Different kernels can therefore be thought of as different feature extractors.

Although CNNs are most popularly used for two-dimensional inputs like images, they can be generalized to other dimensions, and one-dimensional CNNs (or CNN1D) are more suitable for certain applications dealing with sequential data like one-dimensional sensor readings [48]. Furthermore, another advantage of CNN1D is that they have less trainable parameters (i.e., weights), therefore, they are less complex and faster to train compared to 2D-CNNs allowing smaller sets of training data, making them more suitable for real-time and low-cost applications [48].

D. Score Fusion

In the field of behavioral biometrics, information from different sensors is not always processed by a single classifier (sensor level fusion). Instead, multiple classifiers (matchers) are typically used that each produce a matching score. To form a final classification, a vector of matching scores $s = [s_1, \ldots, s_R]$, obtained from R matchers, is fused using special techniques (e.g., score level fusion) in order to classify samples as either *genuine* or *impostor* [56]. Having a broader collection of information available, score fusion models can potentially achieve better performance than each of the individual matchers does on their own. In the following, we describe a selection of fusion and normalization techniques.

1) Normalization Techniques: simple rule-based fusion techniques require matching scores to lie in a common range. This is achieved through normalization methods. Each method has different impact on the fusion technique's performance.

The Z-Score Normalization: is a method that normalizes the j^{th} matcher by utilizing arithmetic mean (μ_j) and standard deviation (σ_j) of the available match scores. The normalized value of the i^{th} match score s_j^i, as specified in [56], is

$$ns_j^i = \frac{s_j^i - \mu_j}{\sigma_j}. \tag{9}$$

where feature distributions are centered around zero with unit standard deviation.

Min-Max Normalization: Min-Max normalization maps values between 0 and 1. Just as z-score normalization, this method is not robust, because outliers can be mapped to the outside of the lower and upper boundaries. Let s_j^i, as defined in [56], be the j^{th} matcher's i^{th} matching score of the N scores that are used for tuning the normalizer. Then the normalized value of the i^{th} match score s_j^i is

$$ns_j^i = \frac{s_j^i - \min_{k=1}^{N} s_j^k}{\max_{k=1}^{N} s_j^k - \min_{k=1}^{N} s_j^k}. \tag{10}$$

where 0 corresponds to the minimum observation and 1 corresponds to the maximum

2) Fusion Techniques: Fusion techniques include (i) weighted sum fusion, (ii) EER-weighted sum fusion, (iii) linear and logistic regression fusion as well as (iv) multi-layer perceptron (MLP) methods. For each technique, matching scores are optionally normalized.

Weighted Sum Fusion: with this fusion technique, the weighted matching scores of all matchers are added up. For a binary classification problem, it is sufficient to focus on just one class (*genuine*) and determine, if the sum passes a certain threshold.

The fused score of the normalized i^{th} match score is defined as

$$f^i = \sum_{j=1}^{R} w_j ns_j^i \tag{11}$$

where w_j is the j^{th} matcher's weight [13]. For pure sum fusion, the weight of all matchers is set to 1.

It is necessary to find an optimal threshold t in order to perform final classification on the produced fusion score f^i (*genuine*, if $f^i \geq t$; *impostor* else). This optimization is performed during fusion training and utilizes a sigmoid

activated linear regression model that takes single fusion scores as input. Such model is described below.

EER-Weighted Sum Fusion: it sets the weights of accurate matchers higher than those of less accurate ones. The weight w_j of the j^{th} matcher is defined as

$$w_j = \frac{1}{EER_j \left(\sum_{i=k}^{R} \frac{1}{EER_k} \right)} \tag{12}$$

where EER_j represents the j^{th} matcher's equal error rate [13]. All weights are then applied on the weighted sum fusion defined in Equation 11.

Linear and Logistic Regression Fusion: both regression fusion techniques are a form of classifier-based score fusion where a boundary between the classes *genuine* and *impostor* is learned [56]. For linear regression, the following function is approximated

$$f^i = \beta + \sum_{j=1}^{R} w_j \cdot ns_j^i, \tag{13}$$

while for logistic regression,

$$f^i = \frac{e^{\beta + \sum_{j=1}^{R} w_j \cdot ns_j^i}}{1 + e^{\beta + \sum_{j=1}^{R} w_j \cdot ns_j^i}} \tag{14}$$

is approximated. The produced value f^i is then passed to a sigmoid function for classification. For optimization, the machine learning model adjusts both the bias β and each j^{th} matcher's weight w_j.

Multi-Layer Perceptron (MLP) Fusion: MLPs are fully connected neural networks with one or more hidden layers. As defined in [13], MLP fusion with one hidden layer applies the function

$$f^i = s_o \left(\sum_{j=0}^{M_H} w_{ji} s_h \left(\sum_{k=0}^{M_I} w_{ki} ns^i \right) \right) \tag{15}$$

on the i^{th} normalized score ns^i to produce the i^{th} fused score f^i. The sigmoid activation functions of the output (s_o) and the hidden layer (s_h) are provided with the sum of M_H weighted (w_{ji}) hidden and M_I weighted (w_{ki}) input nodes respectively, whose weights are optimized during the training process [13].