University of Media Stuttgart

Faculty of Information & Communication
Data Science & Business Analytics

Robert Bosch GmbH

Corporate Research
Department CR/AEU2

THESIS

Continuous Authentication using Inertial-Sensors of Smartphones and Deep Learning

Holger Büch

Mat.-No. 33953

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Data Science & Business Analytics
at the University of Media Stuttgart in June 28th, 2019.

1. Reviewer	Prof. Dr. Johannes Maucher Faculty of Print and Media University of Media, Stuttgart
2. Reviewer	Dr. Michael Dambier Corporate Research, CR/AEU2 Robert Bosch GmbH
Additional Advisor	Dr. Robert Duerichen Corporate Research, CR/AEU2 Robert Bosch GmbH

Holger Büch
Continuous Authentication using Inertial-Sensors of Smartphones and Deep Learning
THESIS, June 28th, 2019
Reviewers: Prof. Dr. Johannes Maucher and Dr. Michael Dambier
Additional Advisor: Dr. Robert Duerichen

University of Media Stuttgart
Data Science & Business Analytics
Faculty of Information & Communication
Nobelstraße 10
70569 Stuttgart, Germany

Declaration

Hiermit versichere ich, Holger Büch, ehrenwörtlich, dass ich die vorliegende Masterarbeit mit dem Titel: “Continuous Authentication using Inertial-Sensors of Smartphones and Deep Learning” selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§ 17 Abs. 5 der Studien- und Prüfungsordnung berufsbegleitender weiterführender Studiengänge, 5-semestrig) einer unrichtigen oder unvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.

Stuttgart, June 28th, 2019

Holger Büch

Abstract

The legitimacy of users is of great importance for the security of information systems. The authentication process is a trade-off between system security and user experience. E.g., forced password complexity or multi-factor authentication can increase protection, but the application becomes more cumbersome for the users. Therefore, it makes sense to investigate whether the identity of a user can be verified reliably enough, without his active participation, to replace or supplement existing login processes.

This master thesis examines if the inertial sensors of a smartphone can be leveraged to continuously determine whether the device is currently in possession of its legitimate owner or by another person. To this end, an approach proposed in related studies will be implemented and examined in detail. This approach is based on the use of a so-called Siamese artificial neural network to transform the measured values of the sensors into a new vector that can be classified more reliably.

It is demonstrated that the reported results of the proposed approach can be reproduced under certain conditions. However, if the same model is used under conditions that are closer to a real-world application, its reliability decreases significantly. Therefore, a variant of the proposed approach is derived whose results are superior to the original model under real conditions.

The thesis concludes with concrete recommendations for further development of the model and provides methodological suggestions for improving the quality of research in the topic of “Continuous Authentication”.

Keywords: Deep Learning, Machine Learning, Sensors, Authentification

Abstract (German)

Für die Sicherheit von Informationssystemen ist die Legitimierung der Nutzer von großer Bedeutung. Der Authentifizierungsprozess ist dabei eine Gratwanderung zwischen Sicherheit des Systems und Benutzerfreundlichkeit. So können etwa erzwungene Passwortkomplexität oder Multi-Faktor-Authentifizierung den Schutz erhöhen, für Anwender wird die Bedienung jedoch umständlicher. Daher stellt sich die Frage, ob die Identität des Nutzers auch ohne seine aktive Mitwirkung zuverlässig genug verifiziert werden kann, um dadurch Anmeldeprozesse sinnvoll ersetzen oder ergänzen zu können.

In dieser Masterarbeit wird die Frage untersucht, ob mithilfe der Inertialsensoren eines Smartphones kontinuierlich ermittelt werden kann, ob sich das Gerät gerade in Besitz seines rechtmäßigen Eigentümers befindet, oder von einem Dritten getragen wird. Hierzu wird ein in der Forschungsliteratur vorgeschlagener Ansatz nach implementiert und genauer untersucht. Der Ansatz basiert auf der Verwendung eines sogenannten siamesischen künstlichen neuronalen Netzwerks, um die Messwerte der Sensoren in einen anderen Vektor zu transformieren, der zuverlässiger klassifiziert werden kann.

Im Ergebnis wird gezeigt, dass sich die berichteten Ergebnisse des vorgeschlagenen Ansatzes unter bestimmten Voraussetzungen reproduzieren lassen. Wird das gleiche Modell unter Bedingungen eingesetzt, die einer realen Anwendung näher kommen, nimmt die Zuverlässigkeit jedoch massiv ab. Daher wird eine Variante des genutzten Ansatzes hergeleitet, deren Ergebnisse dem ursprünglichen Modell unter realen Bedingungen überlegen sind.

Die Arbeit schließt mit konkreten Empfehlungen zur Weiterentwicklung des Modells und gibt methodische Anregungen zur Qualitätssteigerung der Forschung in diesem Themenfeld der “Continuous Authentication”.

Schlagwörter: Deep Learning, Machine Learning, Sensoren, Authentifizierung

Contents

1 Introduction

Verifying a user's identity is crucial for the security of information systems. But the authentication process, in which a person verifies himself as a known and legitimate user of a system, seems to bear an inherent trade-off between reliability and convenience: The convenience of the user decreases if the reliability is improved, e.g., by enforcing complex password rules, multi-factor authentication, or frequent re-authentication. And this is not just a question of comfort, the security of the whole system can suffer from low user acceptance, whether through a written password attached to a computer monitor, a personal physical access token shared between colleagues, deliberately deactivating security features or other workarounds.

Is it possible to validate a user, without the need for his active participation in the authentication process? How can a system continuously collect information provided passively by the user and use unique patterns embedded in this information to identify the user? Does such a system improve the convenience of authentication? How reliable is it, and how does it affect security? Those are common research questions in the field of Continuous Authentication (CA), the topic of this master thesis. Given the omnipresence of personal smartphones, with their steadily increasing computational power and a variety of built-in sensors, it seems appropriate for this thesis to focus on leveraging the capabilities of those devices.

The purpose of this thesis is to assess the feasibility of a CA approach which is based on data provided by the inertial sensors of smartphones. Those sensors, gyroscope, accelerometer, and magnetometer, have the advantage to not depend on the active usage of the smartphone, they only need to be able to capture movements produced by the user. This aspect concludes the motivation behind the thesis: If it is possible to authenticate a user through the smartphone without the need for active usage of the device, it will open up the possibility to not only authenticate the user against, e.g., the phones operating system. The authentication state could also be shared with third-party systems, which cannot recognize the user on their own, to benefit from CA through the smartphone.

The research in this thesis focuses on experiments performed on a desktop computer with datasets collected by smartphones up front, and on verifying the ability of different machine learning models to classify the user as legitimate "owner" or

illegitimate “impostor”. Before this ability is evaluated, further topics seem to be secondary. Therefore, other important tasks like testing the authentication on actual mobile devices, assessing the system’s security against planned attacks, designing an architecture for integration with third-party systems, or keeping the machine learning model up to date have been held out of scope for this thesis.

The idea of using inertial sensors of smartphones for CA already was the topic of many previous studies by different researchers. Such related work is summarized in Chapter 3. While I examined those studies, it became apparent, that most researchers spend a lot of effort into feature engineering, trying to find different representations of the sensor data which can be leveraged effectively as input for machine learning models to classify the sensors’ signals. But with the advancements in the field of Artificial Neural Networks (ANNs), it is tempting to supersede the manual feature engineering with feature learning through deep learning methods.

The promising results published by Centeno et al. (2018) were one of the reasons I decided to reimplement their approach, a combination of a Siamese Convolutional Neural Network (CNN) with an One Class Support Vector Machine (OCSVM), as a significant aspect of this thesis. I elaborate on this and other design decisions, along with further building blocks of my concept, in Chapter 4.

The details of the experiments and the used dataset are presented in Chapter 5. Those experiments include the implementation of an OCSVM baseline model and the mentioned ensemble of a Siamese CNN with an OCSVM, along with an extensive evaluation. Within the experiments, I demonstrate, that the promising reported results are to my best knowledge caused by a setup, which is not applicable in a real-world scenario. I also show that the same models perform significantly weaker if the conditions of a real-world scenario are applied. Further, I evaluate the impact of various model parameters and propose a variation of the original model with improved performance in conditions closer to real-world scenarios, but without getting near the performance reported in the original study.

In Chapter 6, I discuss significant findings I discovered while working on this thesis, propose questions for future research, and make suggestions on how the research in CA could be improved in general.

But before deep diving into the topic of this thesis, the next Chapter 2 introduces into basic aspects specific to the examined subject.

2 Basics

The first section in this chapter explains and contextualizes the domain of this thesis: behavioral biometrics based Continuous Authentication. The second section describes the functional principles and constraints of the smartphone sensors relevant for this thesis to improve the understanding of the data they produce.

2.1 Access Control and Authentication

To protect a computing system against illegitimate access, it has to be equipped with some kind of control mechanism. To pass this mechanism, the user has to follow a specific process containing multiple steps or sub-processes. This topic is too complex to be cover in detail in this thesis. The following subsections focus on the aspects needed to understand and judge the approach of CA as the topic of this thesis. The three aspects most relevant for this thesis are: The general process of access control including authentication, the source of information which is used to perform authentication, and the frequency of authentication with its impact on security and convenience. For a more detailed examination of authentication, refer to Dasgupta et al. (2017).

2.1.1 Access Control Process

The process of gaining access to a computing system usually consists of multiple steps or subprocesses. These often include at least the following three subsequent steps: identification, authentication, and authorization.

Identification is the process where the user has to proof his “true” identity, e.g., his legal identity as a citizen by showing his official passport, or his identity as the owner of a specific e-mail account by entering a piece of secret information that has been sent to that account (Dasgupta et al., 2017, p. 2). Often identification is only needed during an initial registration process or if irregularities question the true identity.

Authentication is the process in which the user provides information to verify that he, as a user, is a person with an already proven “true” identity. Usually, this information

contains an identifier and a secret like a username and a password. Dasgupta et al. (2017, p. 2) define this process as follows:

> "Authentication is the mandatory process to verify the identity of a user and restricts illegitimate users to access the system."

Authorization is the process of granting or denying access right to the authenticated user, respectively his device (NIST, 2018). Usually, those access rights are only temporarily valid during a usage session and get revoked after a certain period, e.g., after a defined time of inactivity or triggered by a particular action. Then the user is prompted to authenticate again if he wants to regain access to the system.

2.1.2 Information Sources for Authentication

One aspect of authentication is the kind of information provided to the system. This information can be knowledge-based, possession-based and biometric-based, or like Li et al. (2018, p. 32555) wrote, based on "what you know", "what you have" and "what you are".

Knowledge-based information has to be memorized by the user. Common types are passwords, PINs, or graphical patterns. This information should be theoretically hard to extract by an attacker, but in practice, its vulnerability through inherent weaknesses has been demonstrated frequently, e.g., on the system side by automatically guessing weak passwords or on the user side by social engineering or observation. Nevertheless, because of its convenience and cheap implementation, the application of this approach is still widespread. (Centeno et al., 2018, p. 2)

Possession base information is bound to a device, like a card, token, or key. The information stored on this device is just as safe as the device itself, which can be stolen or replicated, sometimes even during normal usage and without the knowledge of its owner, like as it keeps happening with credit cards. Possession based authentication requires hardware support to read the information from the device. (Li et al., 2018, p. 32555)

Biometric-based information is supposed to be bound to the user as an individual. Physical biometrics rely on static characteristics of the user, that are considered to be close to unique for any human, like fingerprints or voice. This information is prone to duplication, often caused by the limited precision of the devices detecting the biometrics. Behavioral biometrics are based on the identification of unique patterns in the behavior of human beings during activity, e.g., touch or body movements. While this approach is also prone to duplication attacks, it has the benefit to not

necessarily require the user's active participation during the authentication process. (Li et al., 2018, pp. 32555f)

Nowadays, often two or more different sources of information are combined in a so-called two or multiple factor authentication process to improve security and compensate for the weaknesses of the individual approaches. In this case, the user has to prove, e.g., that he doesn't only know username and password but is also in possession of the phone number associated with his identity by answering an automated phone call.

2.1.3 Frequency of Authentication

The kind of information used for authentication also sets constraints to the frequency, in which the authentication can be performed: Very common is a one-time authentication, where the authentication information is requested only once per session. If the authentication information is gathered repeatedly during a single session in short time intervals, it is called Continuous Authentication (CA).

As already mentioned, knowledge-based, possession-based, and physical biometrics-based authentication approaches require the active participation of the user. He has to enter his credentials, swipe his card, connect a USB-Device, or touch a fingerprint reader. Those one-time authentications stay valid, e.g., until a certain period has been passed or until a certain number of transactions has been reached. The thresholds for such limits have to be balanced between the inconvenience for the user and the desired level of security. An attacker can exploit the delay between authentication and revocation to gain access, e.g., by stealing an unlocked smartphone or slipping through a closing door.

CA[1] on the other hand, continually monitors the legitimacy of the user, e.g., by continuously analyzing behavioral biometrics. The system can revoke access if it detects that another person replaced the legitimate user. Practically there still is a delay between authenticating and revoking. This delay has to be minimized. As authentication needs no active participation by the user, this delay is only restricted by the ability to identify the user through implicit information and doesn't necessarily have to be balanced against convenience. Besides the usage of CA as the primary authentication method, the absence of any user interaction facilitates its application as an auxiliary method to increase security as a second authentication factor or to improve convenience, e.g., by skipping reauthentication if the CA system provides a sufficient certainty about the user's identity. (Centeno et al., 2017, p. 2)

[1] Sometimes also called implicit, transparent or active authentication (Patel et al., 2016, p. 50).

2.1.4 Authentication Metrics (FAR, FRR, and EER)

Various metrics exist to compare the performances of authentication systems. The most common metrics for evaluation of CA systems are Falsse Acceptance Rate (FAR)[2], False Rejection Rate (FRR)[3] and Equal Error Rate (EER).[4], originating from the field of biometrics. (Ashbourn, 2015, p. 12)

FAR is the ratio of the total number of falsely accepted authentication attempts by unauthorized users to the total number of correctly rejected attempts by unauthorized users (Li et al., 2018, p. 32560). The lower the FAR, the more secure can a system be considered. FAR is similar to the so-called *type I error* or α*-error* used in statistical tests (Jain et al., 2004, p. 7):

$$FAR = \frac{\text{false acceptances}}{\text{correct rejects}} \tag{2.1}$$

FRR is kind of the opposite to FAR. It is the ratio of the total number of falsely rejected authentication attempts by legitimate users to the total number of correctly authorized attempts of legitimate users (Li et al., 2018, p. 32560). A lower FRR indicates better convenience for the user because the system is less likely to deny legitimate access. It can be compared to *type II error* or β*-error* in statistics (Jain et al., 2004, p. 7):

$$FRR = \frac{\text{false rejects}}{\text{correct acceptances}} \tag{2.2}$$

As decreasing FAR usually results in a higher FRR and vice versa, a trade-off has to be chosen between those metrics depending on the circumstances of the planned use case (Li et al., 2018, p. 32560). The EER is used to make this trade-off comparable to a certain extent: It is the value where FAR equals FRR (Figure 2.1). A lower EER indicates a more reliable system. Table 2.1 shows EERs for various biometric features as a reference.

The EER cannot be measured directly. Instead, it can be approximated by searching the classification threshold t that minimizes $|FAR - FRR|$ and then calculating $EER = \frac{FFR+FRR}{2}$ for this threshold t.

[2]Also called False Match Rate (FMR; Jain et al., 2004, p. 6).
[3]Also called False Non Match Rate (FNMR; Jain et al., 2004, p. 6).
[4]Used by, e.g., Neverova et al. (2016, p. 1817), Sitová et al. (2016, p. 880), Li et al. (2018, p. 32560).

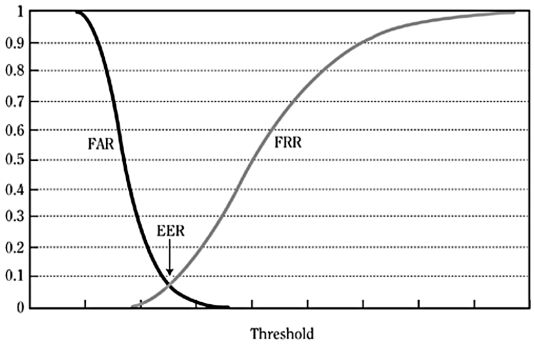


Fig. 2.1.: EER as trade-off between FAR and FRR. Source: Reid (2004, p. 150).

Tab. 2.1.: EERs reached with biometric features. Source: Dasgupta et al. (2017, pp. 48–60).

Biometric Modality	EER
Fingerprint	2%
Gait	7.3%
Keystroke	1.8%
Retina	0.8%
Voice	6%

2.2 Sensors

Before working with smartphone sensor data, it is useful to know how those sensors work, what exactly they measure, and which of their characteristics possibly influence the usage of their data for analysis purposes. While smartphones contain multiple types of sensors, which also vary between vendors and models, this section is limited to cover three types that are used in the later implementation (see Chapter 5).

2.2.1 Accelerometer

Accelerometers measure the acceleration a, which is described as the change in velocity $\mathrm{d}v$ over time t divided by time of the velocity change $\mathrm{d}t$. As the velocity v can also be described as the derivation of distance x over time t, a can also be described by using the change in distance. (Hering and Schönfelder, 2018, p. 371)

$$a(t) = \frac{\mathrm{d}v(t)}{\mathrm{d}t} = \frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2} \tag{2.3}$$

That's how the acceleration can be derived from measuring the time and deflection of a springy mounted mass during the exposure to an accelerating force (Figure 2.2).

Sensors in smartphones are built as Microelectromechanical Systems (MEMSs) and detect the position of a mass mounted in a substrate by measuring electric capacity (Figure 2.3) (Hering and Schönfelder, 2018, pp. 372f). A single accelerometer can measure the linear acceleration along a single axis. Therefore, three of such devices are needed to cover the three-dimensional space.

From the output of the accelerometer, the velocity v is calculated by integrating acceleration a while the position x can be derived from integrating velocity (W3C, 2019, Sec. 3.1):

$$v = \int a \cdot \mathrm{d}t \quad \text{and} \quad x = \int v \cdot \mathrm{d}t \tag{2.4}$$

Therefore, it is theoretically possible to obtain the orientation of the accelerometer in space. But the double integral amplifies the sensors drift (see Section 2.2.5) and renders the position x too imprecise for most applications. (W3C, 2019, Sec. 3.1)

2.2.2 Gyroscope

Gyroscopes are devices to measure angular displacement per time or so-called angular velocity. A single gyroscope can detect the angular rate against one axis. Therefore, three gyroscopes need to be combined to cover all three axes of space (Hering and Schönfelder, 2018, pp. 374f.). For sensors in modern smartphones, three so-called Coriolis Vibratory Gyroscopes (CVGs) are combined in a single MEMS (Kalantar-zadeh, 2013, pp. 143–145).

The gyroscopes that are embedded in consumer devices like smartphones are usually based on a vibrating mass (Figure 2.4). This mass vibrates along one axis and gets deflected by the Coriolis force when it is exposed to a torque. The sensed deflection is used to calculate the angular velocity of the movement. Because of the constant high-frequency vibration, gyroscopes are among the sensors with the highest power consumption. (W3C, 2019, Sec. 3.2)

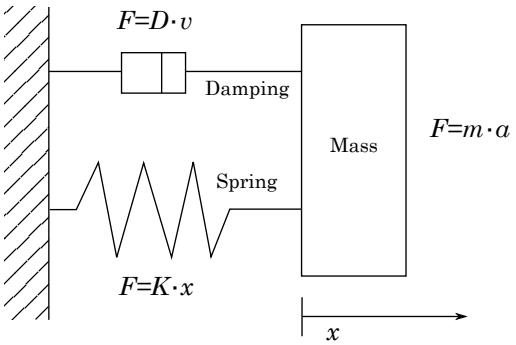


Fig. 2.2.: Schematic of an accelerometer. Source: Hering and Schönfelder (2018, p. 371).

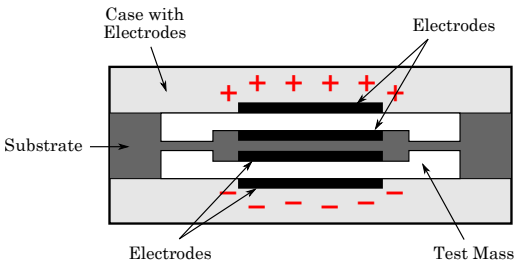


Fig. 2.3.: Schematic of a capacity based MEMS accelerometer. Source: Hering and Schönfelder (2018, p. 373).

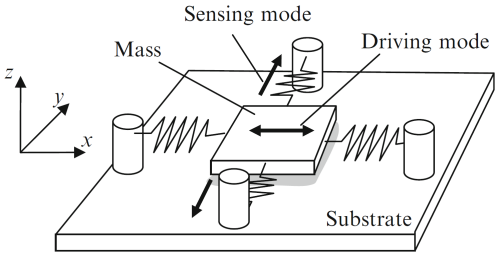


Fig. 2.4.: Schematic of a vibrating gyroscope. Source: Kalantar-zadeh (2013, p. 145).

The angular velocity w can be used to calculate the angular distance moved during the time span dt, which could then be used to determine the current angle θ of the gyroscope itself (W3C, 2019, Sec. 3.2):

$$\theta_n n = \theta_{n-1} + w \cdot dt \tag{2.5}$$

But like with accelerometer drift generated by noise has to be considered.

2.2.3 Magnetometer

While there are different types of magnetometers available, the most common types used in today's smartphones are based on Anisotropic Magnetoresistance (AMR) or Giant Magnetoresistance (GMR). Those types can be miniaturized, show a low power consumption as well as low production costs. But they still provide a magnetic resolution in the order of 1–10 nT which is good enough for navigation purposes on Earth[5]. (Včelák et al., 2015, pp. 1077f)

AMR and GMR are Magnetoresistance (MR) effects: if a conductor of a particular material gets exposed to a changing external magnetic field its electric resistance changes. This change in resistance can be measured and used to infer the strength of the external magnetic field. Both AMR and GMR are effects appearing in ferromag-

[5]The magnetic field of the Earth ranges between 25,000 and 65,000 nT depending mostly on the location of measurement. (Wikipedia contributors, 2018)

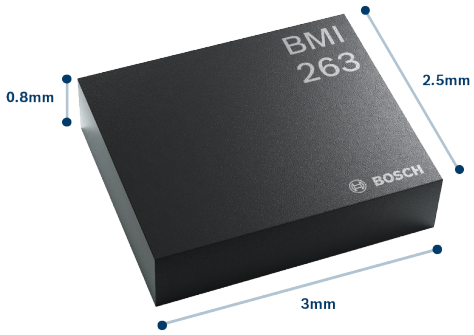


Fig. 2.5.: Bosch SensorTech BMI263 IMU. Source: Bosch Sensortec (2018a).

netic materials. AMR-based magnetometers are cheaper to produce than GMR-based magnetometers, which on the other hand are smaller, more accurate and have a lower power consumption. (Hering and Schönfelder, 2018, pp. 13–16)

Similar to accelerometer and gyroscope, a magnetometer in a smartphone needs to measure the strength of the magnetic field on all three axes to cope with different possible orientations of the phone (Včelák et al., 2015, pp. 1077f). To determine the direction in which the device is pointing the gravity vector, detected by the accelerometer and optionally gyroscope, is needed. (W3C, 2019, Sec. 3.3)

2.2.4 IMU

A three-axis accelerometer and a three-axis gyroscope can be combined in a single chip, sometimes also including a three-axis magnetometer. Such an Inertial Measurement Unit (IMU) can be produced as small as a gyroscope or accelerometer alone. An example of such a device is the BMI263 by Bosch SensorTec (Figure 2.5). This Inertial Measurement Unit (IMU) covers flexible measurement ranges which can cover from 125 °/s up to 2000 °/s of angular rate for the accelerometer and from 2 g up to 16 g for the gyroscope with a digital resolution of 16 bit. One benefit of IMUs like the BMI263 over the standalone sensors is the possibility to include self-calibration features. Some applications for such a device are image stabilization, panorama panning, or gesture recognition. (Bosch Sensortec, 2018b)

2.2.5 Constraints regarding Sensor Data

While the quality of the sensor output generally increases with the development of new smartphone generations, the data quality will keep competing with other factors like production costs, size, and energy consumption. As those compromises affect the data quality and analysis, it's useful to know their possible consequences, which influences their impact and how they can be compensated.

One unwanted effect is the so-called “noise”: Random fluctuations appearing in the output signal of the sensors, even if the measurand doesn’t change. The signal-to-noise ratio is a standard metric to describe its influence and is calculated by dividing the mean of the sensor’s signal by the standard deviation of the noise (Kalantar-zadeh, 2013, p. 14). A gradual change in the sensor’s output occurring even if the measurand stays constant is called “drift”. It is a systematic error and manifests itself in a change of the sensor’s baseline, i.e. the output value that the sensor generates without any stimulus (Kalantar-zadeh, 2013, p. 16). Inaccuracies during calibrating the sensors can lead to a so-called “calibration error”, which systematically shifts the signal’s output away from the input value of the stimulus (Fraden, 2016, p. 42).

Multiple internal and external factors influence noise, drift, and calibration errors. They can even vary between the same sensors of the same batch. Some external factors are temperature, electromagnetic signals, or mechanical vibrations (Kalantar-zadeh, 2013, p. 14). E.g., the magnetometer experiences an offset drift depending on the temperature of the sensor. Hard or soft magnetic material in the surrounding of the sensor can produce magnetic disturbances as well (Včelák et al., 2015, p. 1078). Even the intensity of the Earth’s magnetic field itself changes significantly depending on the distance to the poles. Temperature changes are also known to cause electronic noise (Kalantar-zadeh, 2013, p. 14). Errors from internal factors can be rooted in construction, production, assembling, or calibration. They can slowly appear over time, e.g., through the degeneration of the sensor’s material, which is known to influence the drift of sensors (Kalantar-zadeh, 2013, p. 14).

Some of the mentioned constraints can be reduced by fusing data from multiple sensors into a so-called high-level sensor. E.g., an absolute orientation sensor can be implemented by leveraging the gravitational vector detected by the accelerometer and the vector pointing to the North detected by the magnetometer. Additionally, data from the gyroscope can be used to enhance those vectors. Such a high-level sensor can provide orientation information stationary to the Earth’s plane like it is needed for Augmented Reality applications (Figure 2.6). Such high-level sensors often include preprocessing of the data, like removing noise with low-pass and high-pass filters and also are available as MEMS. (W3C, 2019, Sec. 4)

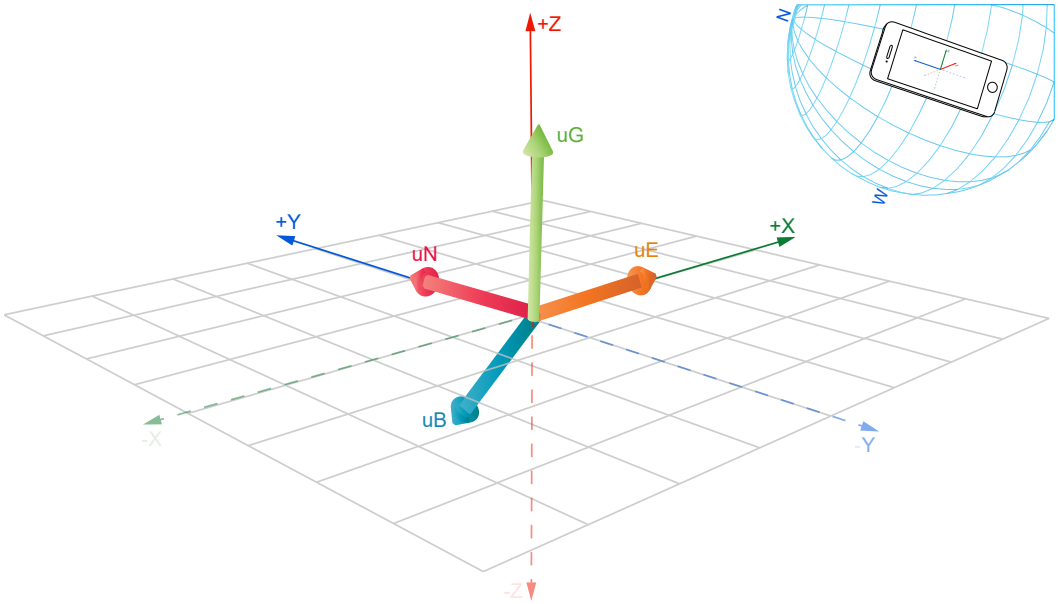


Fig. 2.6.: Smartphone's absolute orientation: Gravity vector uG from accelerometer and magnetic field vector uB from magnetometer are used to calculate the vectors uE pointing to East ($uE = uB \times uG$) and vector uN pointing to North ($uN = uG \times uE$). Source: W3C (2019, Sec. 4.3).

3 Related Work

This chapter presents related work to this thesis in a cross-sectional manner by comparing different methods chosen in previous studies for each step in the machine learning process. That deviates from common practice to present related work by sequentially describing studies one after the other. The intention behind this structure is to provide a better overview of the possible options for the individual steps as a foundation for the choices to make during the implementation. The downside of this structure is that the decisions made by the different authors are taken slightly out of context. Please refer to the original papers to get the full picture of the individual approaches.

Table 3.1 shows an overview of studies closely related to this thesis, which means, they are investigating CA using only smartphones' sensor data, leveraging at least one inertial sensor and relying on regular[6] smartphone usage. Related studies that do not quite match these criteria but are regularly referenced in the field of CA are listed in Appendix A.1. It is important to note that the experimental setups of the different studies differ regarding various important aspects, like the used datasets, the usage of different sensors (between 1 and 30 different sensor modalities) and differences in the evaluated attack scenarios. Due to those differences, the performance metrics reported in the table are not directly comparable!

3.1 Datasets

Even though there are multiple public datasets available containing smartphone sensor data for the purpose of activity recognition[7] those are usually not useful for CA, because the datasets were recorded in very controlled environments, with fixed mounted smartphones or published without subject information. One exception is the "Complex Human Activities Dataset (CHAD)" (Shoaib et al., 2016) as a collection focused on activity recognition, which also has been used for CA by E.-u.-Haq et al. (2017). It is limited to 10 subjects performing predefined activities for about 3–5 min and are thus not comparable to a real-world scenario.

[6] The user doesn't have to adjust his behavior or perform additional actions for the authentication.
[7] E.g. "Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set" (Reyes-Ortiz et al., 2015) or "WISDM Activity Prediction" (WISDM Lab, 2012).

Tab. 3.1.: Key-Studies in the field of CA using mobile phone sensors, ordered by year of publication. Column "One Class" (OC) indicates if only owner data is used during the final classification. The metrics for performance and detection delay are listed as reported by the authors.

Study	Features	Best Model	OC	Dataset	Performance (%)	Delay
Frank et al. (2010)	Constructed	Time-delay embeddings	no	Custom	Acc. 100%	–
Shi et al. (2011)	Constructed	Ensemble of various	no	Custom	Acc. > 95	–
Zhu et al. (2013)	Constructed	Continous n-gram	yes	Custom	TPR 71.3, FPR 13.1	4.96 s
Kayacik et al. (2014)	Constructed	Temporal/spatial model	yes	LLT, Custom	Acc. 53.2–99.5	122 s
Lee and Lee (2015)	Resampled	SVM	no	Custom	Acc. 90.23	~20 s
Roy et al. (2015)	Raw	HMM	yes	Custom	EER 0–12.85	21–1[a]
Sitová et al. (2016)	Constructed	SMD	yes	H-MOG	EER 7.16–10.05	~80 s
Neverova (2016)	Deep	DCWRNN + GMM	yes	Custom	EER 8.8–18.17	30 s
				H-MOG	–	–
Centeno et al. (2017)	Raw	Autoencoder	yes	H-MOG	EER 4.5	20 s
				CrowdSignals	EER 2.2	20 s
Lee and Lee (2017)	Resampled	RF + KRR	no	Custom	Acc. 98.1	6 s
E.-u.-Haq et al. (2017)	Activity Seq.	BN	no	CHAD	Acc. 94.57	5 s
E.-u.-Haq et al. (2018)	Constructed	DT	no	CHAD	Acc. 97.7	5 s
Centeno et al. (2018)	Deep	Siamese CNN + OCSVM	yes	H-MOG	Acc. 97.8	1 s
Shen et al. (2018)	Constructed	HMM	yes	Custom	FAR 3.98, FRR 5.03	8 s
				H-MOG	FAR 5.13, FRR 6.74	–
Li et al. (2018)	Constructed	OCSVM	yes	H-MOG	EER 4.66	5 s
Li et al. (2019)	Constructed	KRR	yes	H-MOG	EER 3.0	–
Deb et al. (2019)	Deep	Siamese LSTM	–	Custom	TAR 97.15 when FAR 0.1	3 s

[a] number of touch actions

The dataset "LiveLab Traces (LLT)" (Shepard et al., 2011) consists of smartphone data from 35 users and includes multiple sensor modalities besides the inertial sensors, like app usage and call information. On the downside, the duration of the data captured per user strongly varies from a few days to close to a year.

Another ambitious project called "CrowdSignals" tried to collect a vast amount of smartphone sensor data for multipurpose analytical use cases. But after their crowdfunding succeeded, they only published a small dataset sample until today (AlgoSnap Inc., 2015). However, at least Centeno et al. (2017, p. 5) achieved to get early access to the dataset for their research.

To my best knowledge, until now, there is only one publicly available dataset specifically for CA, which was collected by Yang et al. (2014b). It's commonly referred to as the "Hand Movement, Orientation and Grasp (H-MOG)" dataset and consists of data by inertial and touch sensors and was produced by 100 subjects during 24 sessions. In those sessions, the users had to perform predefined activities

mirroring realistic scenarios. While the dataset is of good quality, two aspects are limiting its usefulness for the use case of this thesis: All sessions include active usage of the smartphone, and they are of a pretty short duration of 5–15 min. The dataset became quasi-standard because of its public availability, proper documentation and ease of use. It was leveraged by various researchers, as a primary dataset by Sitová et al. (2016), Centeno et al. (2017), Centeno et al. (2018), Li et al. (2018), and Li et al. (2018) or for benchmarking of different approaches by Shen et al. (2018).

Other researchers like Roy et al. (2015), Neverova (2016), Shen et al. (2018), and Deb et al. (2019) undertook the effort to collect custom datasets but did not publish them. Outstanding in the aspect of scale is the collection of Neverova (2016, pp. 179f) which was performed by Google as part of a project called "Abacus"[8]: It is reported to include data from approximately 1500 users captured in real life situation. The dataset with probably the most features was collected by Deb et al. (2019, pp. 3f) and is supposed to include 30 different sensor modalities captured in a real-world setting.

3.2 Data Preprocessing

The input data for machine learning algorithms must be provided in an adequate amount, structure, and format for the particular task. Data preprocessing comprises of a set of techniques to transform raw data into the desired form. The decisions to be made during this process strongly influence the performance and characteristics of the final model. (García et al., 2015, p. 10)

The steps performed during data preprocessing depend on the characteristics of the raw data, the targeted machine learning model, and its use case specific application. It is not surprising that the studies related to CA using smartphone data applied quite different techniques during this process. The following section summarizes relevant approaches. If not explicitly stated otherwise, the studies do not provide clear information about the impact of those specific steps. This section is meant as an overview of options and doesn't conclude any recommendation.

3.2.1 Noise Filtering

During the process of collection, it is unavoidable that the data gets contaminated by imperfections to a certain degree. This kind of noise affects the performance and robustness of the machine learning models trained with this data. Noise filtering

[8]The project "Abacus" was guided by Google Inc. to eliminate the necessity of protecting smartphone applications with passwords. (Neverova, 2016, pp. 6f)

denotes a set of techniques used to improve data quality before feeding it to the model for training. (García et al., 2015, pp. 107f)

To reduce noise in the sensor's signal (see Section 2.2.5), E.-u.-Haq et al. (2018, p. 27) applied a smoothing filter with a length of 3 samples at 50 Hz frequency, while Deb et al. (2019, p. 5) leveraged Fast Fourier Transform (FFT) to handle and remove noise by mapping the measurements of the movement sensors from time domain to frequency domain.

Shen et al. (2018, p. 51) propose a process called "kinematic information extraction" composed of two steps for filtering data of inertial sensors and touch events. First, the gravity component of the accelerometer's signal is removed by employing a Kalman filter under the assumption that the device's acceleration without distortion by gravity reflects the motion of the smartphone more accurately. In a second step, noise reduction is performed by decomposing the signals using wavelet functions, applying threshold analysis to the components, and afterward reconstructing the original signal by using the inverse wavelet functions. This approach acknowledges that signal noise appears at the whole spectrum of frequencies of the signal sensor.

Reyes-Ortiz et al. (2015) also separated the gravitational and movement components from the acceleration signal but decided to use a Butterworth low-pass filter with 0.3 Hz cutoff frequency to target the assumed low frequencies of the gravitation.

3.2.2 Manual Feature Construction

Feature construction is the application of a set of operations to a set of existing features to generate new features. Whereby those new features do not yield any new information, but might describe the existing information in a form that is more suitable for specific models or applications (García et al., 2015, p. 189). This step was applied in the majority of the examined studies "manually": the transforming operations where chosen cognitively by the researchers.

It is common to compute features by applying a sliding window and calculating various metrics for analysis of time series like sensor data. The set of aggregation functions used for CA have already been widely tested in related fields like activity recognition (E.-u.-Haq et al., 2018, p. 28). From the commonly computed features (see Appendix A.2), the following have been reported to be quite useful for the task of CA: energy[9], entropy, minimum, maximum and mean of gyroscope as well as energy and entropy of accelerometer. (Shen et al., 2018, pp. 52f)

[9] "Energy" is also often denoted as "signal magnitude area" or just "magnitude".

The window parameters used for the aggregation range from 0.5 s to 5 s for window size, the step-size from 50% overlap to no overlap (E.-u.-Haq et al., 2018, p. 28; Reyes-Ortiz et al., 2015).

Beside fixed window parameters also cycle-based approaches have been evaluated. Here the features are aggregated from cycle to cycle, e.g., from step to step during gait. This cycle based approach performed weaker than the segment based procedure with fixed window parameters. The main difficulty seems to be inaccurate cycle detection in real-world environments, e.g., if the smartphone is not fixed to the body but is held loosely in hand or pocket. (Al-Naffakh et al., 2018, p. 17)

Li et al. (2018) and Li et al. (2019) transferred "data augmentation" strategies known from image classification to construct additional features by applying permutation, sampling, scaling, cropping and jittering to the samples of a particular window length. They report promising results on the H-MOG dataset (Li et al., 2018, pp. 5f). To me, the study does not clarify if this approach basically optimizes the classification for the given dataset or if it generalizes well enough to be used in a real-world scenario.

Even more advanced strategies were applied by Sitová et al. (2016, pp. 878–880), who crafted the H-MOG features, a set of metrics designed to capture grasp resistance and grasp stability during a touch event, and combined them with features representing tap and keystroke actions.

3.2.3 Data Reduction

It is common to reduce the number of features and instances used as model input to lower the computational effort, while maintaining a large amount of distinctive information of the original data. This process is also known as "dimension reduction" and regularly performed by grouping multiple features or selecting a meaningful subset of features. Equivalent methods performed on the instances are categorized as techniques for "data sampling". (García et al., 2015, pp. 147f)

Shen et al. (2018) reduced their feature space from 192 features to 38 by calculating the Mutual Information (MI) score and the Fisher score regarding the class variable and applying a threshold to discard features with MI score below 0.5. They identified the energy and the entropy of the gyroscope's signal as the most informative features. Kurtosis, skewness, and cross-mean rate resulted in MI scores below 0.5 for all sensors. Besides, Shen et al. (2018, pp. 51f) drastically reduced the number of samples by analyzing only the sensor data generated during touch events.

Al-Naffakh et al. (2018, pp. 20–22) proposed a "dynamic feature vector" and individually selected a particular subset of features for every owner. As selection criteria the means of every feature of the owner have been compared to the means of every individual impostor. The more often the feature's mean of the owner differs from the feature's mean of an impostor by at least standard deviation, the more the feature is considered to be helpful. After this scoring, the most useful features were selected, with the best results between 30 and 80 features. From my point of view, this approach seems to be kind of simplified Gaussian Mixture Model (GMM).

Neverova (2016, p. 182) used the well-known Principal Component Analysis (PCA) to transform the features into lower-dimensional space while preserving as much variation in the data as possible. Sitová et al. (2016, pp. 881f) also applied PCA but with the primary purpose of removing possible correlation between the features as a precondition for their classifiers.

3.2.4 Deep Features

A different approach that aims to reduce the number of manual steps and decisions needed for data preprocessing is to leverage ANNs to transform raw data into a meaningful representation with lower dimensionality. The ANNs are supposed to learn functions equivalent to techniques like filtering, manual feature construction, and dimensionality reduction during their training. The main downsides of this approach are the difficulty in identifying the best network architecture, the low interpretability of the learned transformation functions and often the need of a large amount of data plus compute power for training.

Neverova (2016, pp. 175ff) compared several network architectures regarding their ability to learn deep features in the context of Continuous Authentication (CA): CNN, vanilla Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Clockwork Recurrent Neural Network (CWRNN). She also proposed a new architecture named Dense Clockwork Recurrent Neural Network (DCWRNN) (Figure 3.1).

The DCWRNN is a modification of the CWRNN, which captures patterns within different temporal scales ("slower" and "faster" patterns) by partitioning the units of the hidden recurrent layers in different groups with different update frequency, the so-called frequency bands. In the original CWRNN, different subsets of units in the low-frequency bands are active over time. That slows down the learning process and results in possibly different responses to the same input at different times during testing. The DCWRNN compensates these shortcomings by introducing parallel threads shifted with respect to each other. (Neverova, 2016, pp. 176–179)

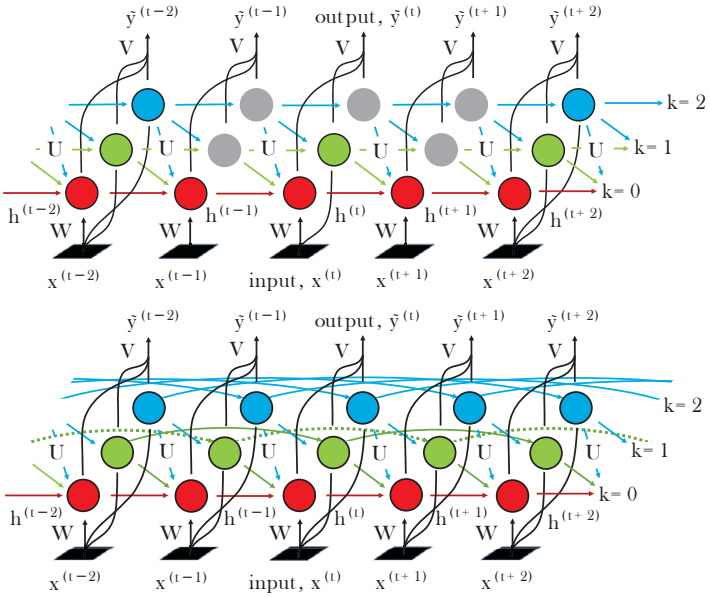


Fig. 3.1.: Comparison of the original Clockwork RNN (top) with its sometimes inactive units (grey) in the lower frequency bands (green, blue) and its modification called Dense Clockwork RNN (bottom). Source: Neverova (2016, p. 177).

Centeno et al. (2018, pp. 3f) used a different network architecture. They trained a Siamese CNN to learn features representing the difference in raw sensor data between the owner and impostors. During authentication, this model computes deep features, which are then fed into an OCSVM. This approach was reimplemented as part of this thesis and is described in detail in Section 4.5.2.

During the writing of this thesis, Deb et al. (2019, p. 5) published another variant: They claim to improve the quality of deep features generated by the Siamese network by using LSTMs instead CNNs for the two sub-networks (Figure 3.2).

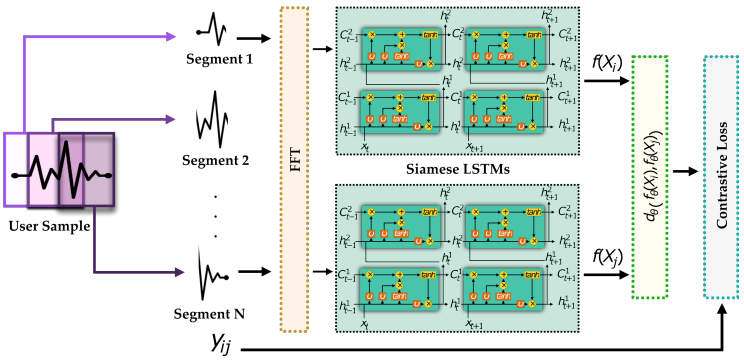


Fig. 3.2.: Architecture of Siamese LSTM. Source: Deb et al. (2019, p. 5).

3.2.5 Context Information

All the studies that evaluated the authentication performance of their models regarding different user contexts (e.g. “sitting”, “standing”, “walking”) revealed significant varying performance of the same model in different settings. E.g., models tend to perform better when the user is walking compared to when he is sitting.[10]

Therefore, it is possible to take circumstances into account by detecting the user’s context and using different authentication models for those different situations. Lee and Lee (2017, p. 304) tried classifying the samples into four different contexts first but ended up using two different context models in the end. E.-u.-Haq et al. (2017, p. 206) differentiated between six activities but provide classification metrics only for the final authentication model and not for the context classification itself.

3.3 Classifiers

The main task during CA is to classify the incoming sensor data of the device as produced by the owner (positive class) or as produced by an impostor (negative class). In related studies, this problem was either interpreted as a binary classification problem or as a problem of novelty detection, whereas only data of the positive class is used during classifiers’ training. Consequently, classifiers based on those two types of machine learning algorithms have been evaluated. This section provides an overview of models which are considered to be most relevant.

3.3.1 Gaussian Mixture Model

Gaussian Mixture Model (GMM) is a generative probabilistic algorithm to model input data as a mixture of multi-dimensional Gaussian probability distributions and can be used similar to k-Means as clustering algorithm (VanderPlas, 2016, pp. 476f). As a consequence, it can also be used to represent the feature density of each class and predict probabilities of membership to these classes for an input vector. This enables the use of GMM for classification tasks (Hastie et al., 2009, p. 463).

One aspect that makes GMM interesting for CA is the possibility to train a so-called Universal Background Model (UBM) first using an iterative Expectation-Maximization algorithm on all data and to later adapt this more generic model to new training data by retraining it using Maximum A Posteriori estimation (Reynolds,

[10] Shen et al. (2018, p. 55) reported the only exception and presented better results during sitting than walking, but the Hidden Markov Model (HMM) used in this study also is quite different from the other approaches.

2009). Hence, it's possible to do a compute-expensive pretraining of the GMM centrally with data from all subjects included in the training dataset, then distribute the UBM to the individual smartphones where it can be updated with the owner's data, which doesn't even have to be included in the central training dataset.

Neverova (2016, p. 173) leverage this approach by pretraining the GMM on a large amount of data with deep-features generated by their DCWRNN (see Section 3.2.4), then retraining this generic UBM individually for every user and combining both UBM and individualized user model as an ensemble classifier.

3.3.2 One Class Support Vector Machine

OCSVMs are a variant of the well-known Support Vector Machines (SVMs), a class of linear algorithms leveraged for solving e.g., classification or regression problems (Zhang, 2017). OCSVMs are trained with data limited to a single class to perform Novelty Detection on the testing data. As OCSVM is one of the models used for this thesis, it is described in more detail as part of the concept chapter in Section 4.5.

Sitová et al. (2016, p. 880) tested an OCSVM with Radial Basis Function (RBF) Kernel against two other single class models, Scaled Euclidean Distance (SED) and Scaled Manhattan Distance (SMD). They report OCSVM to result in the highest and therefore worst EER, but with the remark that the performance of the three models was close to par for walking scenarios and only strongly deviating in sitting situations (Sitová et al., 2016, p. 882). Shen et al. (2018, p. 54) used an OCSVM with an RBF Kernel as one of their baseline models and reported worse results compared to their main Hidden Markov Model (HMM). Centeno et al. (2018, p. 5) leveraged the OCSVM with an RBF Kernel as their only authentication model.

Lee and Lee (2015, pp. 5f) used a standard SVM as their only authentication model first, but later tested it against Kernel Ridge Regression (KRR), a Naive Bayes classifier, and a Linear Regression. The SVM was almost as good as the best performing KRR model regarding the classification results but was rejected in favor of KRR because of SVM's higher computational needs (Lee and Lee, 2017, pp. 302f). In the study by E.-u.-Haq et al. (2018, p. 31), the SVM performed better than the competing models k-Nearest Neighbors (k-NN) and Decision Tree (DT).

3.3.3 Hidden Markov Model

Roy et al. (2015) was among the first employing Hidden Markov Models (HMMs) for CA using smartphone sensor data. The underlying idea is that subsequent user

interactions can be interpreted as states in a Markov Process, where a stochastic function can represent the transitions between different states. Such HMMs can be used as a one class classifier and trained with only data from the owner. Roy et al. (2015) analyzed user interactions with their smartphones and identified the two most common touch gesture types: slide and tap gestures. For those they build two separate classifiers, each consisting of three HMMs, one for each data source: “touch”, “vibration” and “rotation”. The likelihood scores of the classifiers are averaged to get a combined score. That ensures robust results, even if data from a single sensor is unavailable for a certain time. When classifying multiple subsequent gestures, their mean in a specific window is used as the authentication score to provide more robust results. (Roy et al., 2015, pp. 1311–1313)

Shen et al. (2018) implemented a similar HMM based classifier. By using a larger pool of subjects from their own data collection, they claim to have been able to extract more informative behavioral features and improved the results of the classifier to an EER between 4.74% and 9.73% depending on the usage scenario.

Quite refreshing in the study by Shen et al. (2018) is their extensive evaluation. E.g., they discovered, that EER seems to decrease when the amount of training data gets too big: They tested short-, medium-, and long-term scenarios with mean training events of 154, 201 and 624, and reported EERs of 6.23%, 4.03% and 8.11%. Shen et al. (2018) suspect the longer time span between training and testing events as a reason for this observation. During the evaluation, the HMM based classifier revealed better results than a competing SVM and an ANN model. The authors argued the HMM captures the temporal nature of the sequence of events better than the other models. (Shen et al., 2018, pp. 56f)

3.3.4 Artificial Neural Networks

Besides being leveraged to generate deep features (see Section 3.2.4), ANNs were also studied as classifiers during authentication. Shen et al. (2018, pp. 54f) tested a three-layered Multilayer Perceptron (MLP) as comparative model against their main HMM classifier as well as against an OCSVM. The architecture consists of n input nodes for n features, $n \cdot 2$ nodes in a single hidden layer and a single node in the output layer. This node’s output is then used as the classification score. The neural network performed worst for all tested scenarios. However, the publication misses information on how exactly the MLP was trained in a one class approach.

Centeno et al. (2017) studied the use of an autoencoder as authentication classifier. They presented very promising results, but there are some open questions regarding the implementation, which I want to describe in the following in more detail.

In general, an autoencoder is a type of Artificial Neural Network (ANN) that is unsupervised trained to first encode a given vector from the input layer into a representation, the so-called code, which is described by a hidden layer of the network, and afterward, reconstruct the input from the code. The reconstruction is served in the output layer. It would not be especially useful if the autoencoder would be able to reconstruct the input vector exactly. Therefore, some restrictions are set upon the network which forces the autoencoder to approximate and only learn those aspects of the input data, which are most useful to reconstruct input vectors as good as possible. For some applications of autoencoders, the ability to reconstruct is only a means to an end for learning to generate the code, which can be leveraged for feature learning or dimensionality reduction. (Goodfellow et al., 2016, p. 499)

For an overview of the different autoencoder variants, I recommend Goodfellow et al. (2016, Chapter 14), while Hinton and Salakhutdinov (2006) describe the under-complete autoencoder in more detail.

The variant of autoencoder, that Centeno et al. (2017, p. 4) leveraged for smartphone-based CA, is the under-complete autoencoder (Figure 3.3). In this architecture, the hidden layer representing the code has smaller dimensions than the input data and thereby forces the autoencoder to learn a useful representation. During the learning process, a loss function, which describes the dissimilarity between the input vector and its reconstruction, is minimized. (Goodfellow et al., 2016, pp. 500f)

Centeno et al. (2017, p. 4) trained the autoencoder with samples from the owner in a single class approach. The idea is that the network learns to reconstruct the data from the user, which it was trained on, better than the data from unknown users. The distance between the input vector and its reconstruction is expected to be lower for samples from the owner, than for a sample from an impostor. The samples are classified binary by applying a threshold on this distance.

One aspect in the given implementation surprised me: Centeno et al. (2017, p. 6) used 1500 units in each hidden layer (they tested three setups using one, three, and five hidden layers). As they took 500 samples as input vector, and each sample consists of three values for each axis of the accelerometer, it means that the autoencoder actually was not set up as an under-complete autoencoder like depicted in their architecture chart (Figure 3.3). Also, no other kind of regularization like the sparse penalty is mentioned but would be expected for such an architecture. Via correspondence[11] I learned that different numbers of hidden units have been tested (e.g., $1500 \times 750 \times 350 \times 750 \times 1500$) but ended in similar results which was also surprising to hear.

[11] E-Mail by M. P. Centeno, 26th Feb. 2019.

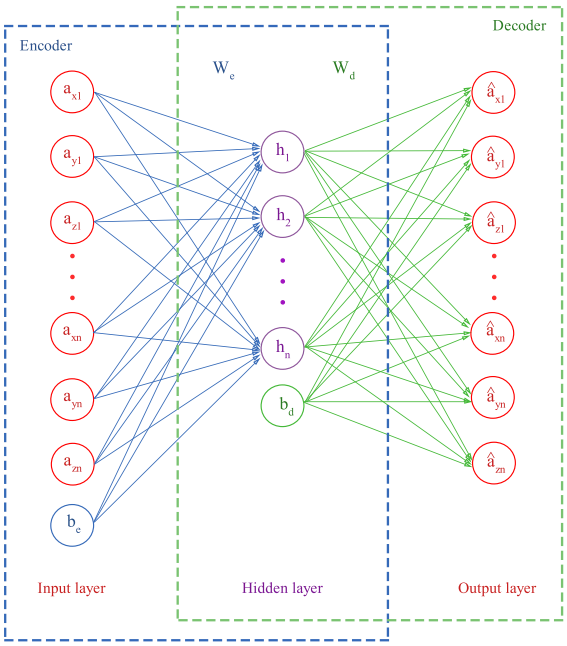


Fig. 3.3.: Autoencoder architecture with one hidden layer. a_{x1}, a_{y1}, a_{z1} ... a_{xn}, a_{yn}, a_{zn} are a set of accelerometer measurements taken as input, h_1 ... h_n are activation functions generating the code. W and b denote weights, and bias vectors learned by the encoder and decoder sections. Source: Centeno et al. (2017, p. 4).

3.4 Evaluation Settings

An aspect that came to my attention during the review of related studies is the absence of any kind of standardized approach for evaluating the performance of the models in the context of CA. The different evaluation settings used by the authors prevent any meaningful comparison between the studies. It is concerning that this situation does not stop most authors from comparing the metrics directly. This section describes aspects of the evaluation which are considered to be most relevant and tend to be handled differently in the various studies.

One of the issues which make direct comparison impossible is that the studies use different metrics to report the performance of their approaches (s. Table 3.1). The Equal Error Rate (EER), one of the standard metrics for the biometric authentication system (see Section 2.1.4), is reported as a metric by roughly one-third of the studies. Nearly half of the studies report accuracy (*number of correctly classified objects* divided by *total number of objects*) as the classification metric. Even if this metric is applied to balanced datasets[12], it doesn't reveal if the model has weaknesses regarding false positives or false negatives, which has important implications for authentication use cases. Most studies provide at least Receiver Operating Character-

[12]For datasets with unbalanced classes, accuracy is significantly harder to interpret and compare. If in such cases the class distribution is not reported along with the accuracy, the metric becomes meaningless.

istic (ROC) plots to give these insights. A minority of studies reported only FAR and FRR as their primary metrics. As those metrics are parts of a trade-off, it needs a justification for reporting specific values. None of the affected studies explicitly states such an explanation, but at least they all provide plots comparing FAR and FRR. Those plots suggest that the reported values maybe were selected using elbow-points. Deb et al. (2019, pp. 7f) reported the True Acceptance Rates (TARs) corresponding with FAR 1% and 0.1% without stating reasons for those thresholds.

Another problem is *what* is reported as the final authentication metric. E.g., Deb et al. (2019) present metrics for the ability of their Siamese LSTM to distinguish positive pairs and negative pairs as "authentication performance", but without stating how the classification was done.[13] It is also unclear to me how this approach should be implemented in a real authentication situation, especially how the positive and negative pairs should be generated in such a case, and if the results would be comparable. E.g. Deb et al. (2019, p. 6) compared their metrics with the metrics reported by Centeno et al. (2018), but they both were derived quite differently: Deb et al. (2019) reported the outcome of classifying positive and negative pairs with their Siamese LSTM, while Centeno et al. (2018) used the Siamese CNN to generate deep features as an input for a subsequent OCSVM and reported the final authentication metrics. In my opinion, Deb et al. (2019) compares metrics of a binary classification approach with metrics of a more difficult one class approach.

Some studies distinguish the performance regarding different usage scenarios like walking or sitting, and report the metrics separately (Sitová et al., 2016, p. 877; Shen et al., 2018, pp. 55f), while others report overall results (Centeno et al., 2017, pp. 7f; Li et al., 2018, p. 11).

As a third problem, the evaluation setup varies a lot between the studies. Centeno et al. (2017, p. 7) report the average of cross-validating 10 random "attack scenarios", where such a scenario is defined as classifying testing samples from the owner, whose data was used to train the model, against the same number of samples from another randomly selected user (1 vs. 1). On the other hand, Li et al. (2018, p. 6) classified a certain amount of the owner's samples along with the same amount of samples randomly drawn from all 98 other users' data (1 vs. all). Additionally, while most studies reported the metrics for classifying a single sample, others like Shen et al. (2018, p. 54) and Roy et al. (2015, p. 1313) report a "robust" value, which improves accuracy by calculating the mean of the individual sample's score across a sliding window. Those different approaches are valid and justified, but make the results harder to compare.

[13] One possibility would be a threshold on the Euclidean distance between the deep features. Another would be an additional single node layer in the network using the deep features as input.

Finally, different datasets contribute to the difficulty of comparison. At least Shen et al. (2018) and Centeno et al. (2017) report results using the H-MOG dataset as a reference along with a second dataset. But due to limitations of the dataset, this is not always possible, e.g. Neverova et al. (2016, p. 1815) stated, H-MOG is too small for their approach, other studies like Deb et al. (2019, p. 2) leveraged sensor data that is not included in H-MOG. To my best knowledge, Sitová et al. (2016) (the authors of H-MOG) were the only researchers to publish their datasets. Therefore, all other studies with custom datasets are not reproducible.

As described above, the direct comparison of the various studies' results is difficult. Therefore, it is important to value the contributions of Khan et al. (2014) and Shen et al. (2018) who are so far the only ones to provide comparative studies for the different approaches. Khan et al. (2014) reimplemented six approaches published between 2010 and 2014, including three approaches leveraging inertial sensors. They evaluated accuracy, training delay, as well as computational complexity. Shen et al. (2018), as part of assessing their own approach, also reimplemented six different approaches published between 2013 and 2016. They used their custom dataset as well as H-MOG for the evaluation. The result of this comparative study provides a different view on the performance of the various models than the individual results reported by the different authors do. Unfortunately, Shen et al. (2018) only provide FAR and FRR as classification metrics. They also stated that their results should not be generalized, and further investigation would be needed for a realistic comparison. (Shen et al., 2018, pp. 58f)

4 Concept

This chapter describes the ideas behind the implementation of Continuous Authentication (CA) using smartphones' inertial sensors presented in this thesis. It condensates the options and knowledge gained from related studies (see Chapter 3) into a coherent concept fitting to the circumstances of this thesis.

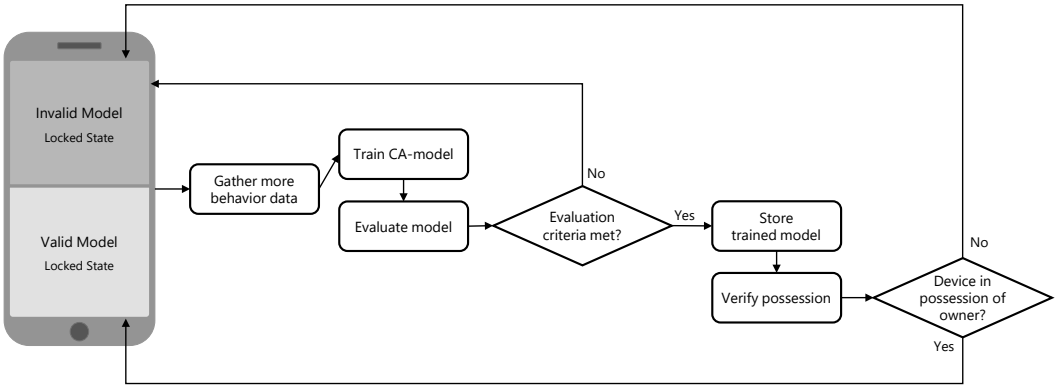
First, I describe the general idea of how to apply smartphone based CA in practical use cases. Then I justify the design decisions I made before implementation and explain the algorithms which have been identified as promising. Evaluation criteria for testing the performance of the applied models are presented as well as the evaluation setup in which those metrics should be computed to gain valid results.

4.1 General Idea

The general idea behind smartphone based CA is to estimate the probability of the current smartphone's user being the smartphone's owner. That idea is built upon the premise that a smartphone is typically tied to a single person, the owner, and not regularly shared among multiple persons.

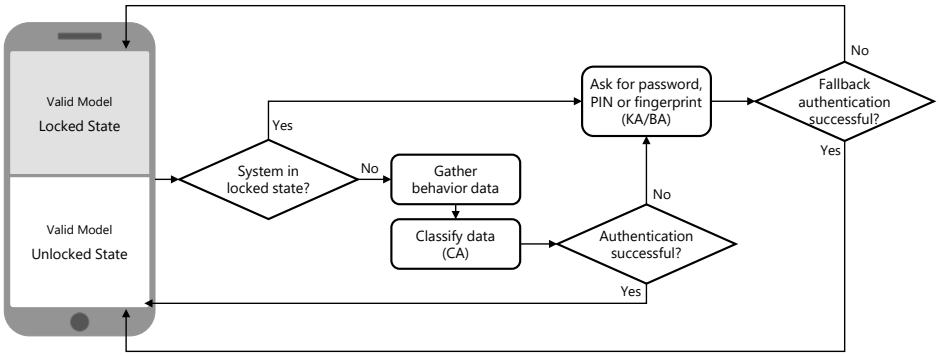
In a practical use case, the authentication process would be divided into three phases: First, an enrollment phase, followed by continuous authentication and updating phases. The last phase, which is needed to keep the model up to date over a longer time-span, is not in the scope of this thesis but could look similar to the enrollment phase. I present the first two phases in the following as a generic framework, not a concrete architecture. It is designed upon an authentication framework presented by Crouse et al. (2015, p. 137).

In the enrollment phase, the smartphone gathers data, which is used to train a model for authentication (Figure 4.1). This model is then evaluated, e.g., by testing it with the owner's data and a predefined set of impostor data. If the model is considered as not good enough, more data will be gathered, and the model will be retrained and reevaluated. If the model meets the evaluation criteria, it is stored for usage in the authentication phase. But before switching to the authentication phase, it is necessary to check additional criteria like if the smartphone is still in possession of



CA: Continuous Authentication

Fig. 4.1.: Schematic of the enrollment phase of the proposed authentication framework.



CA: Continuous Authentication, **KA**: Knowledge-based Authentication, **BA**: Biometrical Authentication

Fig. 4.2.: Schematic of the authentication phase of the proposed authentication framework.

its legitimate owner. Otherwise, an attacker could take the smartphone during the enrollment phase, train the model with his data, and get access to the system, before the legitimate owner reports the loss of his device.

In the authentication phase, the device starts in a locked state and begins to gather data for analysis continuously (Figure 4.2). If the classification detects an appropriate certainty of the device being in possession of its legitimate owner, the authentication is considered successful and the device switches to an unlocked state. The device gets locked if the likelihood falls below a threshold. As a fallback, the user should be able to authenticate using standard methods like a PIN or two-factor authentication.

The steps for the enrollment phase could be processed locally on the phone, but model training and evaluation in the cloud could be possible, too. The authentication phase should not rely on a network connection and should be processed only locally. The authenticated “locked state” does not necessarily need to correspond with the lock state of the smartphone’s operating system. It is meant more generic and can, e.g., reflect the authentication state against a third party system that communicates with the phone over network or Bluetooth.

4.2 Use Case

Different use cases for smartphone-based CA are tightly bound to the smartphone usage itself, where, e.g., login data is automatically provided to applications running on the smartphone, or where the phone is automatically locked if it is taken out of the owner's hands.

The general use case for CA considered in this thesis, on the other hand, is focused on authentication against third-party systems unrelated to the owner's usage of the smartphone. Such systems could be services that create an alert if a smartphone was taken from its legitimate owner, or physical access control systems like safes, which could use the presence of an authorized person as an additional authentication factor. The proof of authentication could be provided as a token over network or radio contact to the third party system performing the authorization, but this transmission is out of this thesis' scope. Nevertheless, the differentiation between those use case types has an impact on the design decisions presented in the following section.

I will reason in the following section, that the use case considered in this thesis has some different requirements than use cases that are bound to the smartphones' usage. The targeted operational area is office space and also has an impact on the requirements. It has to be considered that the behavior of the users and their smartphone usage might differ in business situations from private situations.

4.3 Design Decisions

Some decisions strongly influence the implementation as well as the productive operation of the system. It is essential to justify the decisions transparently, as there is no clear right or wrong for choosing between the various design options.

4.3.1 Active vs. Passive

Passive CA is restricted to sensor data that is produced independently of smartphone usages, such as audio signals, location data, or inertial sensors, which data can be collected continuously. Active CA depends on active usage of the smartphone, e.g., texting, browsing, navigating, or calling, which produces additional data, like voice recordings, touch data, or application usage information.

Obviously, active CA is easier to accomplish, as there is additional data available. Also, this additional data produced by active interactions can be directly attributed to

the smartphone user, while there are multiple possible sources for the data gathered during passive CA. E.g., the touch data generated while texting via the devices' onscreen keyboard is very likely produced by the current user, but the movements detected by an accelerometer could be caused by the current user as well as by a moving vehicle which transports the user.

The described use case (see Section 4.2) requires constant availability of the authentication state, independently of the smartphone's usage. Therefore, this thesis targets passive CA. Nevertheless, data produced by active interactions could enhance the reliability of the system during phases of active smartphone usage. Such a combination of both approaches is out of scope for this thesis.

4.3.2 One Class vs. Binary Classification

The main task during CA is to classify the incoming sensor data of the device as produced by the owner (positive class) or as produced by one of an unlimited number of possible impostors (negative class).

One possibility is to interpret this requirement as a classical binary classification problem. The selected model could be trained using data of the positive class against a large number of impostors in the negative class (one vs. all). This approach would make sense, if the model could generalize good enough to detect even unknown impostors during testing, or if the use case scenario does not include any unknown impostors at all.[14] However, it would be difficult to predict, how the model would behave with input data from an unknown impostor that's very different from the training data. It also would require access data from all known users available during training or updating the model, which in practice probably would be implemented by centralized data collection and model training, followed by distributing the models to the smartphones. The challenges of such a setup are operational costs for central data collection plus model distributions and privacy concerns, as the data from all users have to be stored in a central location. On the other hand, this solution would provide more control to improve data quality, model training, and testing, because the data classified on the device could later be leveraged to improve the model.

Another option would be to interpret the situation as a one class problem, where only data of the current owner is available. With algorithms of the novelty detection or anomaly detection, a model can be trained using only the owner's data. Such a model can estimate if new data was produced by the same person or not. This approach is usually more difficult than the binary classification because the suitability of the

[14] A possible scenario could be a facility, where only a limited amount of known people get access via conventional access control systems, and where CA is only used inside this closed system.

features to distinguish between users cannot be assessed during training time. On the other hand, a one class model could be trained locally on the user's smartphone. Therefore, it would not have the operational or privacy-related downsides.

Of the reviewed studies (see Table 3.1), two thirds used the one class approach for the final classification. Most of them additionally leverage multi-class data for optimizing the feature preparation, either through manual feature engineering (e.g., Sitová et al., 2016), for creating a generic background model as a basis for the individual owner models (Neverova et al., 2016), or by training a model to generate deep features as input for the one authentication class model (Centeno et al., 2017). Such an approach combines operational benefits from the one class approach with the benefits of having more information available for model building. The privacy concerns for the still needed central collection of user data could be avoided by recruiting dedicated persons who could be asked to upload their data manually after giving their consent. For those reasons, I considered such an approach as the most suitable and selected it for implementation.

4.3.3 Sensor Selection

While all sensors which provide information about the smartphone's user could be leveraged to improve the reliability of the authentication system, there are restrictions to be considered. As already mentioned above, some data sources require active usage of the smartphone, which I previously discarded for this thesis (see Section 4.3.1). Reading data from the GPS sensor needs appropriate permissions and is not very useful in buildings. Reading information from other applications, like messengers or keyboards, additionally requires elevated access rights to the device, which might not be desirable for security and warranty reasons. Using the microphone or camera as sensors could invoke privacy issues as well. Other sensors, like gyroscope or barometer, are often missing in cheaper smartphone models. Another aspect is the energy consumption: e.g., the continuous use of the GPS sensor would result in a significant decrease in battery life. Patel et al. (2016, pp. 51ff) describes more potential data sources for smartphone-based CA.

Regarding the limited scope of this thesis, I decided to use only inertial sensors: gyroscope, accelerometer, and magnetometer. They are easily accessible, as they don't need any permissions on the devices operating system, they are present on many current devices, they do not collect sensitive data from the surrounding and have an acceptable energy consumption. And they are available in the public datasets (see Section 3.1). It is important to note that those sensors are a good starting point into CA for the mentioned reasons, but data from other sensors could and should be included in a productive system to improve its reliability.

4.3.4 Manual Feature Construction vs. Deep Features

As stated in Section 3.2.2, the majority of related studies constructed the features for their model using manual feature engineering, while four studies leveraged deep learning algorithms. As the generation of deep features has the potential to combine the process of manually optimizing signal filtering, feature construction, and dimensionality reduction in a single automated step. This approach is quite tempting, the authentication performance reported by the authors (see Table 3.1) seems quite promising and at least on par with other approaches. I decided to implement such a deep feature learning approach, as one major downside of using deep learning, missing interpretability, is not part of the evaluation criteria chosen for this thesis (see Section 4.4),

4.4 Evaluation Criteria

It is good practice to define concrete evaluation criteria upfront to asses the performance of a machine learning model. It facilitates more objective judgment and improves the focus while proceeding with the concept and implementation phase.

A variety of metrics are available to support the quality assessment and performance measurement of machine learning models. The challenge is to pick the right ones for the given use case. The criteria used in this thesis are described below and have been selected to cover three aspects:

1. Reliability of the approach as an authentication method.
2. Comparability with prior research in the same domain.
3. Constraints regarding practical application in potential use cases.

Other possible criteria had to be left out of this thesis, but have been studied by other researchers in the domain. Those include the evaluation regarding resilience against planned attacks (Kumar et al., 2015; Kayacik et al., 2014, p. 8–10), the adaption to changes in the individual gait pattern over time (Horst et al. (2016); Kayacik et al., 2014, pp. 7f), the interpretability of the created machine learning model (Horst et al., 2018) and the energy consumption needed for authentication (Sitová et al., 2016, pp. 887f). The computational complexity of the system is another commonly considered criterion, especially for the part of the CA system running on the mobile device, where processing power and battery are limited. To get meaningful results, it would be necessary to measure the power consumption during classification on real smartphones and set it into relation to the power consumption needed for gathering

the sensor data. But looking at the performance improvements of smartphones as well as machine learning frameworks, this criterion is considered to be less important in the early stage of this study and not evaluated.

4.4.1 Authentication Reliability

Falsse Acceptance Rate (FAR), False Rejection Rate (FRR) and Equal Error Rate (EER) are standard metrics used to describe the performance of authentication methods (see Section 2.1.4). As the thresholds can be adapted to favor one of those metrics on behalf of the other one, reporting FAR and its trade-off FRR seems not to be useful in this state of model evaluation. The use case of this thesis does not contain requirements regarding FAR vs. FRR. Therefore those metrics will be discarded in favor of EER as a more generic and objective metric.

4.4.2 Training Delay

A classifier for CA based on behavioral metrics needs to be trained upfront with a certain amount of behavioral data from the user who is going to be authenticated. A new user, who is not yet known by the system, first needs to produce behavioral data by carrying the smartphone around. This data is used to train the classifier. The authentication functionality becomes available after this enrollment phase when the classifier has reached an accuracy threshold. The time-span of data-gathering needed to train the classifier upfront is the training delay. (Khan et al., 2014, pp. 266f)

The training delay is an essential metric for practical applications, as it constitutes a limitation on potential use cases. The training delay depends on the desired accuracy threshold to be reached and correlates with the availability of behavioral data that the user produces during this time. It is necessary to consider these dependencies for the interpretation of this criterion.

Khan et al. (2014, pp. 266–268) used the training delay as one of their criteria for comparative evaluation of six different CA approaches on the H-MOG-Dataset. Depending on the approach, a training delay between 165 seconds and 3.2 weeks was needed to reach a classification accuracy of at least 80%.

4.4.3 Detection Delay

During the authentication phase, the trained classifier has to predict whether new incoming sensor events were produced by the legitimate user or by an impostor. As it is not feasible to accomplish this with a single sensor measurement, multiple

consecutive measurements are needed to decide reliably if the device is in possession of its owner or not.

The time-span needed to gather enough consecutive events to do a classification is called the detection delay[15]. An average detection delay of 30 seconds means that a CA-system usually needs 30 seconds of consecutive data before being able to detect that the smartphone is in possession of an illegitimate user. (Khan et al., 2014, p. 267)

Like the training delay, the detection delay also has implications on potential use cases. E.g., the CA-system would be of limited usefulness if it would need 60 seconds of data before detecting an impostor, but the impostor could use the smartphone to gain access to critical systems in less than 60 seconds. Like the training delay, the detection delay also depends on the desired accuracy threshold and the device movements during this time.

In their comparison (Khan et al., 2014, p. 268) reported detection delays between 2 and 10 seconds for the fastest approach and 15 minutes for the slowest approach.

4.4.4 Evaluation Setting

Selecting the metrics to use for model evaluation is important, but deciding how the setting for evaluation should look like is crucial. It strongly influences the results. I have already described the different configurations used in related studies (see Section 3.4), along with some difficulties I see. As this thesis aims at getting a realistic view of the feasibility of smartphone-based CA in a real-world application, the evaluation setting should reflect this goal.

In general, data splitting into training, validation, and testing sets should be used. The validation set is used to optimize preprocessing and the model's hyperparameters. The testing set is used only for the final model evaluation. It is crucial that no information from the testing set leaks into the training or validation set and vice versa. That does not only include data samples themselves, but also indirect information like scaling parameters, which might hold information about the distribution of the data in the target classes.

For ensemble approaches, where a generic model is trained upfront on data from all users, and afterward a one class model is trained as the authentication classifier, it would be desirable to include additional splitting, to train, validate and test both models with separate data. However, due to the limited amount of data available it

[15] Sometimes also the term "classification delay" is used.

might be a valid compromise to reuse data of the first model's optimization to train or validate the second model, as long as the dataset used for the final testing of the authentication model is kept strictly separated.

A realistic and probably the most common attack scenario is that another individual takes the owner's phone. Therefore, testing samples from the owner against samples from another single user per run is appropriate. As we consider the attacker to be unknown, it's essential, that no information of those impostors leaks into training data. To get reliable results, the test owner vs. impostor should be repeated multiple times per owner with different users as an impostor.

Regarding the different scenarios of data collection, it is appropriate to use an equal amount of samples per walking and sitting scenarios from both the owner and impostors to avoid bias towards one of the scenarios in the final testing. It would be interesting to know how the distribution of the scenarios in training and testing sets could influence the results. But this is out of the scope of this thesis.

K-fold cross-validation should be used at least for testing the authentication classifier. $K = 10$ would be desirable to reduce the bias introduced in case of the training set is too small. Depending on the training speed, it might be necessary to compromise and decrease K. Usually 5–10 folds are recommended. (Hastie et al., 2009, pp. 242f)

4.5 Model Selection

The following sections describe the two models used for implementation. I focus on the reasons why I selected those two models and on their central principles. For detailed descriptions, I refer to other literature to stay in scope. The implementation details are included in the documentation of the experiments (see Chapter 5).

The general idea is to select as baseline model and compare it against the reimplementation of a more promising model from related studies. Originally, I planned to improve the reimplemented approach afterward, after the performance evaluation, I decided to take a step back and varied the evaluation settings instead.

4.5.1 One Class Support Vector Machine

An SVM is a model used for classification and regression problems. It belongs to a class of classification models, which model the discriminators of the classes, instead of the classes themselves. In a two-dimensional feature space with data points from two classes, it finds a line that separates the classes from each other as reliable as

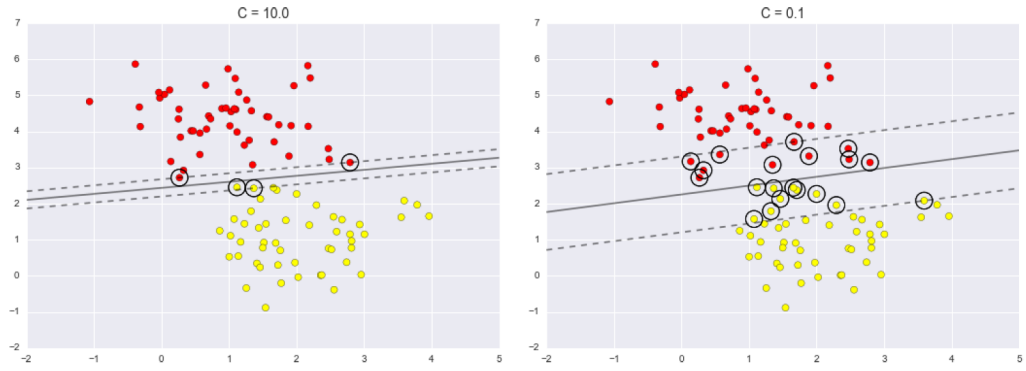


Fig. 4.3.: Example of an SVM classifier fitted to data with two classes (red, yellow), showing margins (dashed lines) and support vectors (circles), for a fit with a higher (left) and lower (right) C value. Source: VanderPlas (2016, p. 416).

possible while maximizing the margin between the line and the classes' data points (Figure 4.3). In higher dimensions, the discriminating line becomes a hyperplane. (VanderPlas, 2016, p. 405)

The data points touching the margin are pivotal elements for that fit and are called support vectors. The "sharpness" of the separating line can be tuned by a so-called soft margin parameter C, which controls the number of data points allowed inside the margin to enable a better fit (VanderPlas, 2016, pp. 415f). Additionally, different kernels can be used to project the data points in higher dimensional space to improve the fit for nonlinear boundaries. (VanderPlas, 2016, pp. 411f)

The idea of the supervised learning algorithm SVM, which needs classified data during training, was transferred into the field of unsupervised novelty detection. Instead of fitting discriminators between multiple classes, a discriminator is fitted between regions with and without data points from a single class (Schölkopf et al., 2000, pp. 582f). For regularizing the algorithm, parameter ν (nu) was introduced (Figure 4.4). Its value is bound between 0 and 1 and is directly connected to the fraction of detected outliers, e.g., if $\nu = 0.1$ the model will be fit such that at least 10% of the data points are considered as outliers (Schölkopf et al., 2000, p. 588).

For in-depth explanations of the SVMs, I recommend Hastie et al. (2009, Chapter 12.3), while the original paper by Schölkopf et al. (2001) describes the background of OCSVMs.

I selected the OCSVM as the baseline model for three reasons. First, the model was already applied in multiple related studies (see Section 3.3.2), which indicates the general suitability of the model for the given problem and, despite the variations regarding the evaluation approaches (see Section 3.4), might provide rough reference

[16] Source code: `/notebooks/chapter-4-5-1-ocsvm-parameter-demo.ipynb`

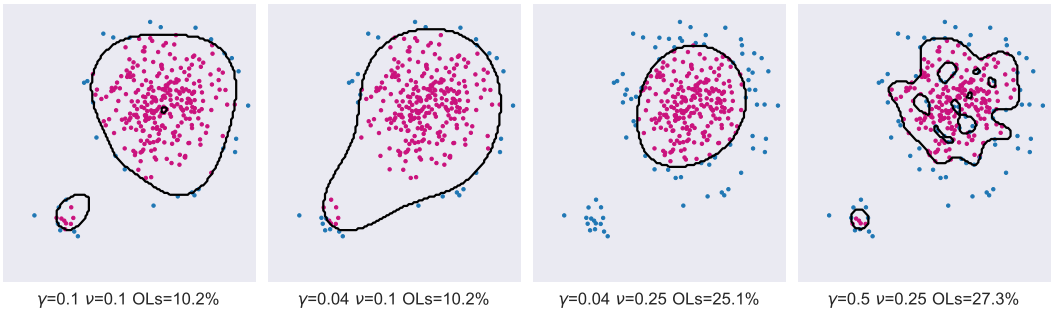


Fig. 4.4.: Example for the influence of OCSVM's hyperparameters ν and γ when fitted with dummy data using an RBF kernel. The decision boundaries (black) separate normal data (magenta) from outliers (blue). Depending on parameters' values, the small cluster in the lower left will be interpreted as a separate cluster of normal data (1st and 4th plots), as part of a single but larger region of normal data (2nd plot) or as outlier (3rd plot).[16]

metrics. Second, Centeno et al. (2018), who's main approaches were chosen to be reimplemented, also used OCSVM as a baseline model. As Centeno et al. (2018) also used the same H-MOG dataset, the results should be directly comparable. And last but not least, OCSVM also was used by Centeno et al. (2018) as the classification model of their primary approach which I chose for reimplementation. That leads to synergy effects reducing effort.

4.5.2 Siamese Network

A Siamese network consists of two separate subnetworks, each with its own input vector, and a single output. Both networks share the same weights. The output value is derived from the distance between the output of the subnetworks and corresponds to the similarity of the two input vectors. (Bromley et al., 1993, p. 740)

In the architecture (Fig. 4.5) used by Centeno et al. (2018, pp. 3f), the pairs of input vectors $x^{(i)}$ and $x^{(j)}$ have a positive label $y^{(i)}$ if both inputs were generated by the same user or a negative label in case the two inputs were generated by different users. Both input vectors are transformed by CNNs into vectors of lower dimensionality, and the distance d between those vectors is calculated. A so-called contrastive loss function $\mathcal{L}$ takes both distance d and pair label $y^{(i)}$ as inputs and produces a loss value, which is high if d is low and the label is negative, or d is high, and the label is positive. If d is low and the label is positive, or d is high and the label is negative, the loss becomes low. That effectively forces the networks to identify and learn aspects of the data that are useful to distinguish between samples from the same user and samples from different users.

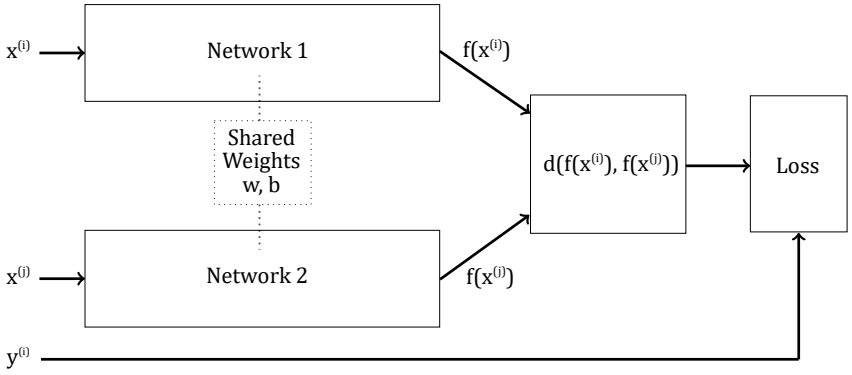


Fig. 4.5.: Architecture of a Siamese network. Source: Centeno et al. (2018, p. 3).

To leverage the Siamese network for CA, it is upfront trained with both positive and negative pairs from a labeled training dataset of known users. Afterward, distance and loss functions get discarded. Only one of the networks[17] is used to output the lower dimensional features only. Those deep features are then used as input for training and classification using an OCSVM, just like in the baseline model described above (see Section 4.5.1). (Centeno et al., 2018, pp. 4f)

I selected this model as primary for reimplementation because it perfectly fits the idea of extracting information from a larger dataset containing multiple users into a generic model, which afterward can be leveraged for a one class approach with only owners' data (see Section 4.3.2). The input for a Siamese network is raw sensor data and might not need effortful feature construction. Additionally, I saw a potential for enhancement by testing different types of ANNs for the subnetworks that can capture temporal aspects of the data, e.g., LSTM[18] or CWRNN. Last but not least, the results reported by (Centeno et al., 2018, pp. 5f) were promising, and the Siamese network architecture was new and exciting to me.

[17] It does not matter which of the two networks is used: as they have the same architecture and share weights, they are technically identical.

[18] While writing this thesis, a study using Siamese Network with LSTM for CA was published by Deb et al. (2019).

5 Experiments

This chapter describes the reimplementation of two approaches presented by Centeno et al. (2018), leveraging an One Class Support Vector Machine (OCSVM) and a Siamese Convolutional Neural Network (CNN). Because of missing information in the original paper, I had to make educated assumptions regarding the implementation, which are also documented clearly.

The evaluation of both models is presented along with arguments, that some steps used for preprocessing and testing applied in the original study have specific weaknesses which lead to poor performance in more realistic scenarios.

Last but not least, I propose a variant of the Siamese CNN approach that is better applicable in a real-world scenario and leads to improved performance compared to the original approach in the same scenario.

The following documentation is intended to contain all necessary information for the reproduction of the experiments. The specifications of the desktop computer used as a processing device include an Intel Xeon 3.60GHz CPU, 64GB RAM, Windows 10 and a 500GB SSD. An included CUDA capable Nvidia GPU was not leveraged for deep learning to stay more independent of the hardware and make it easier to execute the same code on other machines. All experiments were implemented using Python 3.6 and commonly used public libraries. The source code has been made publicly available on GitHub (Büch, 2019). All file paths referenced in the following are relative to the root directory of this repository.

5.1 Project Setup

The project is structured in three main sections: All data, except for temporary files, is stored in own section (`/data/`), nested into four subsections reflecting its state (`/data/external`, `/data/raw`, `/data/interim` or `/data/processed`). Steps related to initial data transformation and data loading are implemented as Python modules (`/src/`). Data Exploration, modeling and evaluation are implemented in Jupyter (`/notebooks/`), because the clarity and traceability those notebooks provide by combining source code, documentation and output are considered to be beneficial

given the experimental stage of this project. Those notebooks were also used to produce the data visualizations depicted in this thesis.

The repository contains a file with meta information denoting the versions of Python itself as well as of the imported third-party libraries (`/environment.yml`) which can be used reproduce the programming environment. It can be passed to the package manager of the Anaconda Python Distribution[19] to recreate the Python environment I used for development. The enclosed documentation (`/README.md`) contains a detailed description of all necessary steps.

5.2 Dataset

The H-MOG dataset is the basis for the following experiments. It has been collected and made available for public download (Yang et al., 2014b). The dataset is described in detail by its authors Yang et al. (2014a).

The data was collected in a controlled environment where multiple sessions of smartphone usages were recorded for 100 users. In each session, the users performed predefined tasks of the 3 types “reading”, “writing” or “map-navigation”. Additionally, the scenario was varied as each task had to be performed while “sitting” and “walking”. This results in 6 different session types per user. Each session was repeated 4 times resulting in 24 sessions per user. The sensor data in the dataset covers data of the accelerometer, gyroscope, and magnetometer at 100 Hz frequency. Also, sensor data from screen interactions like touch, key-press, scroll, pinch, and stroke are provided, but not relevant in the scope of this thesis.

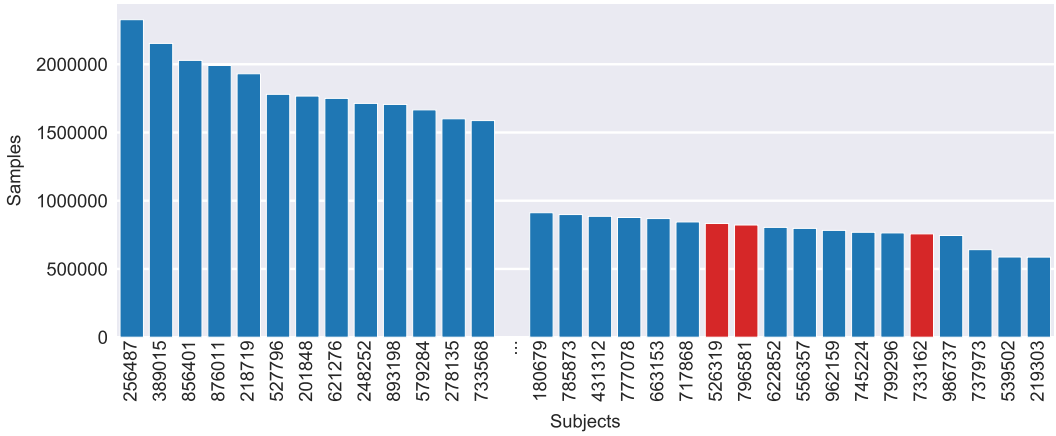


Fig. 5.1.: Distribution of sample counts among subjects. The three subjects with incomplete sessions are marked red. Only the tails of the distribution are depicted here, in the center the sample counts decrease quite linearly.

[19] See https://www.anaconda.com/distribution/.

As the usage of the H-MOG dataset is already documented in multiple papers, its exploration here is focused on the completeness of the data. In a first step, the relevant CSV-files from the dataset have been converted into tables in a single file of the Hierarchical Data Format (HDF) for performance reasons[20]. During implementing the conversion process, I identified four issues:

1. Six sessions (9 to 14 of user 733162 have empty `Accelerometer.csv` files.
2. Some sessions have non-unique IDs in `Activity.csv`, resulting in an unclear assignment of the metadata to the sensor data (e.g., for ID 100669012000002 in `/100669_session_1/Activity.csv`).
3. User 526319 and user 796581 have only data of 23 instead of 24 sessions.
4. The name of the data folder `/hmog_dataset/207969/` is inconsistent with its containing data, where the user is named 207696.

After the format conversion, I performed a data exploration, of which some results are presented in the following.[21]

The inspection of value counts per user (Figure 5.1) revealed an imbalanced distribution. For an unknown reason, the subjects with the most sensor data have over 3 times more data than the subjects with the fewest sensor data.

A histogram of the sessions' durations (see Figure 5.2) revealed that the mean duration of a session is around 8.6 minutes, and most of the sessions are even shorter. That contradicts Centeno et al. (2018, p. 4) stating that each session of the H-MOG datasets lasts about 15 minutes.

[20] See source code: `/data/transform_to_hdf5.py`.
[21] The full investigation, including, e.g., the complete versions of the following partial plots, can be found in the source code: `/notebooks/chapter-5-2-1-exploration-hmog-statistics.ipynb`.

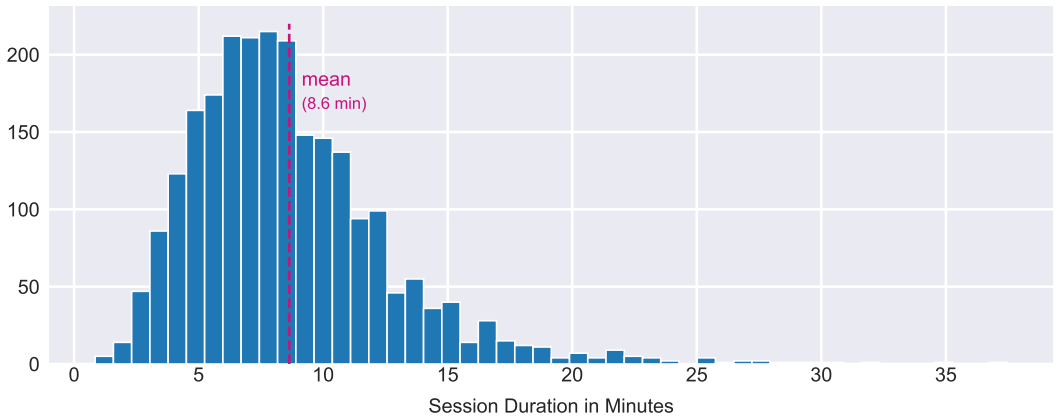


Fig. 5.2.: Histogram of the durations of all sessions in the H-MOG dataset.

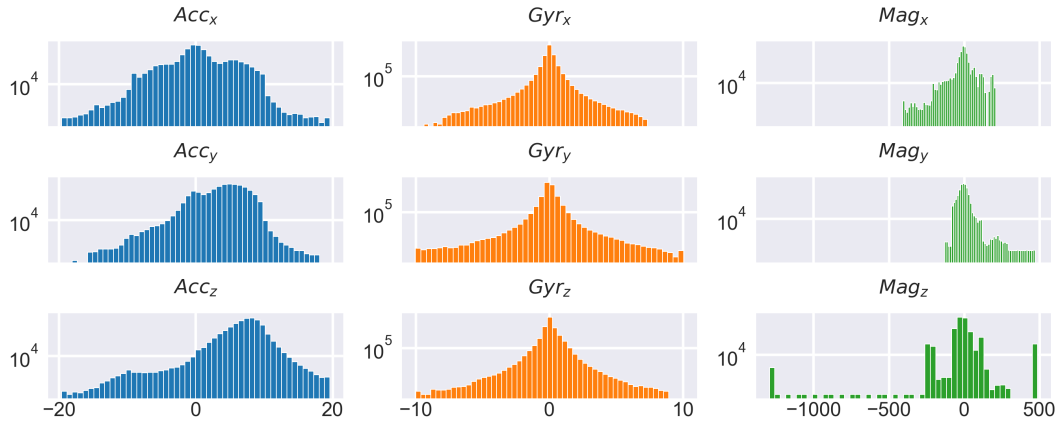


Fig. 5.3.: Distribution of sensor values of all subjects, plotted with log scale.

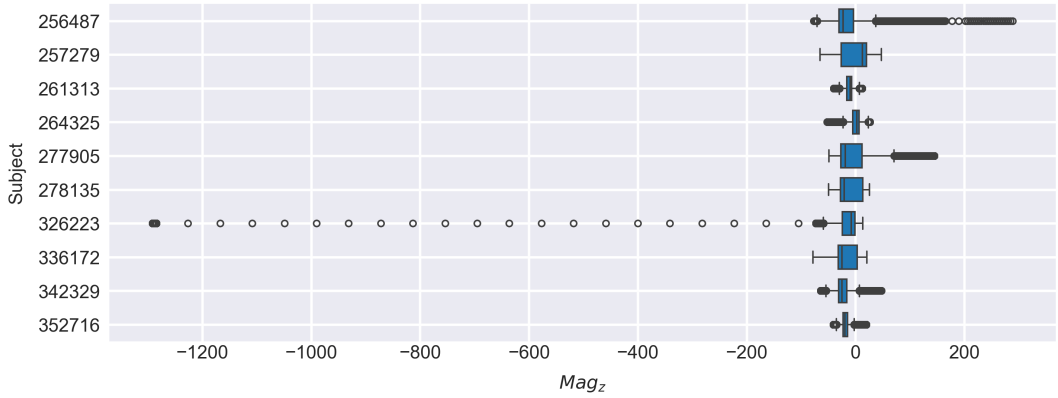


Fig. 5.4.: Exemplary distributions of magnetometer values along z-axis for 10 random subjects, with outliers visible for subject 326223. Such conspicuous values are found for all three axes, although less extreme.

Plotting the distribution of sensor data (Figure 5.3) of the whole dataset revealed, that gyroscope's values are very evenly distributed around zero, while accelerometer's distributions are left-skewed towards 10 (especially on y- and z-axis), because of the gravity component. The x- and y-axis of the magnetometer are quite unevenly distributed, and the z-axis includes some severe outliers. When I inspected those magnetometer values subject-wise (Figure 5.4), it became clear that only a few users produce those outliers.

5.3 Initial Data Preparation

The original format of the H-MOG dataset is a ZIP file containing Comma-separated values (CSV) files organized in nested folders for users and sessions. The performance of extracting and reading 28 GB of those files from a hard drive is quite weak. Additionally, the data from the three sensors accelerometer, gyroscope, and magnetometer is provided in separated files, requiring expensive joins of the sensor values.

For those reasons, I implemented an initial transformation process (Figure 5.5) to read the data from the CSV files, perform the necessary joins, remove unneeded attributes and store the resulting tables in the Hierarchical Data Format (HDF) format for faster read access to the data in the desired structure. The implementation of this process has been balanced between computing performance, reusability, and clarity, and is described in the following.

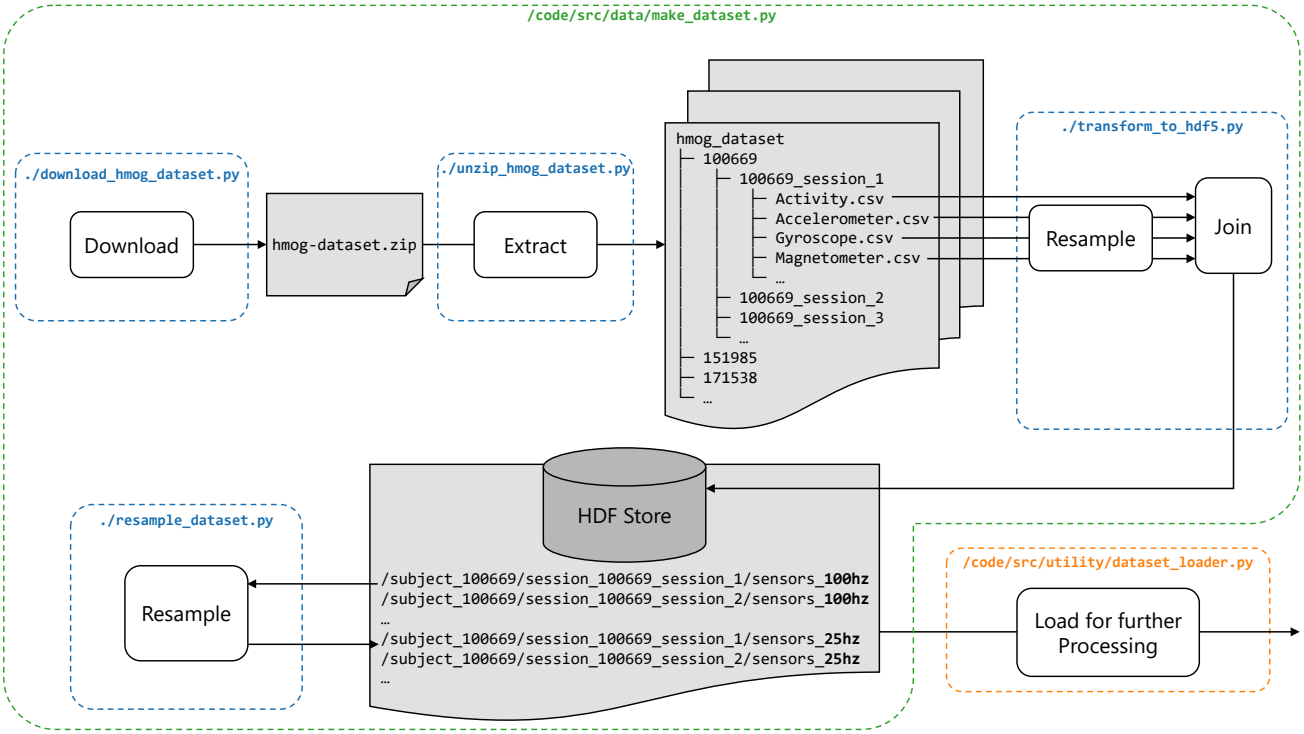


Fig. 5.5.: Process for initial data preparation mapped to corresponding Python files.

While **joining the sensors' data**, I noticed, that the three sensor data files can, for a single session, contain different numbers of sensor values and are annotated with different timestamps. Investigating further, I discovered gaps of varying length in the sensor readings: The sensor data is not uniformly sampled and would generate time shifts between the sensor values if the data would be joined on row indices. Without information, how those issues were handled by other researchers who worked with this dataset (see Table 3.1), I decided to transform the data into a uniform sampled time series. First, the values of each sensor are resampled to 10 ms based on their timestamps, and existing gaps are interpolated linearly. Afterward, the resampled values of the three sensors are joined into a single table. This created sequences with incomplete sensor measurements on the start and end of the sessions, which were truncated.

Metadata about the "task" performed by the proband and the "scenario" of the recording session available in the `Activity.csv` files are joined with the sensor data. The H-MOG dataset differentiates between 24 different tasks, with own IDs.

Each ID can be mapped to 1 of 6 different task-scenario-combinations, where a task can be “write”, “read”, and “map navigate”, and a scenario can be “walking” or “sitting”. I discarded the information about the exact task ID (1–24), because this information was not used by Centeno et al. (2018), and joined the less granular task-scenario-combination (1–6) instead. The metadata in the `Activity.csv` files does not cover the entire time span of the sessions’ sensor recordings: There are gaps before, after and between individual subtasks, effectively leading to periods without any meta information. Again, in none of the related studies, this issue was indicated, and it remains unclear how this situation was handled in previous studies. As a single session always consists of the same task-scenario-combination, I decided to ignore those gaps and assigned all sensor measurements of a session to the same task-scenario combination. Measuring the effect of this decision would be desired, but is not in the scope of this thesis.

As a **sampling rate**, Centeno et al. (2018, p. 5) stayed with the original frequency of 100 Hz for the OCSVM baseline model and tested 100 Hz as well as a subsample to 25 Hz for their Siamese CNN approach. For this ensemble approach, they reported better results using the lower frequency. The original study does not explain how exactly the subsampling was performed. Regarding the **sampling scope**, the correct way is to sample each users’ sessions individually. Resampling multiple sessions per user together in a single run would introduce a leakage of information between the sessions through the overlapping sample-range at the start and end of the sessions. As the sampling method, I calculated the mean over a sliding window of 4 samples. The resampling was implemented as a step in the initial data transformation (see Section 5.1) to avoid repeating this expensive task during the development of the models afterward.

An option to **filter out users with incomplete data** was implemented in a helper module, that is used to load the data from the HDF file for further processing.[22] During implementation, I discovered discrepancies between the related studies regarding the dataset’s completeness:[23] Centeno et al. (2018, p. 4) used the H-MOG dataset but excluded 10 of its 100 users. They stated that 90 users performed all 24 sessions of about 15 minutes in length. Those sessions consist of 8 reading sessions, 8 writing sessions, and 8 map navigating sessions. Sitová et al. (2016, pp. 879f), Neverova et al. (2016, p. 1814) and Shen et al. (2018, p. 59) did not mention excluding any of the 100 users. Li et al. (2018, p. 32560), also using the H-MOG dataset, reported 2 users with “unusual” data, without revealing more details.

[22] Source code: `/src/utility/dataset_loader.py`.

[23] Different versions of the H-MOG dataset would explain those irregularities, but to my best knowledge, no other than the currently available version has been published.

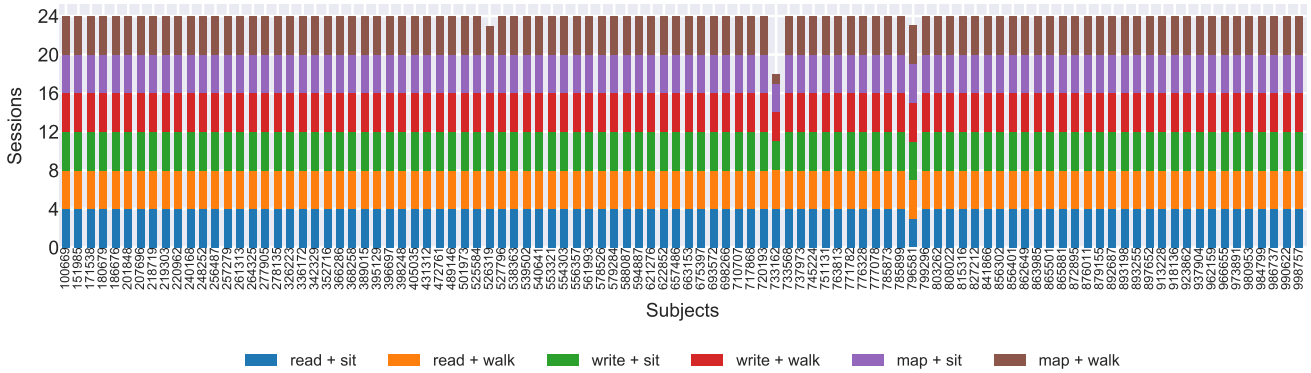


Fig. 5.6.: Sessions per user for all 100 subjects of the H-MOG dataset, stacked by task-scenario-combination. Five sessions with incomplete data from user 733162 have been excluded beforehand, the two subject 526319 and 796581 are missing one session each.[24]

The results of my data exploration do not conclude with those statements. As already shown in the previous section (see Section 5.2), the sessions are usually shorter than 15 minutes, with a mean duration of 8.6 minutes (see Figure 5.4). Of the 100 users, only 2 did not perform all 24 sessions. Including the subject with missing accelerometer data for multiple sessions, there are 3 users with incomplete data, not 10. The distribution of the session and their types (reading, writing, map navigating) did not reveal further irregularities (Figure 5.6).

As it is unknown, which specific subjects Centeno et al. (2018) excluded, but I wanted to have the same number of 90 users to have a comparable diversity in the data, I decided to exclude the three users with incomplete data (526319, 733162 and 796581 along with users having an unusually large amount of data (256487, 389015 and 856401) or an exceptionally small amount of data (219303, 539502, 737973 and 986737) (see Figure 5.1).

The normalization of the data was one of the most significant issues during implementation. The information available about how this step was executed in the original study is limited: The only information provided by Centeno et al. (2018, p. 3) is, that they performed channel wise min-max normalization into a range between 0 and 1, Centeno et al. (2017) does not indicate any normalization at all. On which scope the normalization was applied is unknown: It could be applied on the whole dataset at once, for every subject individually, session-wise or even for every window of samples. But not all variants make sense regarding the application scenario. Also, it is unclear, if or how a differentiation was made between normalization of training and testing sets: Was the data normalized before or after splitting it into training and testing sets? A legitimate approach would be to fit a scaler on the training set and normalize both training and testing sets using the same scaling parameters to avoid leakage of information between the sets. An additional pitfall

[24] Source code: `/notebooks/chapter-5-2-1-exploration-hmog-statistics.ipynb`.

to avoid is the use of different scaling parameters for samples from the owner and impostors during testing. All samples have to be normalized equally, as it is not possible in a real-world application to distinguish between samples of the owner and the impostor in this stage.

Through correspondence[25], I learned that the normalization was performed subject-wise and for all three models (OCSVM, autoencoder, Siamese CNN) equally.[26] It remains unclear, if and how the normalization was performed differently during the testing phase than during the training phase. Due to those uncertainties, I decided not to normalize the data during the initial data preparation, but instead treat the normalization as an unknown parameter to be investigated further during modeling in the Jupyter notebooks.

5.4 Modeling OCSVM

The OCSVM baseline model (see Section 4.5.1) in this thesis was build according to the information published by Centeno et al. (2018). As they used the same public H-MOG dataset, results were expected to be comparable but not equal, as not all relevant aspects of the modeling are described in the original study. E.g., the provided model parameters include a range used for γ, but not its actual value (Centeno et al., 2018, p. 5), information on the parameter C necessary for the Radial Basis Function (RBF) is missing. Also, Centeno et al. (2018) seem not to clearly distinguished between validation and testing sets and did not mention, how exactly "the best accuracy rate obtained between all the results" (Centeno et al., 2018, p. 5), which they report as the performance metric, was collected.

For those reasons, I took the information provided by Centeno et al. (2018) as the basis for my implementation of the modeling and evaluation process (Figure 5.7), but included additional steps not described in the original paper, notably:

- Two variations of the normalization step.
- A dedicated step for hyperparameter optimization using a validation split.
- Three evaluation steps for reporting authentication performance, detection delay, and training delay.

Those and other adjustments are described in the following in detail, while Appendix A.3 summarizes the information provided by Centeno et al. (2018) and my

[25] E-Mail by M. P. Centeno, 14th Apr. 2019.
[26] This concludes, e.g., with Sitová et al. (2016, p. 880), who performed scaling based on standard deviation but also subject wise and separately for every single feature.

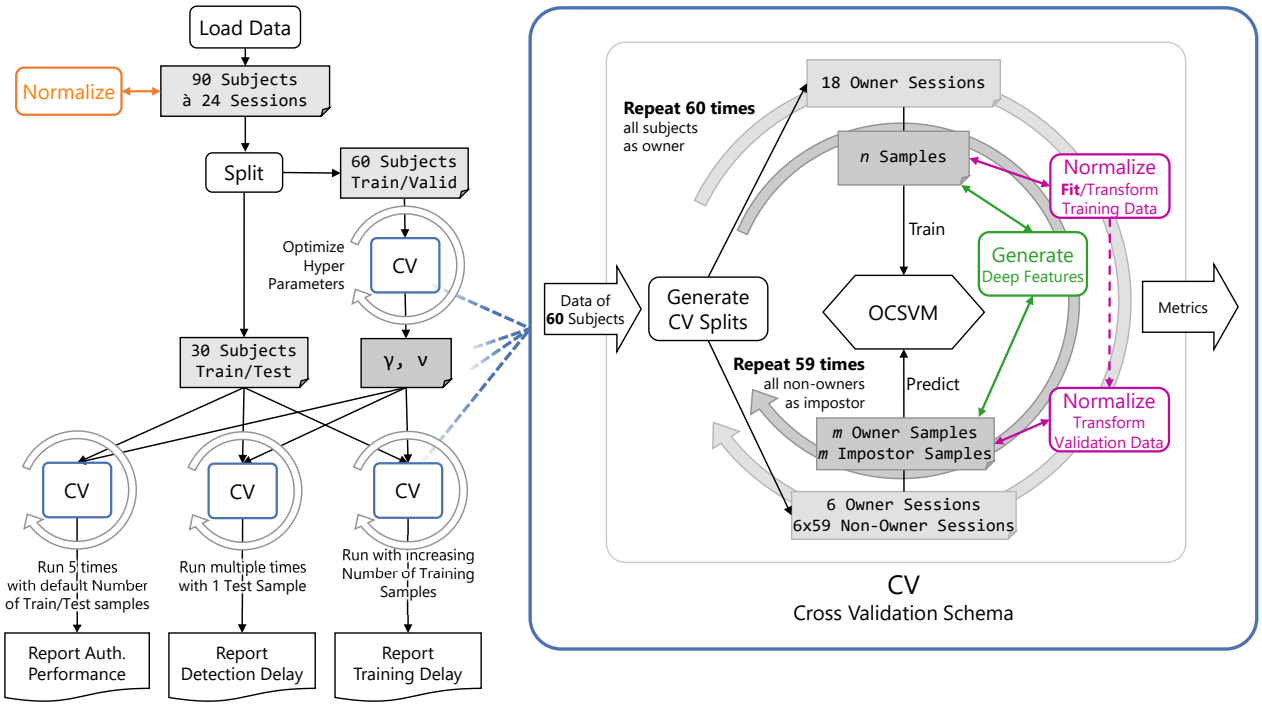


Fig. 5.7.: Implemented process for the OCSVM (left) and details of the cross validation scheme used in several steps (right). Three steps were executed optionally: The normalization step right after data loading (orange) was applied for the "naive approach", while in the "valid approach" the normalization (purple) was applied after selecting training & testing/validation samples. The generation of deep features (green) was only used for the Siamese CNN model (Section 5.5).

open questions along with the assumptions and modifications I made during the implementation (Table A.3).

The **Cross Validation (CV)** used in the original study for training and testing is described by Centeno et al. (2018, pp. 4f) quite well. The training of the OCSVM is performed using data from one of 30 randomly selected subjects.[27] 18 sessions per subject are reserved for training; 6 sessions are held out for testing (Figure 5.8). Both folds consist of sessions with the same proportion of the six task-scenario-combinations (training: 3 x 6 combinations; testing: 1 x 6 combinations). The testing is performed in one versus one attack scenario, where 6 of the legitimate user's sessions are tested together with 6 sessions of another subject as an illegitimate user. 29 of those attack scenarios were tested per owner, covering all possible combinations with the other subjects. The whole procedure is repeated with each of the 30 subjects as a legitimate user, resulting in 870 overall test scenarios.

The study does not provide information, how the **hyperparameters optimization** was performed, and if or how a validation split was used. Therefore, I implemented CV as follows: First, the 90 selected subjects of the dataset are split randomly into two

[27] Only 30 subjects were chosen because the remaining 60 subjects were used to train the Siamese CNN in the main approach, and the number of subjects should be equal for both approaches.

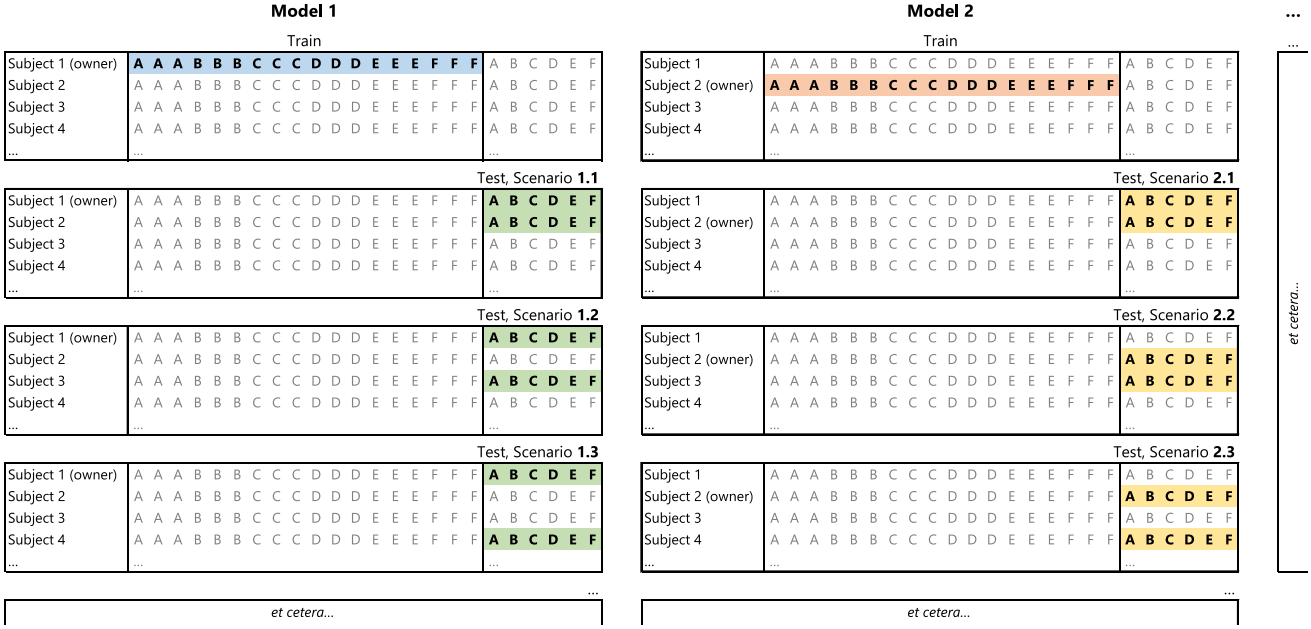
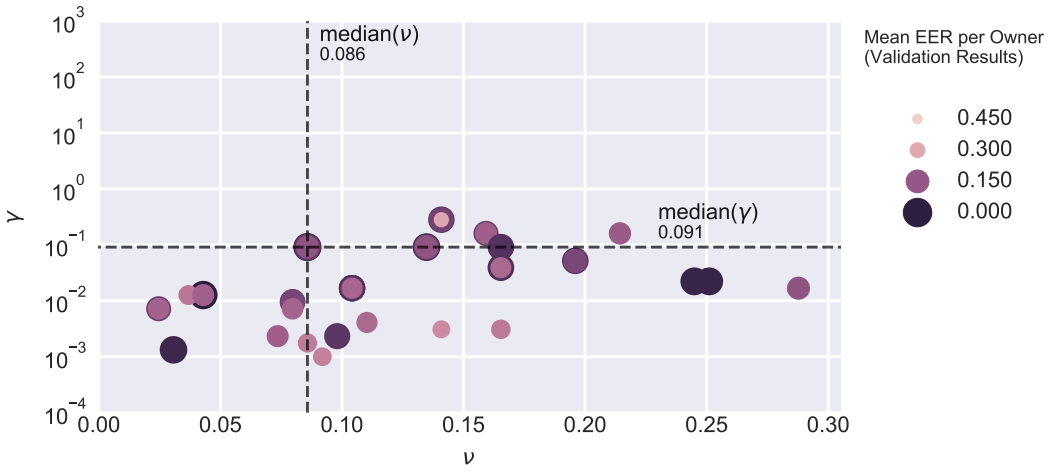


Fig. 5.8.: Schema of data splitting for training and testing. The six different session types are denoted A–F. 18 sessions from one subject are used for training, 6 additional sessions from the training subject are tested along with six sessions from one other subject. Every subject is once selected as owner and tested against all remaining subjects.

sets: A validation set, containing 60 subjects used for hyperparameter tuning, and a testing set, containing 30 subjects for model evaluation. These sets are then split into training sessions and validation/testing sessions. Then one subject is selected as "owner", and samples from his training sessions are used for model training. Afterward, pairs of owner's and impostor's samples from the validation/testing sessions are selected for prediction, as depicted in Figure 5.8. Every subject is once denoted as the owner, which leads to training and validation of 60 models during optimization. To gain more robust results, I repeated the split into training and validation/testing sessions 5 times using different seeds for the random operations. The hyperparameter optimization was implemented as a random search, testing 80 combinations of γ and ν per run. After pretesting with a subset of the data and a broad parameter range, the range for the random search was narrowed down and set to the region of interest of 0.001 to 1000 for γ and 0.0001 to 0.3 for ν.

It is important to mention that the hyperparameters were optimized only once based on the results of all subjects in the validation set. Optimizing the parameters for every individual subject as an owner would be desirable and would undoubtedly produce better results, but requires significantly more effort. To choose the values of γ and ν, which are later used for the models of all subjects during testing, I calculated the medians of the best combinations per owner found during the validation step of the random parameter search (Figure 5.9).

Fig. 5.9.: Best combinations of the parameters γ and ν found per owner during the random search, using EER as scoring metric. The size and hue indicate the mean EER across all attack scenarios per owner.



Centeno et al. (2018, p. 5) limited the **amount of training data per user** to 6750 observations or 67.5 seconds. It is not clearly stated, but I expect this applies to both their OCSVM baseline model and their primary Siamese CNN model for comparable results. With a window length of 0.5 seconds, this results in 135 training samples per user. An equal number of samples is selected from every user's session; in the case of 18 sessions, these are 8 samples per session rounded up. It is not mentioned, how many samples were used for testing, so I decided to use the same number of 8 samples per session as used for training. It remains unclear why the number of 6750 observations was chosen as training set size. Therefore, I covered this aspect during my evaluation of the training delay (see Section 5.6).

Due to the already mentioned lack of clarity regarding the **normalization** (see Section 5.3), I decided to test two different normalization approaches at this point. First, I implemented a version, in which the data is initially normalized feature-wise per subject using the min-max algorithm, before splitting into training and testing sets. I refer to this variant as the "naive approach" because it introduces bias into the testing data and is not applicable in a realistic scenario, because during the authentication phase data from the owner and a potential impostor cannot be distinguished upfront and therefore must be normalized using the same parameters. Second, I implemented a variant which I named "valid approach", where I split the data into training and testing set upfront, fit the normalization scaler using only the owner's training samples, and apply the same fitted scaler to the testing samples for both owner and impostor.

The **evaluation of the model performance** was split into three parts, one for each evaluation criterion (see Section 4.4). To assess the authentication performance, CV was performed as described above on the 30 subjects of the testing set and the EERs

for all owner–impostor scenarios were computed. This step was performed 5 times using different seeds for the random operations to gain more robust results.

The same CV was used to evaluate the **training delay**, where the CV was repeated 14 times with an increasing number of samples for model training. The tested training set sizes range from 1 and 750 samples or 0.5 and 350 seconds for the proposed window length.

The **detection delay** can hardly be assessed directly, as it would require labeled sensor recordings of scenarios, where impostors took over the phone of the owner during the same recording session. Injecting impostor data from another session into the owner's session and detecting this injection would be a possible approximation, but I rejected that idea, because it seems to be far from a realistic scenario, where the sensor values usually do not change abruptly between one sensor reading and the next one, as they might do in this testing scenario.[28] Instead, I decided to take the number of samples used to reach a certain confidence interval of EER as a proxy for the detection delay. Following Illowsky and Dean (2013, pp. 445–447) the error bound for a population mean EBM can be estimated based on the standard deviation σ, the number of samples n and the z-score z for $\frac{\alpha}{2}$, where α can be obtained with the chosen confidence level CL through $CL = 1 - \alpha$:

$$EBM = \left(z_{\frac{\alpha}{2}}\right) \cdot \left(\frac{\sigma}{\sqrt{n}}\right) \tag{5.1}$$

As we want to derive the number of samples n for a certain margin of error EBM, we can transform the formula to:

$$n = \left\lceil \left(\frac{z_{\frac{\alpha}{2}} \cdot \sigma}{EBM}\right)^2 \right\rceil \tag{5.2}$$

Because it is impossible to evaluate fractions of a sample, the resulting value is rounded up $\lceil\ \rceil$ to the next integer value.

For my evaluation, I chose $CL = 95\%$ which results in the z-score $z_{\frac{1-CL}{2}} = z_{0.025} = 1.96$, and a maximum margin of error $EBM = 0.0025$. With this formula, we can estimate the number of samples n needed to state with 95% confidence, that the population's true mean EER is inside an interval of $\pm 2.5\%$ EER.

In the implementation, I tested single owner samples against single samples from all other subjects as impostors, to obtain EERs for 400 individual samples per owner, plotted the narrowing confidence interval and calculated n. As already mentioned, n is not the detection delay. It is a proxy that indicates, how many samples are needed

[28] E.g., the magnetic fields of the various sessions vary a lot, maybe due to different surroundings. Such a difference might be easy to detect by the model but is unlikely to happen in a realistic scenario.

until we can estimate the mean EER with a certain confidence, as one property of detection delay.

It is also important to mention, that this calculation provides only a rough approximation: For the specific use case of biometrical authentication, where an owner's sample is tested against impostors' samples pairwise, Schuckers (2009) propose a formula, that also takes the intra-comparison pair correlation and error rate into account for calculating the confidence interval. I kept this more elaborated approach out of this thesis' scope, but recommend its use in future research undertakings.

5.5 Modeling Siamese CNN

The implemented process of this approach is divided into two phases (Figure 5.10). In the first phase, a Siamese CNN (see Section 4.5.2) is trained, in the second phase, the trained CNN is used to generate deep features that are then used to train an OCSVM as authentication model, just like described in the previous section 5.4. The implementation follows the parameters used by Centeno et al. (2018), which are summarized in Table A.4. As not all necessary information is available, and some details remain ambiguous, I again had to make some educated assumptions.

The most important open question of the OCSVM baseline model (see Section 5.4) also applies to the Siamese CNN approach: How was the subject wise min-max **normalization** performed? Exactly as with the OCSVM, I decided to test a naive approach with normalization before splitting into test and training sets, and a more realistic approach, where normalization of the owners training set is repeated on the whole testing set using the parameters fitted during training.

The **data splitting**, as described by Centeno et al. (2018, pp. 4f), is performed in such a way, that 60 users are used to train the Siamese CNN, while the remaining 30 users are used to train and test the final OCSVM authentication model. However, it remains unclear, how the validation and testing of the Siamese CNN were performed, and if and how a validation set was used for the final authentication model. Centeno et al. (2018, p. 4) state to sample data from 18 sessions[29] of every 60 users for training the Siamese CNN, which could lead to the assumption, that the remaining 6 sessions per user were used for testing/validation. But such a splitting would not provide valid results: If validation is performed with sessions from the subjects, whose data was also used for training, the results would be biased. Regarding the

[29] The statement "From each user, we sample data from 18 different sessions, 6 sessions for each task-motion condition activities (reading-sitting, reading-walking, writing-sitting, writing-walking, navigating-sitting, navigating-walking)." (Centeno et al., 2018, p. 4) is assumed to contain a typo. It would lead to 36 sessions.

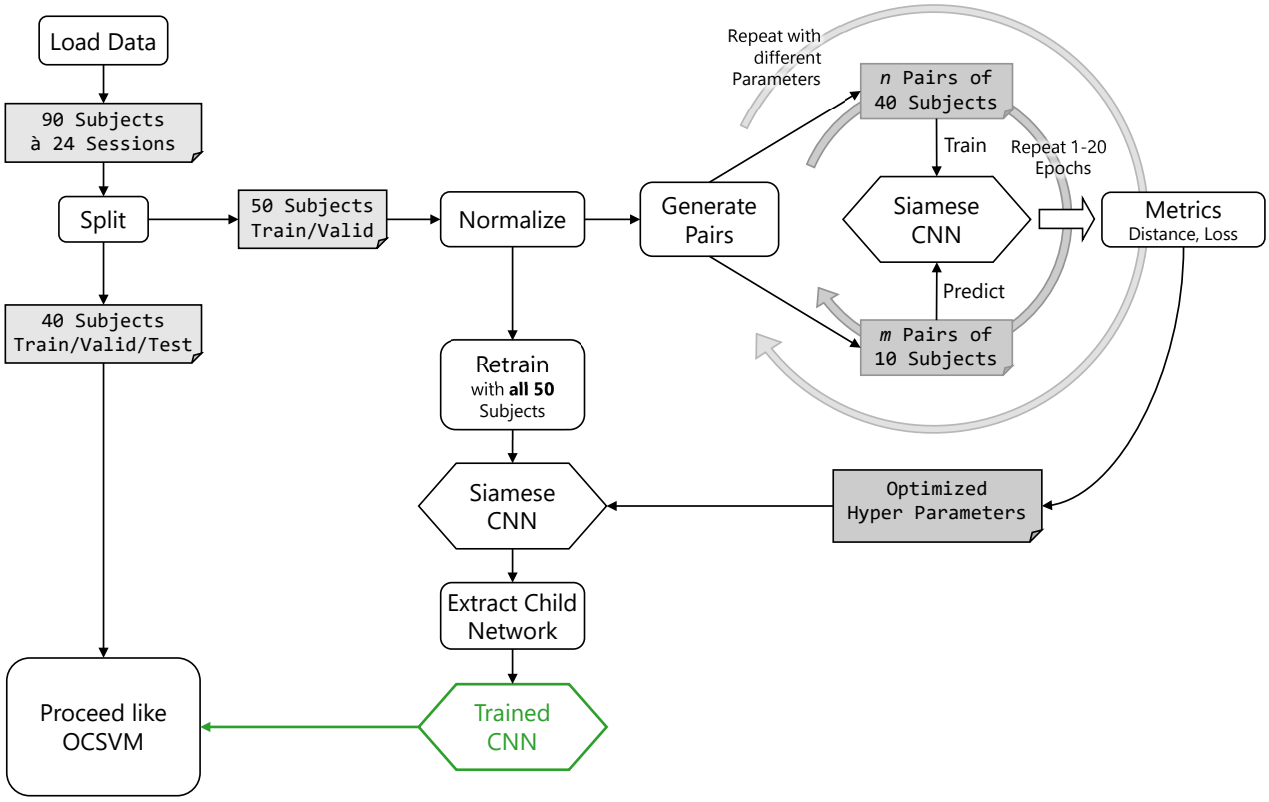


Fig. 5.10.: Implemented process for the Siamese CNN approach. A subset of 50 subjects are used to train and validate the Siamese CNN (middle and left), the remaining 40 subjects are used to perform the OCSVM process as described in Figure 5.7, with the single difference, that now the trained CNN (green) is used to transform the raw samples into deep features, before feeding them into the OCSVM.

final OCSVM authentication model, it is not mentioned that a validation set was used for tuning its hyperparameters.

In my implementation, I decided to introduce specific **validation sets** to be able to optimize the hyperparameters without introducing bias into the model. The downside is that I had less data for training the Siamese Network than proposed by Centeno et al. (2018): Data from 10 subjects are used as a validation set for the OCSVM. The final training and testing of the authentication model should be done with the same number of 30 subjects to stay comparable with Centeno et al. (2018). Consequently, 50 instead of 60 subjects remained for training the Siamese CNN. For this deep learning step, I decided to compromise between a bias-free model and the size of the training data. After using data from 40 subjects for training and 10 subjects for validation during the hyperparameter optimization, I retrained the network with data from all 50 subjects under the assumption, that the benefit of additional data has a more significant impact than the bias introduced in the Siamese CNN. The final training and testing of the authentication using the OCSVM are not influenced by this bias anyway, as it is performed with the 30 subjects unknown to both models.

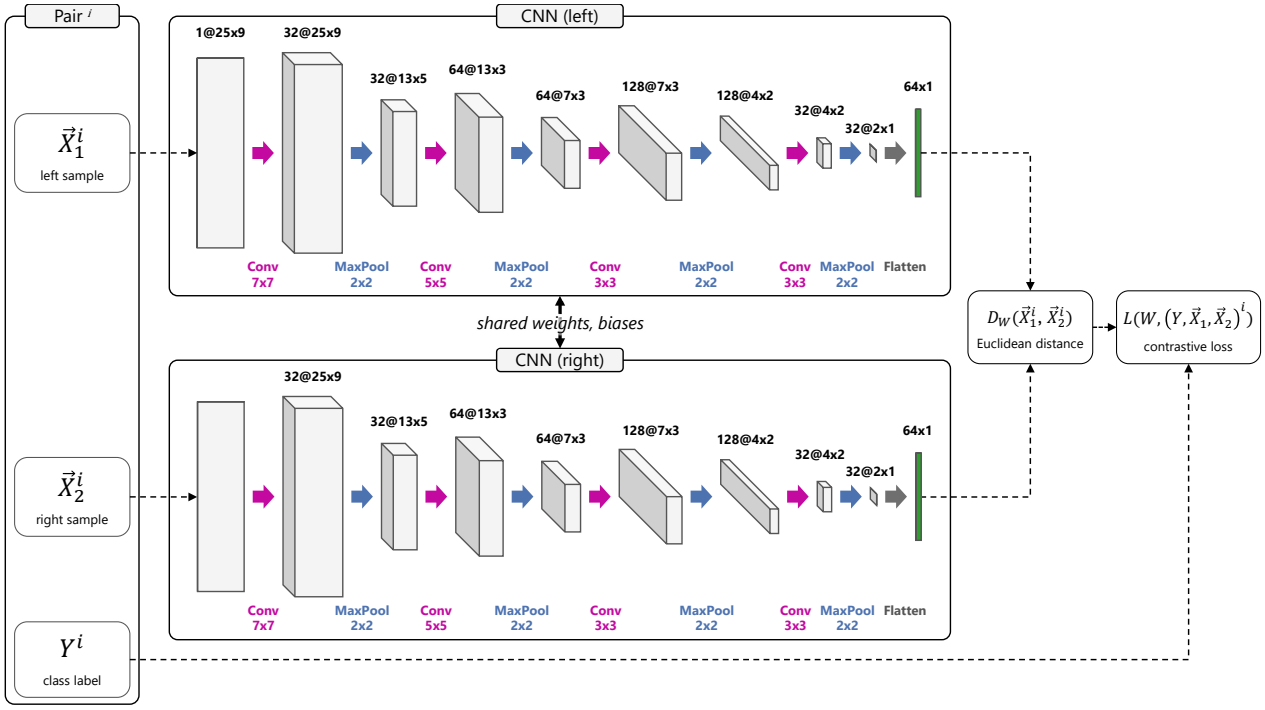


Fig. 5.11.: Architecture of the Siamese CNN, modeled based on the available information by Centeno et al. (2018). All filters use padding. The vector of the last CNN layer (green) is taken as deep feature.

The reimplementation of the **Siamese CNN architecture** (Figure 5.11) was a challenge, as it is not described consistently: Centeno et al. (2018, p. 5) state, that a 2D vector of size $features \times window\ size$ is provided as input for the first of four 2D convolutional layers. However, 2D convolutional layers need 3D input data. Therefore, I assume, that the input of "a 2D block of motion data [...]" is meant to be a "flat" 3D vector of size $features \times window\ size \times 1$. In this case, the filter would be moved along the time axis and the sensor axis of the input vector. This is possible, but only, when using padding for both convolutional and pooling layers, which is not mentioned in the original paper. Otherwise, the given filter sizes would cause negative dimension size in the network. Also, the ordering of the features, the axes of the three sensors, might influence the results of this setup, but information about the order of the feature vector is not provided. Nevertheless, I implemented that architecture at this stage (and propose an alternative architecture in Section 5.7). Another important parameter of the CNNs architecture not mentioned by Centeno et al. (2018) is, which activation functions were used for the convolutional layers. Without having time to go deeper into architecture optimization, I decided to use the common activation function Rectified Linear Unit (ReLU), which is known to have several advantages in CNNs (Nair and E. Hinton, 2010, pp. 2f).

The distinctive aspect of a Siamese Network is its **contrastive loss function**, which was introduced and explained in detail by Hadsell et al. (2006, pp. 2–4): Consider a set of training vectors $I = \{\vec{X}_1, \cdots, \vec{X}_p\}$. In our case, the Siamese network consists

of two CNNs with shared weights W. The input I is provided during the training phase to the Siamese network in pairs, e.g., input vectors $\vec{X_1}$ for one of the child CNNs and $\vec{X_2}$ for the other. The class label Y of the pair is binary, $Y = 0$ if $\vec{X_1}$ and $\vec{X_2}$ is a so-called positive pair where both vectors come from the same user, and $Y = 1$ if $\vec{X_1}$ and $\vec{X_2}$ is a negative pair with vectors produced by two different users. During the training, the CNNs learns a single[30] parametric function G_W to transform the input vectors into output vectors $G_W(\vec{X_1})$ and $G_W(\vec{X_2})$. A parameterized distance function D_W is learned as the Euclidean distance between the output vectors:

$$D_W(\vec{X_1}, \vec{X_2}) = \left|\left|G_W(\vec{X_1}) - G_W(\vec{X_2})\right|\right| = \sqrt{\sum_{i=1}^{n} (G_W(\vec{X_1})_i - G_W(\vec{X_2})_i)^2} \quad (5.3)$$

If we shorten the notation of $D_W(\vec{X_1}, \vec{X_2})$ to D_W^i, then the contrastive loss function L, which depends on trained weights W and the i-th input pair $(Y, \vec{X_1}, \vec{X_2})^i$, is defined as:

$$L(W, (Y, \vec{X_1}, \vec{X_2})^i) = (1 - Y^i) \cdot \frac{1}{2} \cdot (D_W^i)^2 + Y^i \cdot \frac{1}{2} \cdot \{max(0, m - D_W^i)\}^2 \quad (5.4)$$

where $m > 0$ is the so-called margin, which limits the contribution of the negative pairs to the loss. This leads to the following behavior: The function leads to a low loss value, if the input pair has a positive class label ($Y = 0$) and the distance between its vectors $D_W(\vec{X_1}, \vec{X_2})$ is small, or if the pair is negative ($Y = 1$) and the distance is high. Margin m can be used to tune the influence of the negative pairs. Unfortunately, Centeno et al. (2018) do not provide the value they used for m or any indication of how they obtained it. Therefore, I consider the margin as another hyperparameter to optimize.

It might be useful to mention, that the **implementation of the contrastive loss function** in Keras differences slightly from the formula given above:

```
K.mean(Y * K.square(D) + (1 - Y) * K.square(K.maximum(0, m - D)))
```

which is equivalent to:

$$L(W, (Y, \vec{X_1}, \vec{X_2})^i) = Y^i \cdot (D_W^i)^2 + (1 - Y^i) \cdot \{max(0, m - D_W^i)\}^2 \quad (5.5)$$

This is because in Keras it is more convenient to use $Y = 1$ for the positive pairs, and $Y = 0$ for the negative pairs (inverse to Hadsell et al., 2006, p. 3), as complies better with the default metric functions. Additionally, factor $\frac{1}{2}$ is removed, as it does not change the optimization results of the network. (GitHub.com, 2016)

[30] Because of shared weights and biases, the two networks are technically identical.

As described in the concept (see Section 4.5.2), the Siamese CNN is trained with pairs of input vectors: positive pairs, were both input vectors are from the same user, and negative pairs, with input vectors from different users. For **generating the pairs**, which is done repeatedly on the fly for CV, I used an approach optimized for computational speed. I decided to split all samples subject wise into halves, each half containing the same number of samples for the same user. The first half is used to generate the positive pairs: The samples of every subject are shuffled, divided into halves, and aligned as the left and right side of the positive pairs. The second half is shuffled across all subject, split into halves, and aligned as the left and right sides of the negative pairs. This approach is quite fast but has the downside that there is a certain probability, that "accidentally" some positive pairs are created within the negative pairs. The classes are balanced as a last step of the pair generations, but the distribution of samples across the subjects remain slightly unbalanced (see Figure 5.12) as a compromise for computational efficiency.

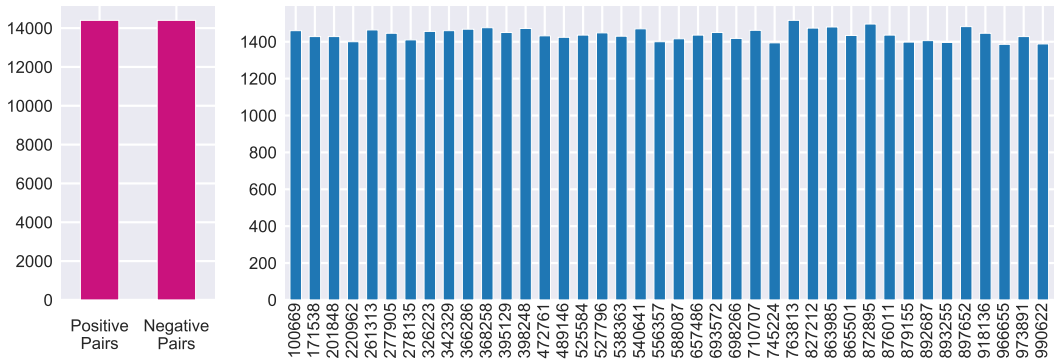


Fig. 5.12.: Training Pairs: The two classes of pairs are balanced (magenta), but the distributed of subjects among them are slightly imbalanced (blue).

The training process of the Siamese CNN can be visualized nicely by plotting histograms of the Euclidean distances for both positive and negative pairs (Figure 5.13). Before training, both distributions are quite similar. During the training, when the network learns to distinguish between both classes, the distributions are getting more and more separated. The distances of the positive pairs are drawn towards zero, while the distances of the negative pairs approach the margin. Of course, the network learns this separation more precise for the training data. By looking at the evolution of the loss for both training and validation sets (Figure 5.14), it is possible to determine the epoch in which the loss of the validation data becomes stable. Then the model is retrained to the chosen epoch and stored for further usage.

After the Siamese CNN is trained with those pairs, the model is prepared for generating the deep features. This is done by dropping the last two layers, that were used to

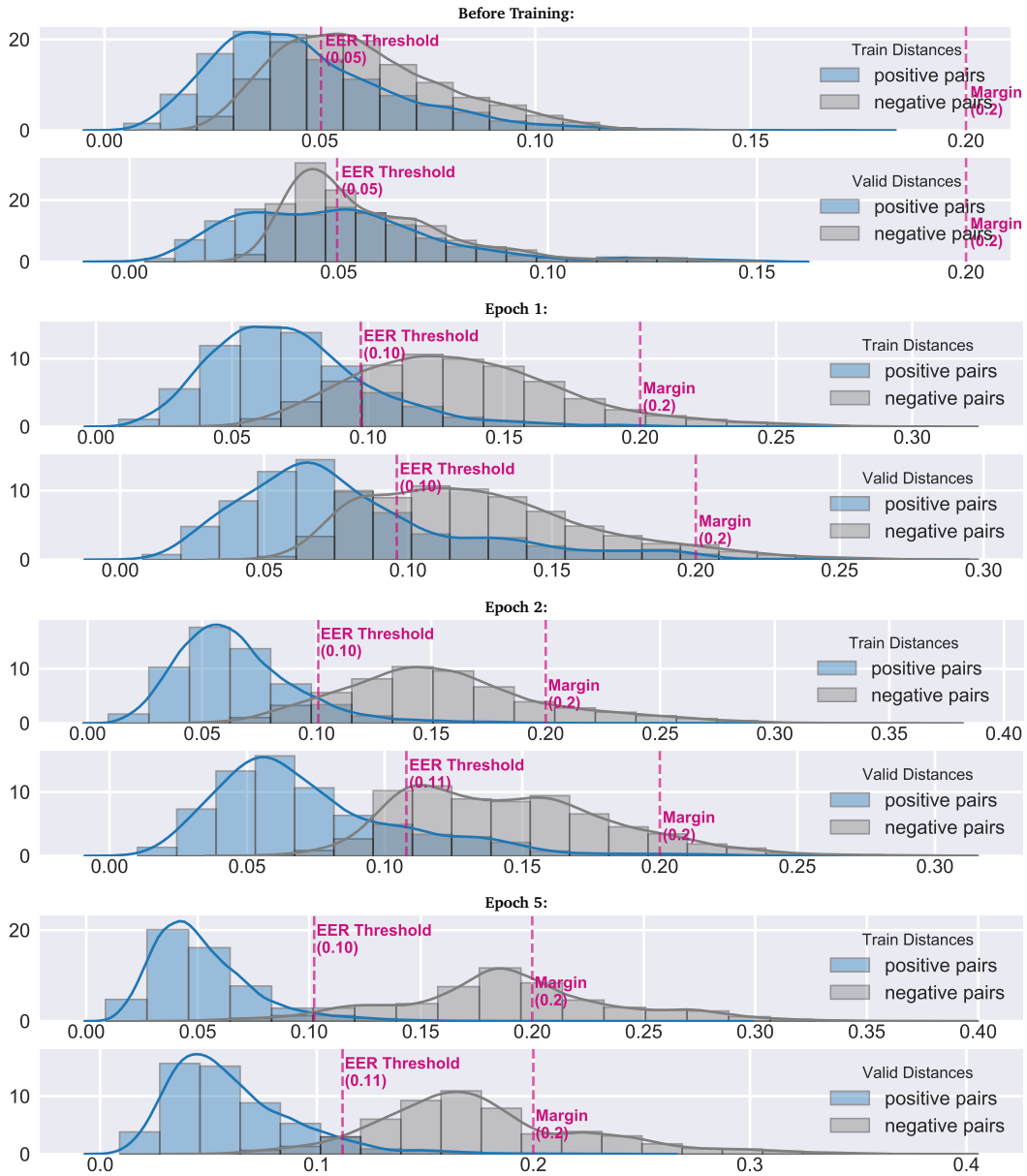


Fig. 5.13.: Distribution of Euclidean distances between positive and negative pairs during training, for training (uneven rows) and validation set (even rows).

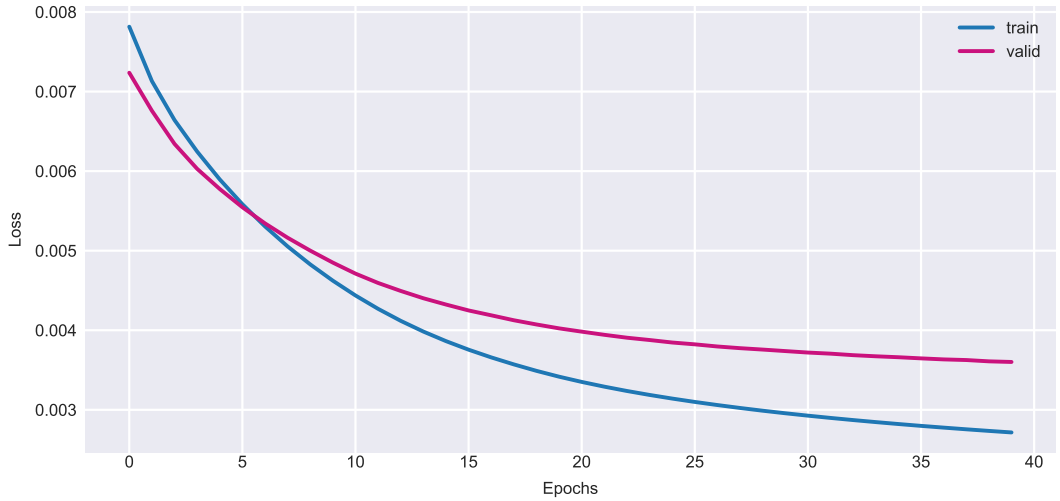


Fig. 5.14.: Loss of training and testing sets during the epochs of training.

compute the Euclidean distance and the contrastive loss during training, and using the last hidden layer of one of the CNNs as output layer instead.[31]

For the implementation of the **OCSVM authentication model**, I struggled with the same missing information on hyperparameters as with the baseline model (see Section 5.4). Again, I proceeded with a random parameter search. The training, validation, and testing of the OCSVM authentication model in the Siamese CNN approach is implemented identical to the baseline model, except for the CNN, which gets injected to generate the deep features during the cross-validation (see Figure 5.7, green). Instead of feeding the samples with the given window length directly into the OCSVM, they are routed into the CNN, and the resulting deep features are used as the OCSVM's input.

All assumptions and decisions made during the reimplementation are summarized in Table A.4, along with parameters that were derived from Centeno et al. (2018).

5.6 Evaluation Results

In this section, I will present the results of testing both suggested approaches, the OCSVM baseline model, and the ensemble model consisting of a Siamese CNN for feature learning and an OCSVM as authentication classifier. Both approaches have been evaluated twice with different normalization strategies: the "naive approach", were the normalization was performed initially, on all data, and individually for every subject, and a "valid approach", where the normalization scaler was fitted on the training data of the owner only and then applied to transform the testing data for both owner and imposter.

Both models were tested using data of 30 subjects. No data of those subjects was involved during the hyperparameter optimization of the models. The testing process itself includes a CV, in which 1 of the 30 subjects is declared as "owner", and the model is trained with data of 18 sessions of this subject. Data from 6 different sessions (1 session for each of the 6 task-scenario-combinations) of this owner were tested against data from 6 sessions of another subject as "impostor" in 1 versus 1 scenario. The number of testing samples was equal for both the owner and impostor. As this test was repeated with every remaining subject as an impostor, 29 tests were performed per owner. Also, each of the 30 subjects once served as owner, leading to 870 test repetitions.

[31] Unfortunately, I was not able to adjust the input layer in Keras correctly: The network still needs a pair of vectors as inputs. This might affect the speed of the prediction, but not the results.

Unlike Centeno et al. (2018), I repeated this whole process 5 times using different seeds for the random operations during sample selection. The reason for this procedure, which concludes in a total of 4.350 tests, is the small number of 8 random testing samples per subject according to the original study. The additional repetitions are supposed to increase the robustness of the results.

In the following, I first describe the results regarding all three criteria and discuss their implication afterward.

5.6.1 Authentication Reliability

As explained in the concept (see Section 4.4.1), the Equal Error Rate (EER) is the preferred metric regarding authentication reliability for this thesis. Additionally, the accuracy was calculated as another standard metric for classification: It was necessary for comparison with the original study, where it was one of the metrics reported by Centeno et al. (2018, p. 5). While the charts for visualizing the EER are shown in this section, the detailed charts regarding accuracy can be found in Appendix A.5.

The boxplot in Figure 5.15 shows the classification results for the "naive approach" of the OCSVM baseline model. The data points compromise of all 145 tests (29x5) per owner. Additionally, the mean EER 18.8% for all tests is depicted. The differences among the different owners are quite interesting, spread and mean vary quite a lot. The calculated mean accuracy was 86.3% (see Appendix A.5, Figure A.1).

Repeating the same test, but using a "valid approach" with normalization performed closer to a real-world setting revealed very different results. Figure 5.16 visualizes significant weaker classification performance. Only a single owner's mean EER reached the overall mean of the "naive approach". It is also remarkable that the owners' individual results do not seem to correlate between the two approaches. E.g., 862949, which performed extraordinary well in the "naive approach", is no longer conspicuous in the "valid approach", performing worse than average. The charts also indicate that fewer outliers are present in the "valid approach" compared to the "naive approach". The mean EER of the "valid approach" is 36.9%, while the mean accuracy is 65.4%.

The differences between the two normalization approaches are even larger when the Siamese CNN was used for generating the deep features as the OCSVM's input: when using the "naive approach", the deep features as input for the authentication model performed with an EER of 14.7% much better than the raw features (Figure 5.17). When normalized with the "valid approach" on the other hand, the Siamese CNN

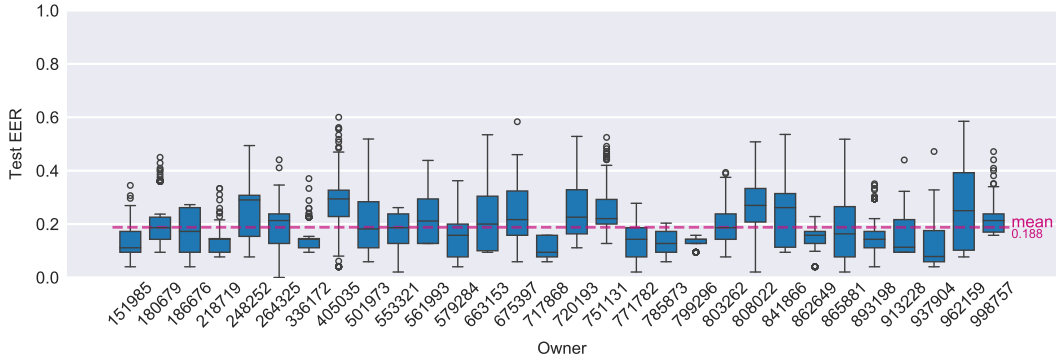


Fig. 5.15.: EERs of samples per owner – OCSVM (naive approach).

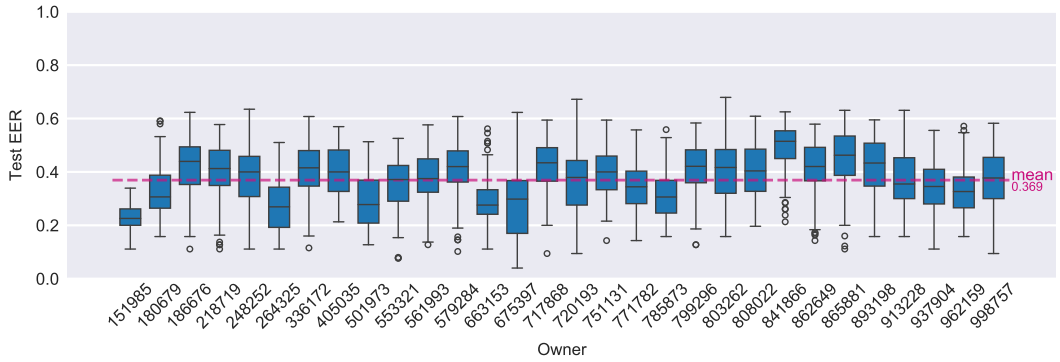


Fig. 5.16.: EERs of samples per owner – OCSVM (valid approach).

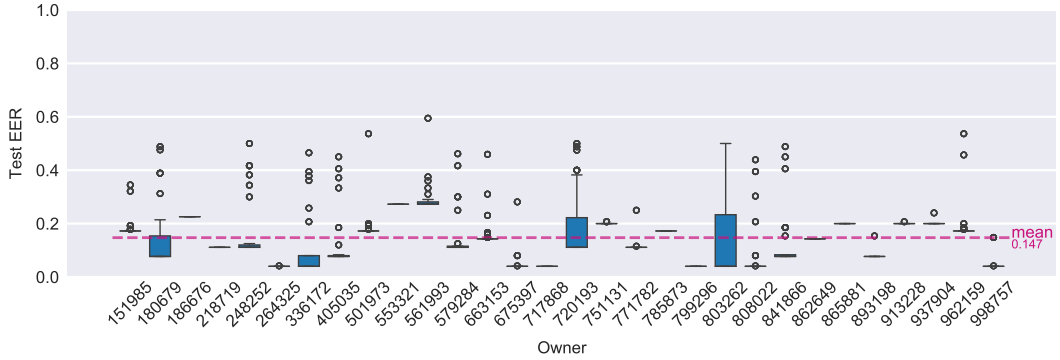


Fig. 5.17.: EERs of samples per owner – Siamese CNN (naive approach).

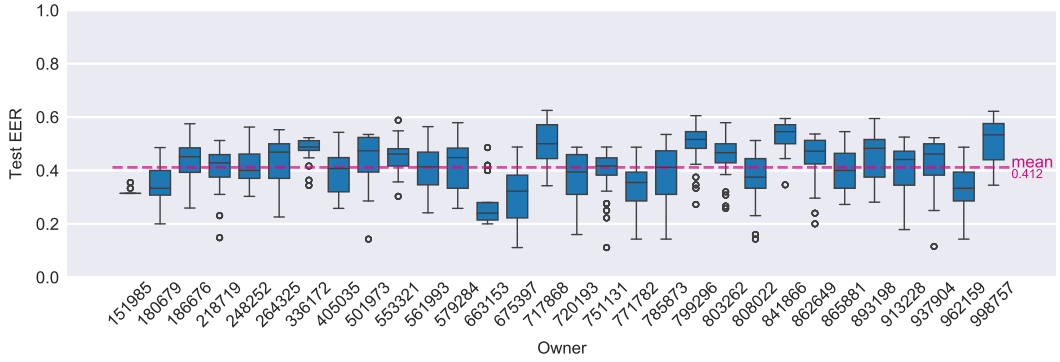
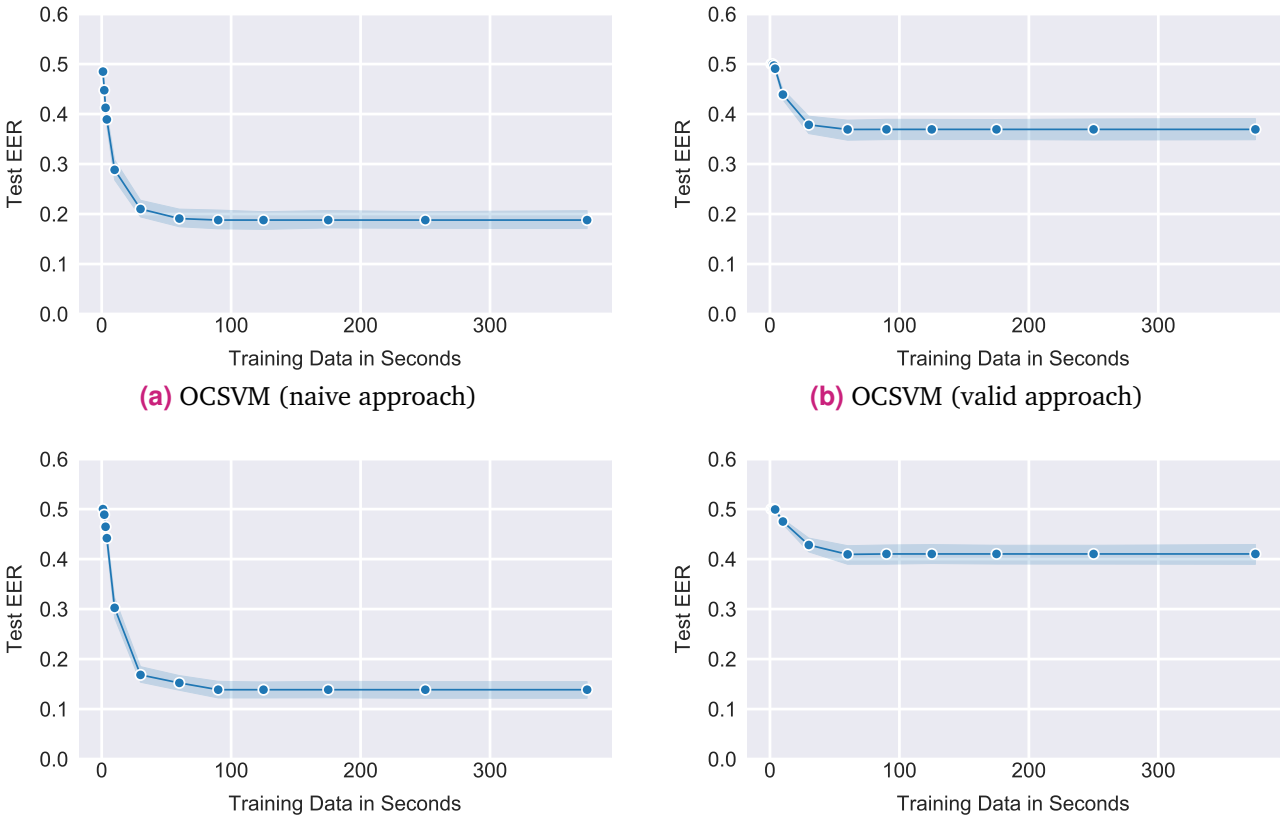


Fig. 5.18.: EERs of samples per owner – Siamese CNN (valid approach).



(a) OCSVM (naive approach)

(b) OCSVM (valid approach)

(c) Siamese CNN (naive approach)

(d) Siamese CNN (valid approach)

Fig. 5.19.: Training delay: mean EER and 95% confidence interval for different lengths of training data for authentication model.

based model performed with an EER of 41.2% worse than the baseline model (Figure 5.18). Again, there does not seem to be a correlation between the classification performance of the individual owners between the normalization approaches.

5.6.2 Training Delay

The training delay was evaluated by repeating the same steps used to evaluate the authentication reliability, but with an increasing number of training samples. 12 different training set sizes (2, 4, 6, 8, 20, 60, 120, 180, 250, 350, 500 and 750 samples) were tested. For the baseline model, with its proposed sampling rate of 100 Hz and window length of 50 measurements, this corresponds to time spans between 0.5 and 375 seconds. The same time span was covered during the evaluation of the Siamese CNN approach with its 25 Hz sampling rate and window length of 25 (1, 2, 3, 4, 10, 30, 60, 90, 125, 175, 250 and 375 samples).

If the classification results for the different training set sizes are plotted (Figure 5.19), It becomes visible, that the EER is stable above ~100 seconds training data, regardless of the used approach. The OCSVM “valid approach” gets stable earlier, with

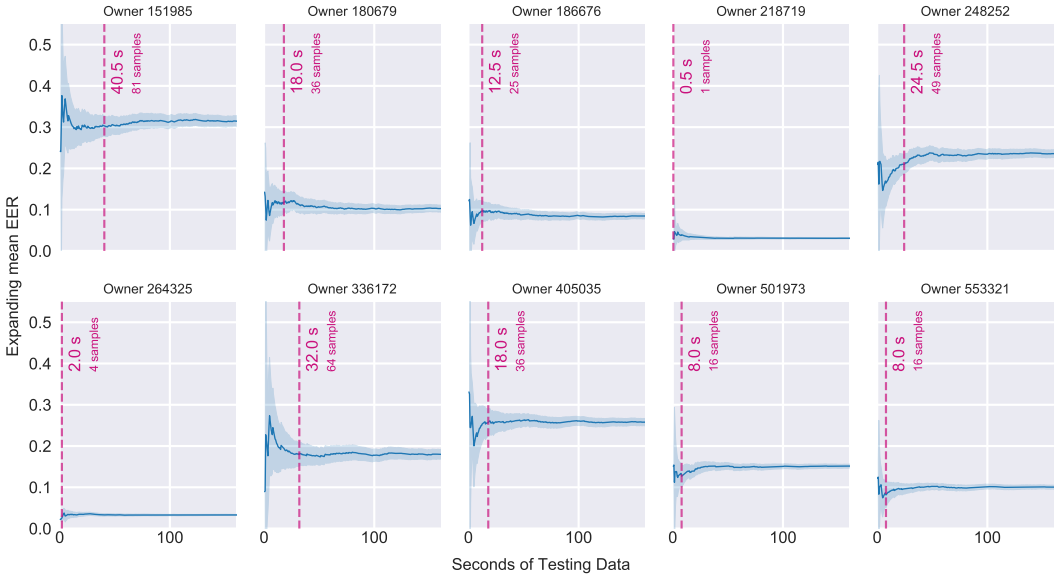


Fig. 5.20.: Confidence Intervals – OCSVM (naive approach).

~50 seconds. This might explain the choice of 6750 measurements as training set size by Centeno et al. (2018, p. 5): The training set length of 67.5 seconds is a good compromise, with a performance very close to the optimum with relatively small training data. s

5.6.3 Detection Delay

First, samples of the owner were tested one by one against a single sample from every other subject as an impostor, and the corresponding EER was calculated. The variance in the EER was visualized by plotting its expanding mean and expanding confidence interval of confidence level 0.95 (Figure 5.20; for other approaches see Appendix A.6) for the first 10 owners. Additionally, I calculated the number of samples needed to estimate the mean EER with a confidence interval of 0.05 EER width and confidence level 0.95. The differences among the owners are again interesting: while the confidence interval unsurprisingly seems to correlate positively with the mean EER, we can also see with owner 405035, that also subject with below average classification score can have a quite narrow confidence interval. 16.4 seconds of testing data were needed on average for the OCSVM "naive approach" to reach the determined confidence threshold.

5.6.4 Interpretation

The results of the three evaluation criteria presented in the previous sections raise questions, which need to be discussed. At this stage, the variations among the tested approaches regarding the authentication reliability are certainly the most important

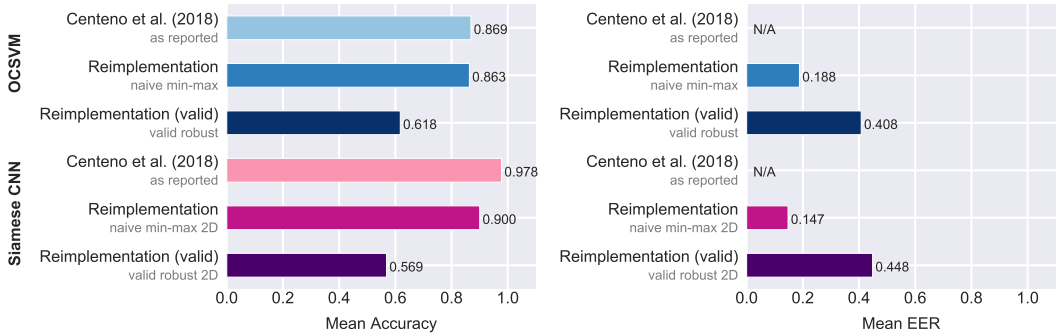


Fig. 5.21.: Classification results – mean of all tested owners.

topics. The classification performance of the models strongly influences the two other criteria, and therefore, are considered to be secondary.

When comparing the metrics of tested variants with the reported accuracy of the original study by Centeno et al. (2018, p. 5), differences are apparent (Figure 5.21). Due to a lack of details regarding the implementation of Centeno et al. (2018), it is impossible to say, if they have chosen a normalization variant similar to the one I introduced as a "naive approach" in this thesis, or if there are other reasons for their promising results. Either I succeeded to reproduce the result of the original study within the "naive approach", which I consider to be not valid, or I failed to reproduce the results, e.g., due to missing information. Further information is needed to resolve this ambiguity.[32]

What I can tell is that the results of the OCSVM baseline model using the "naive approach" can be considered as on par with the original study. The accuracy of my implementation of the Siamese CNN approach is 7% lower, but considering the more complex setup with more assumptions regarding the implementation, fewer number of training data caused by my additional validation split and the unsolved question, how exactly the accuracy was obtained in the original study, I judge this difference also as minor.

More interesting to me is the question: Why did the normalization according to the "valid approach" performed significantly worse than the "naive approach"? The accuracy of the "valid approach" is close to random.

To answer this question, I explored the data set regarding normalization in an extra Jupyter Notebook.[33] Figure 5.22 shows the distributions of the features for 5 random subjects. For the individual sensors, the upper and lower quartiles of the distributions are similar between the subjects. The z-axis of the accelerometer is drawn towards

[32] According to the author of the original study, a publication with comprehensive details is planned to be published later this year.
[33] Source code: `/notebooks/chapter-5-6-4-explore-normalization.ipynb`

10 by the gravity component, the median for the y-axis is lower and lowest for the x-axis, with a median around zero. The values of all three axes of the gyroscope are grouped close to zero. The values for the magnetometer vary more between different subjects. For 4 of the 5 subjects, the z-axis has the lowest median, the x-axis is in the middle, and the y-axis axis has the highest median. However, the absolute values vary a lot, and there is also one subject (orange) with a different distribution. Together with the distribution of the overall magnetometer values (see Figure 5.3), this supports the assumption, that the surrounding might have a higher impact on this sensor than the movements of the smartphone produced by its user.

But it is more important to investigate the outliers of the distributions, because the maximum and minimum of a distribution has a huge effect on the normalization using the min-max algorithm, which linearly transforms all values into a provided range, e.g., 0 to 1, with the minimum value being transformed to 0 and the maximum value to 1, which makes it sensitive to outliers. Moreover, we have a lot of those in the data. I plotted the data of the same 5 subjects (Figure 5.3) after normalizing them with the same min-max implementation used for the "naive approach" to demonstrate this effect (Figure 5.23). The formerly similarly distributed values of gyroscope and accelerometer have been shifted under the influence of their outliers: median, lower and upper quartiles are now easily distinguishable between the subjects, and also the distributions of the magnetometer values shifted.

This can explain the observed difference between the results of the "naive approach" and the "valid approach": It is very likely, that both models learned the differences in the features' distributions per subject, instead of the granular patterns of individual samples. This hypothesis is hard to proof, especially in the Siamese CNN approach. However, it is possible to look for further indications: If the hypothesis is correct, then both models should perform worse, when a normalization method is used, that is less sensitive to outliers, e.g., the standard scaler or even better the robust scaler, which ignores values below the 1st and above the 3rd quartile during fitting. The performance should be significantly weaker for both "naive approach" and "valid approach". Exactly this happened (Figure 5.24): The EER of the OCSVM approaches increased to 50.0% (random-like) for the "naive approach" and 40.8% for the "valid approach". For the Siamese CNN, the EER of the "naive approach" increase to 48.1% and to 44.8% for the valid approach, when the robust scaler was used. The histograms of the pairs distances during training (see Appendix A.7, Figure A.8) reveal, that the network is unable to learn a transformation for differentiating the pairs, the loss is around 3–5 times higher, and quite stable for the training set (see Figure A.9). Surprisingly, the loss for the validation set is decreasing, but looking closely at the histograms, it becomes visible, that is due to a tiny number of negative pairs, which get extremely well transformed with a resulting distance around 0.7.

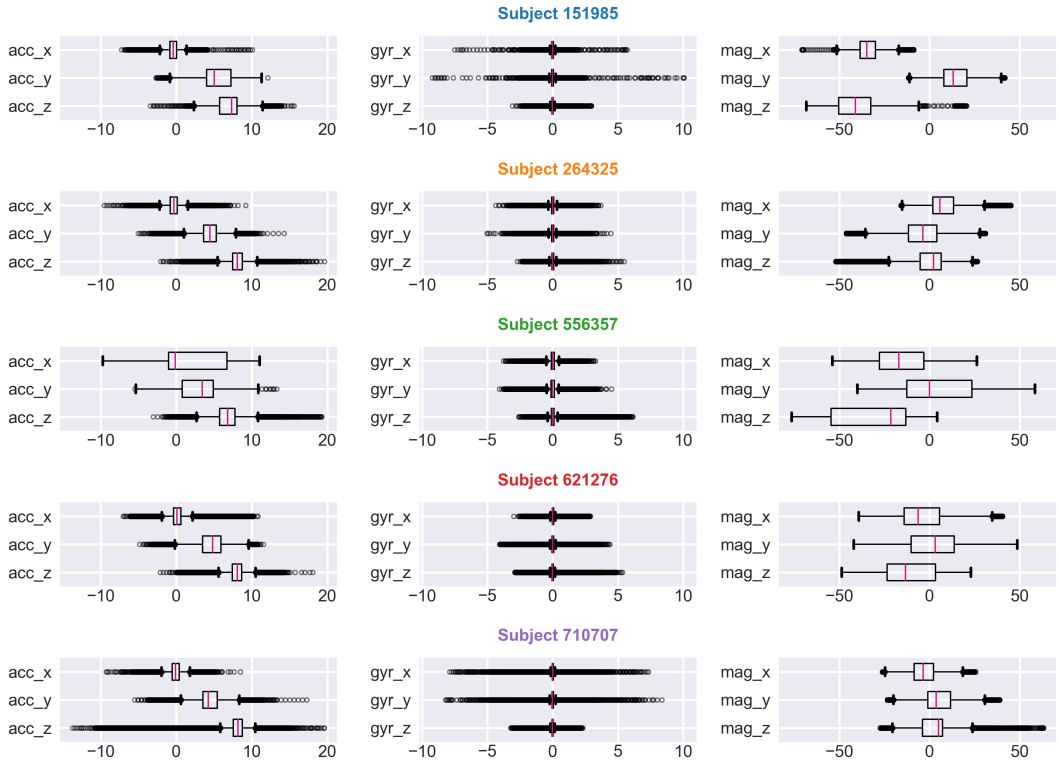


Fig. 5.22.: Distribution of raw sensor data for random subjects.

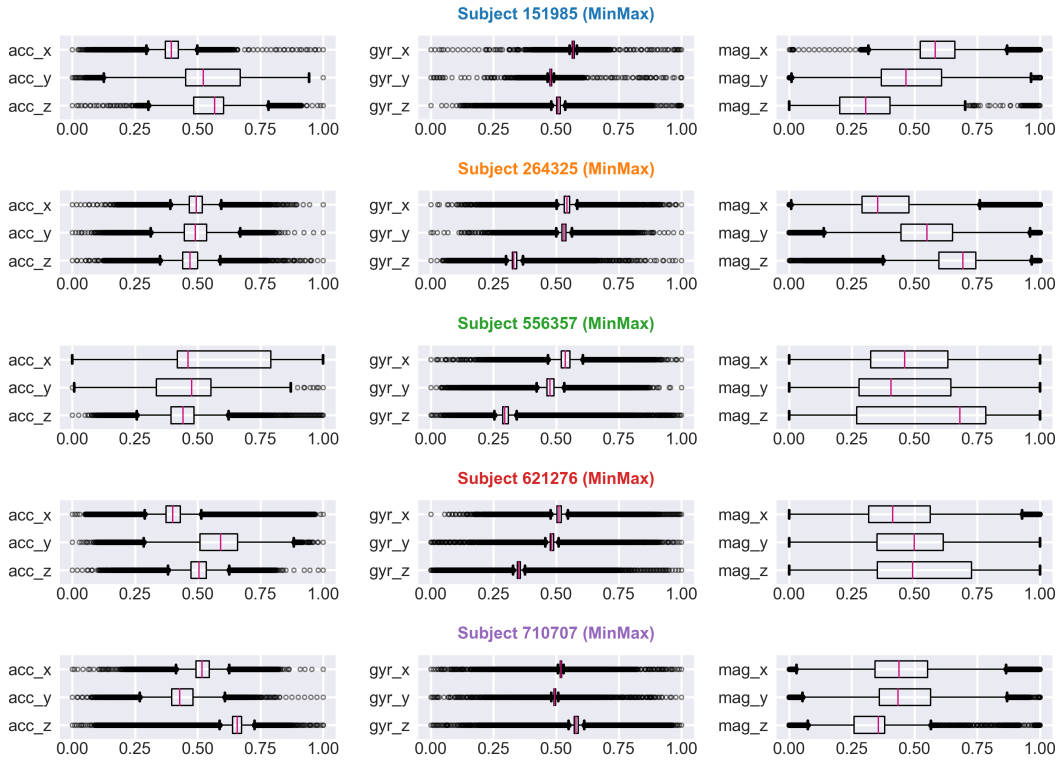


Fig. 5.23.: Distribution of sensor data for random subjects after min-max normalization.

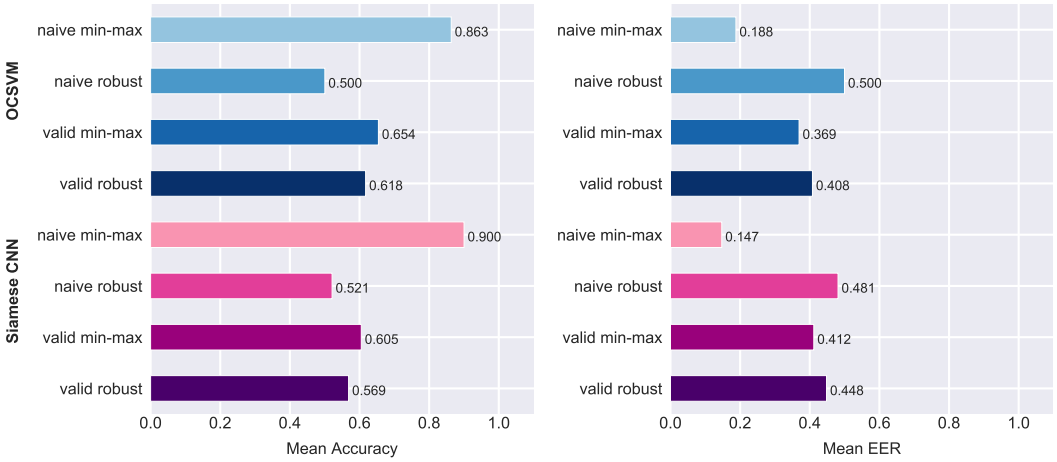


Fig. 5.24.: Classification results – original scaler (min-max) vs. robust scaler.

However, it is vital to notice, that more extensive research would be needed in order to fully understand the causal relationships and rule out other possible causes, e.g., the non-linear standardization performed by the robust scaler could affect pattern detection in the inertial data. It is also surprising, that with robust normalization the "valid approach" performs better than the "naive approach" for both models, but the difference is quite small.

The following three conclusions can be drawn from the evaluation so far:

1. The excellent performance of the "naive approach" is likely caused by the bias introduced by upfront applying a subject-wise min-max normalization on data containing outliers.
2. The performance of both models drops significantly in a scenario closer to a real-world application, where the normalization scaler can be fitted only on the training data and is applied to the testing data of both owner and impostor ("valid approach").
3. Sensor values produced by the magnetometer might have a more negative than positive effect, as they probably introduce additional bias due to the dependency on the surroundings, and do not reveal much information about a smartphone's movement patterns.

However, does this mean, that the proposed ensemble model of Siamese CNN and OCSVM is the wrong approach to solving Continuous Authentication (CA) in a real-world scenario? Not necessarily. With the knowledge about those pitfalls, I tried to optimize various model parameters to improve the performance of the "valid approach". I selected the accuracy of the Siamese CNN using the "valid robust approach" as a new baseline for my tests, as this variant is considered to be close to a

real-world application and not influenced by bias introduced through normalization. Then multiple attempts were made to improve the model.

5.7 Improvement of Modeling

After I had identified the weaknesses in my reimplementation of the study by Centeno et al. (2018), I decided to search for variants performing better with normalization performed closer to a real-world application, and without exploiting random outliers in the data. The focus hereby lies on the Siamese CNN approach, as it is the more promising model. The various parameters and the testing results are described and interpreted in the following subsections.

5.7.1 Test Parameters

Due to missing resources to implement and perform an automated parameter search, I had to manually execute tests with a set of parameter changes assumed to be most beneficial. Table 5.1 shows the prioritized parameters I altered, along with the reasoning and the tested variants. Due to the given limitations, I left the testing of all combinations of parameters to future research, and limited myself to test the influence of the parameters sequentially like in a search tree: I started with the parameter assumed to be most influential, and used the best performing option as given for the next step. This process was repeated until all parameters were tested. It is important to mention, that due to the computational complexity, I assessed the parameters influence through the plots of the Siamese CNN's training evolution, and did not follow all subsequent steps of deep feature generation and OCSVM evaluation. Plots can be found for all tests in Appendix A.8, HTML exports of the corresponding notebooks are available in my repository[34].

The **CNN architecture**, as the first and most influential parameter, needs to be explained in more detail. The Siamese CNN architecture that I derived and from the incomplete information in the study by Centeno et al. (2018) has some aspects which seemed odd to me: A 3D input with 2D filters used for 2D time series data? Around 164000 parameters to learn with a comparably low number around 28000 training samples? This seems like an unusual setup for me. Therefore, I decided to test two other architecture.

The first alternative architecture is the 1D filter equivalent to the 2D filter architecture I derived from the original paper (Figure 5.25). Besides the filter dimensions, I only

[34]Reports of parameter search: `/reports/optimization/` (Büch, 2019).

Tab. 5.1.: Variations of parameters tested for Siamese CNN approach. The initial values of the "valid approch" variant selected as new baseline are emphasized (**bold**).

Prio	Parameter	Variations	Reasoning
1	CNN architecture	**Orig. CNN (2D filters)**, Orig. CNN (1D filters), FCN (1D filters)	In my opinion, the architecture with 2D filters seems not to be the best option. Also it is quite common to use a dense layer as last layer, instead of flattening the final pooling layer.
2	Features	**{acc, gyr, mag}**, {acc, gyr}, {acc}	The uneven distribution of magnetometer values among the subjects might introduce bias to the model.
3	Window size	0.5, **1**, 2, 5 sec.	The window should be large enough to cover more than one significant event, like steps.
4	Sampling rate	100 Hz, **25 Hz**	Certainly has an influence, I test the two rates already generated.
6	Scenarios	**{sit, walk}**, {walk}, {sit}	Previous researches indicated, that CA can be performed more reliable in walking scenarios (see Section 3.2.5).

adjusted the activation function: When I used ReLU, I got issues with "dying" units In such a situation, the ReLU activation functions output always the same value (zero) for any input. Such a state can happen if a large negative bias term for its weights was learned. I exchanged the ReLUs with the Exponential Linear Unit functions to solve this problem.

The second alternative architecture is based on a Fully Convolutional Network (FCN). I became aware of this FCN architecture through a comparative study on deep learning architecture for time series classification by Fawaz et al. (2018, p. 14f), where that architecture performed quite well (only ResNet performed better), and its relatively low number of parameters was mentioned as an important advantage, which could be beneficial for a small amount of training data. Therefore, I decided to test this architecture as part of the Siamese network. The FCN architecture was proposed by Wang et al. (2016, p. 2), and consists of three sets of layers, where a single set consists of a convolutional layer, followed by a batch normalization layer and an activation layer. The three sets are ordered one after another before a global average pooling layer followed by a softmax layer completes the architecture. The 1D convolutional layers are configured with kernel sizes of $\{8, 5, 3\}$ and $\{128, 256, 128\}$ filters, without striding. The padding is set to "same" for preserving the length of the input vector. ReLU was used as an activation function. I made two adjustments in my implementation of the architecture: I reduced the number of filters drastically to $\{32, 64, 32\}$ to reduce the number of parameters considering the small amount of training data. Also, I exchanged the softmax layer used for classification by a fully connected dense layer with 32 units, which are supposed to generate the deep features to be fed into the OCSVM authentication classifier (Figure 5.26).

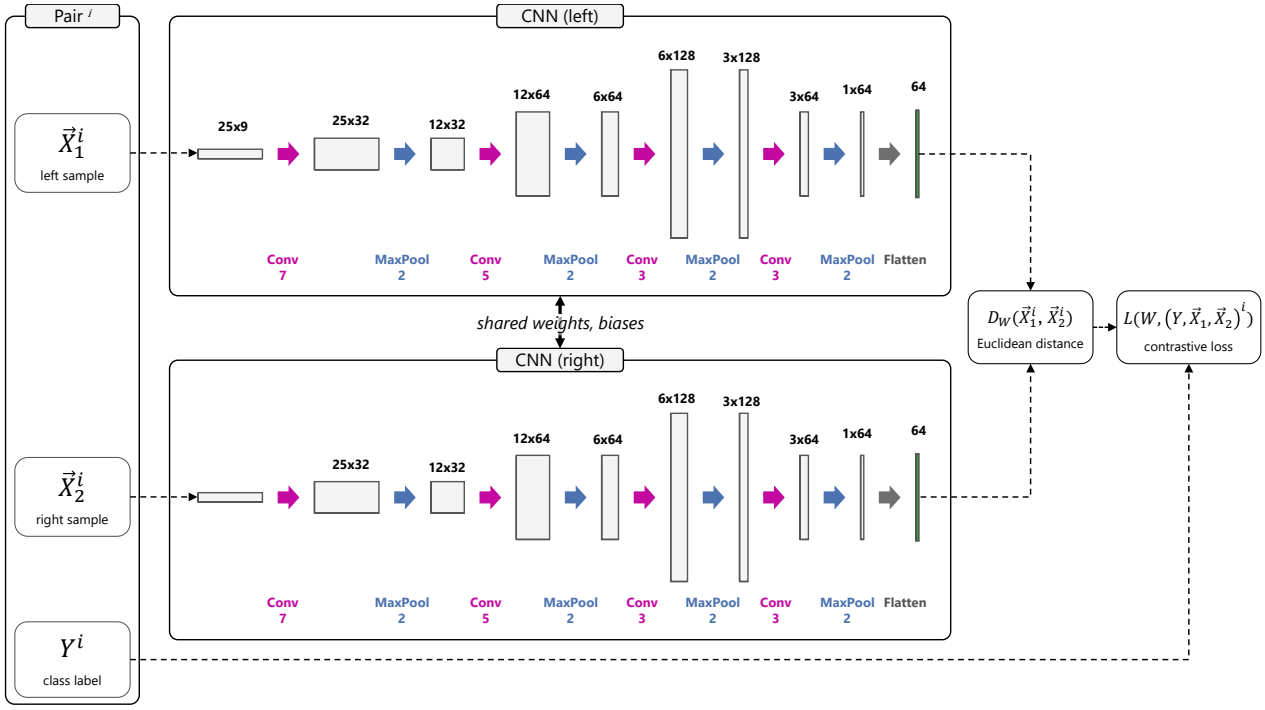


Fig. 5.25.: Architecture of the Siamese CNN with 1D filters. All filters use padding. The vector of the last layer (green) is taken as deep feature.

5.7.2 Results of Parameter Search

The results of the parameter search for the Siamese CNN were assessed by looking at the distance distribution plots, the loss history, and the pairs' "classification accuracy" when applying a threshold to separate the distance distributions of both classes (like the EER threshold in Figure 5.13). Please keep in mind that this accuracy is correlated with but different from the authentication accuracy of the OCSVM. The more or less surprising results (Figure 5.2) are commented in the following.

After training all three variants of the Siamese network, it became evident that all three **architectures** performed quite poor (Figure A.10). None of the architectures was able to distinguish between the positive and negative pairs of the validation set (please remind, that the robust scaling using the "valid approach" was selected as starting point for the parameter search). However, the FCN variant achieved significantly better separation of the training set. Despite the odd loss history, which probably indicates overfitting, I decided to choose this architecture as most promising due to learning ability on the training set.

Regarding the tests with varying **features**, the bad performance of the feature set, including magnetometer data, was expected. The final distance distribution plots (see Figure A.11) visualize that the performance on the training data was better than the other tested variants, but the performance on validation set was quite poor.

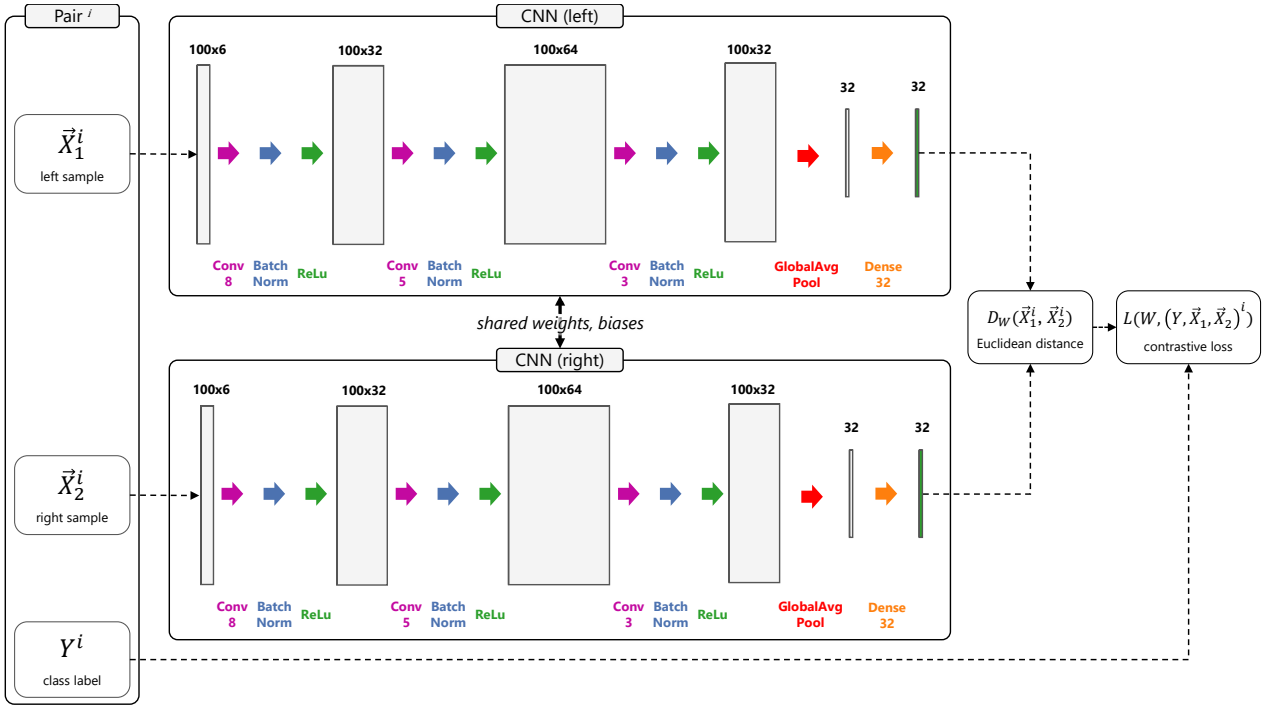


Fig. 5.26.: Architecture of the Siamese CNN, with the FCN subnetworks modeled after Wang et al. (2016, p. 2). All filters use padding. The vector of the last layer (green) is taken as deep feature.

This affirms the already stated assumption that those noisy sensor measurements are probably not suitable for distinguishing smartphone users. Surprisingly, the performance of the test with accelerometer and gyroscope data is slightly lower than the test with accelerometer only. I assume that the performance regarding the number of features, the **window size**, and the **sampling rate** might depend and correlate with the number of trainable parameters in the network as well as the available amount of training data. With rising dimensionality of the input vector, the number of the network's parameters to train increases, which requires more training data. Looking at the test results regarding features, window size, and sampling rate, I assume, that window sizes covering a longer time-span increase the model's performance until a certain point above 5 seconds. The effect that a lower sampling rate and fewer features perform better, on the other hand, might be the result of the smaller Fully Convolutional Network (FCN) implemented due to the limited amount of training data. Of course, this hypothesis needs to be tested, but I leave this to future research.

The fact that the model performed much better for **scenarios** of "walking" than of "sitting", and that "walking & sitting" scenarios result in medium accuracy, was expected, as it confirms the results of related studies (see Subsection 3.2.5). However, I did not miss the opportunity to repeat this test using the originally proposed Siamese CNN with 2D filters, and min-max normalization using the "valid approach". The results were a low accuracy of 77.1% for "walking", 94.8% for "sitting", and

Tab. 5.2.: Parameter search for model improvement. Includes the parameters teste per step (**bold**) and the best achieved accuracy during all epochs (small) with the best parameter emphasized (magenta).

Step	CNN architecture	Features	Window size	Sampling rate	Scenarios
-	Orig. CNN (2D filters)	{acc, gyr, mag}	1 sec	25 Hz	{sit, walk}
1	**Orig. CNN (2D filters)** 50.0% **Orig. CNN (1D filters)** 50.3% **FCN (1D filters)** 0.51%	{acc, gyr, mag}	1 sec	25 Hz	{sit, walk}
2	FCN (1D filters)	**{acc, gyr, mag}** 51.0% **{acc, gyr}** 56.7% **{acc}** 57.4%	1 sec	25 Hz	{sit, walk}
3	FCN (1D filters)	{acc}	**0.5 sec** 56.6% **1 sec** 57.4% **2 sec** 58.3% **5 sec** 63.0%	25 Hz	{sit, walk}
4	FCN (1D filters)	{acc}	5 sec	**25 Hz** 63.0% **100 Hz** 59.0%	{sit, walk}
5	FCN (1D filters)	{acc}	5 sec	25 Hz	**{sit, walk}** 63.0% **{sit}** 57.3% **{walk}** 73.0%
-	FCN (1D filters)	{acc}	5 sec	25 Hz	{walk}

93.8% for the combination of both. I consider this different behavior as an additional indication that the models trained on min-max scaled data are indeed not learning patterns of the actual movements, but more general differences of the distributions.

The parameters identified as best during this incomplete manual search (bottom row in 5.2) were used for a final evaluation of the complete ensemble. The result of this evaluation, with the usage of a larger number of training samples as the only difference to the setting during the parameter search, is reported in the next section.

5.8 Final Evaluation

The evaluation of the final model was performed exactly like for the original and the baseline model (see Section 5.6). Additionally, some insights into the training process are presented. Figure 5.27 display the development of the pair distances during the epochs of training, and Figure 5.28 show the loss history for training and validation set. The latter looks a bit noisy.

The quality of the generated deep feature can be assessed by visualizing them after reducing the vectors into 2D using Principal Component Analysis (PCA) for display in a scatter plot, just like Centeno et al. (2018, p.4, Fig. 3) did. 300 random samples per subject were selected from 10 subjects of the testing set, unseen by the Siamese CNN.

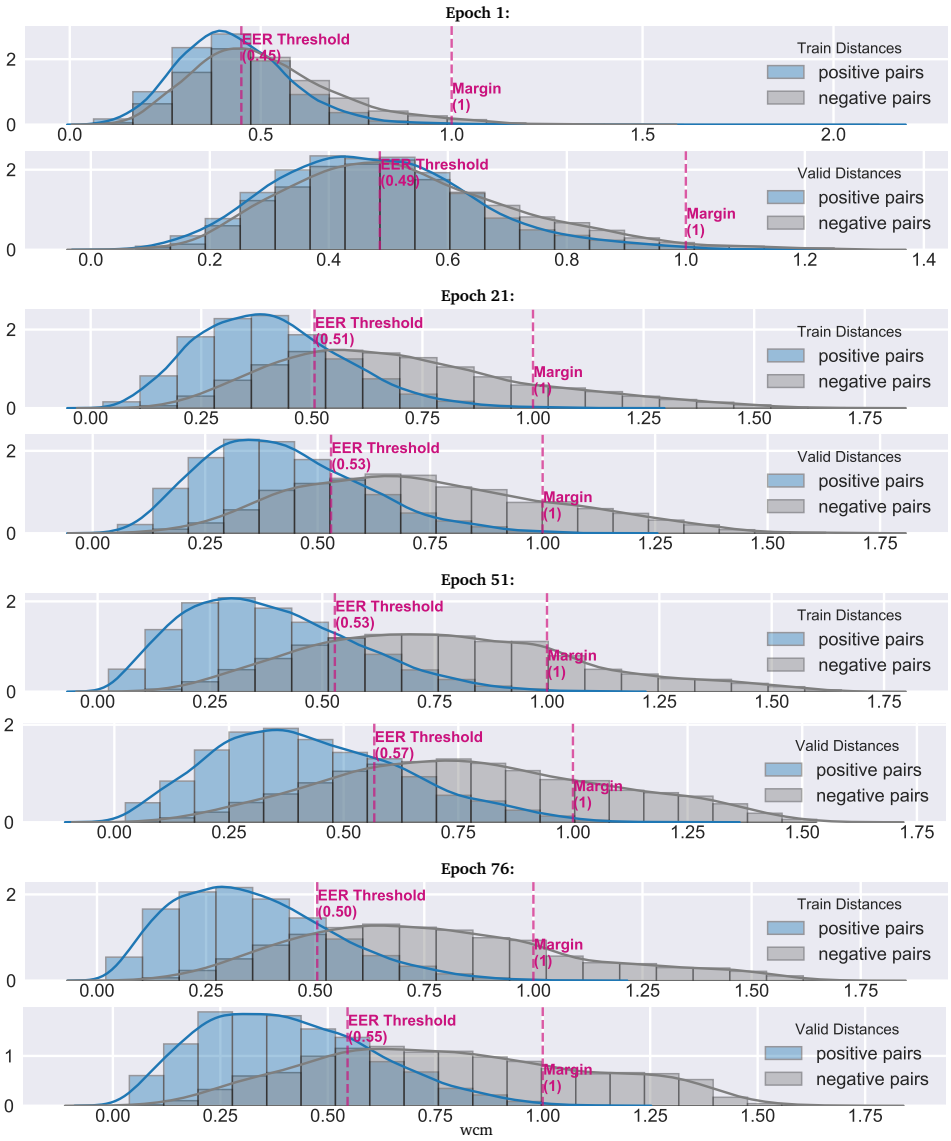


Fig. 5.27.: Siamese FCN (robust) – Distribution of Euclidean distances between positive and negative pairs during training, for training (uneven rows) and validation set (even rows).

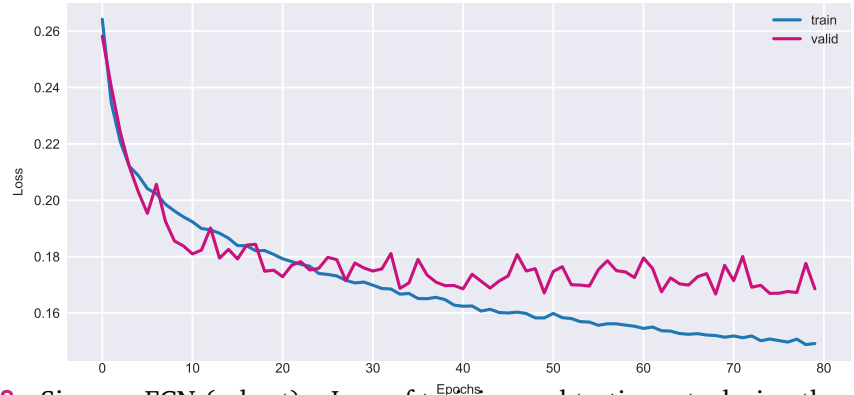
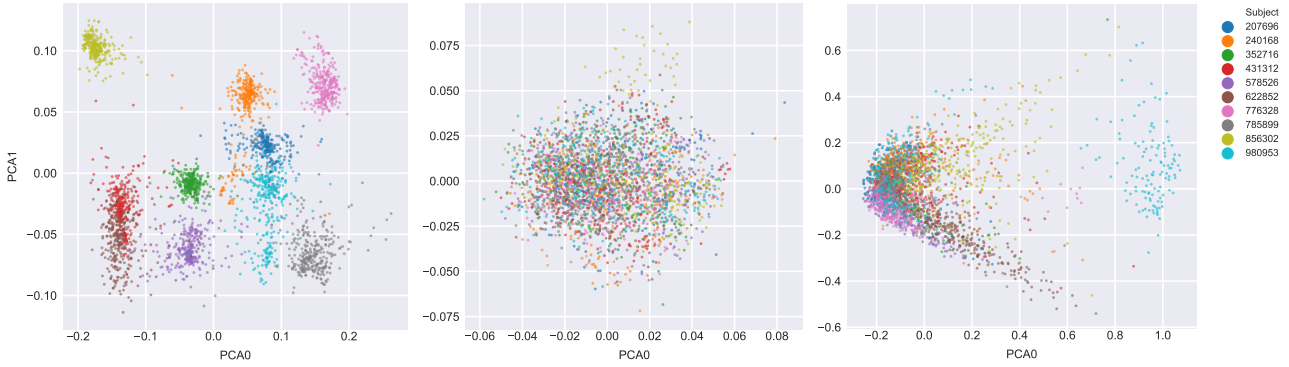


Fig. 5.28.: Siamese FCN (robust) – Loss of training and testing sets during the epochs of training.

The deep feature generated using the “naive approach” with min-max normalization (Figure 5.29, a) compose clearly separated clusters, with some overlaps, e.g. for the brown and red colored subjects. The same network architecture with the same parameters is unable to learn distinguishing patterns if the same data is transformed using the robust scaler instead of min-max. After transformation into 2D using PCA, the subjects mix up in a single cluster. The last plot shows the transformed deep features of the proposed Siamese FCN with some structure, but besides a cluster of a subset of samples from the turquoise subject, we can’t distinguish between subjects either, indicating, that the classification performance will probably be not satisfying.



(a) Siamese CNN 2D (min-max) **(b)** Siamese CNN 2D (robust) **(c)** Siamese FCN (robust)

Fig. 5.29.: Principal Componentent Analysis of deep features from subjects of the testing set, generated by the original Siamese CNN with 2D filters (a, b) and by the proposed Siamese FCN (c).

And indeed, the authentication performance resulted in a mean EER of 36.9% and a mean accuracy of 65.3 % (Figure 5.30 and 5.33). One of the owners performed much worse than the others for an unknown reason. The five seconds long window size leads to a longer “detection delay” of 120.5 s or 24.1 samples on average, with some owners needing significantly longer than other (Figure 5.32). However, the training delay did not increase and is again around 60 s (Figure s5.31).

If we compare the results of this Siamese FCN, using a robust scaler and a valid normalization approach, against the other tested variants (Figure 5.33), it is evident, that its performance is far worse than with “naive approaches”. However, compared to the other variants which are closer to a real-world application, by using a “valid approach” to normalization and reducing the effect of outliers by using a robust scaler, this new model performs best. Of course, the performance is not yet good enough for an application, but during the implementation, I already had lots of ideas for improvements, which I will share in the next chapter.

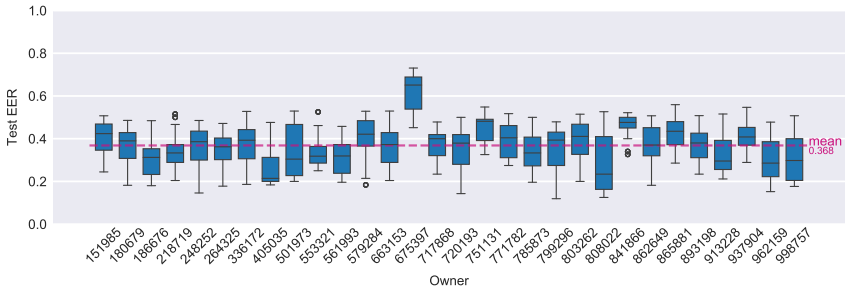


Fig. 5.30.: Siamese FCN (robust, valid approach) – EERs of samples per owner.

Fig. 5.31.: Siamese FCN (robust, valid approach) – Confidence intervals of mean EER expanding over seconds.

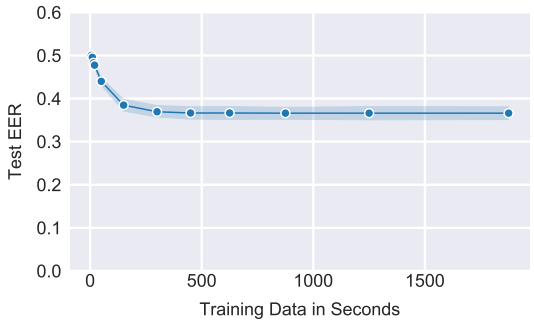


Fig. 5.32.: Siamese FCN (robust, valid approach) – Training delay.

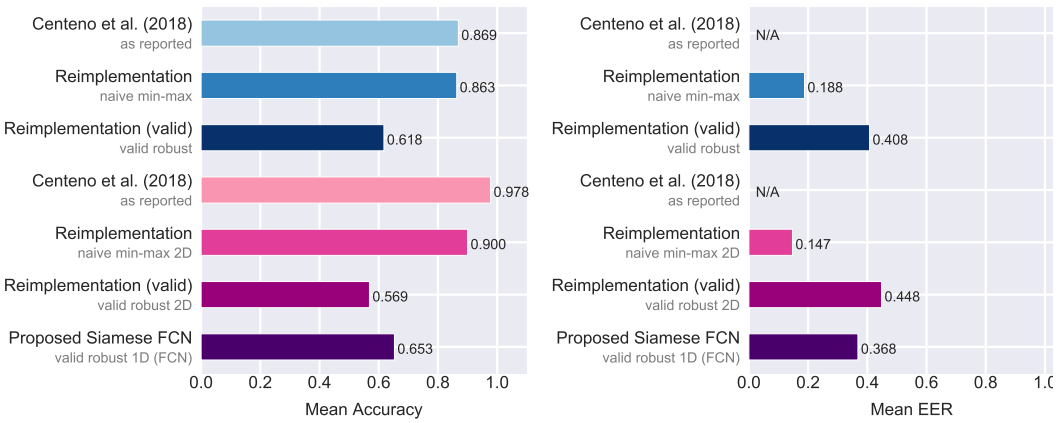


Fig. 5.33.: Classification results - Comparison of major variants.

6 Discussion

After presenting my experiments and the evaluation of the original approach and my proposed variation in the last chapter, it is appropriate to integrate the results into the broader context. First, I present considerations regarding the challenges I faced and discuss their impact on the goal of developing a smartphone base Continuous Authentication (CA) system in general. Afterward, I give concrete recommendations for further improving the approach presented in this thesis. Besides, a compilation of minor bits of knowledge learned during my study can be found in Appendix A.9. They seemed too specific or subjective for the central part of this work, but also too interesting to discard.

6.1 Considerations

The biggest challenge I faced during this thesis was the reproducibility of related studies due to missing information. While this thesis focused on reproducing a single study, I also searched in other studies for essential details, without success. While the bigger picture and general approach are usually presented appropriately, information less important but necessary for a reproducible description is often missing, e.g., how exactly was the normalization or standardization performed? Which methods were used for resampling? Which subjects of the dataset were excluded?

Of course, it is probably impossible to publish all information relevant for implementation in the format of a research paper within a limited number of pages. Also, it is quite difficult to not forget any information interesting for others after working focused on the topic for several months. But there is a simple solution to such obstacles: Publishing the source code. It necessarily contains at least the parameters and processing steps performed during study, which is a vital aid for people trying to reproduce the results. The source code could even provide some reasoning or references in the comments. To my best knowledge, none of the related studies (see Table 3.1) published the source code of their experiments. In my opinion, the research on this topic could be accelerated a lot if this changes.

The results of the various studies are basically incomparable. Whether it is the use of different datasets and metrics, differences in the evaluated scenarios, testing

with and without Cross Validation (CV), or reporting the best or mean results: such differences (see Section 3.4) make it hard to estimate the achieved progress. It would be beneficial for research, if there would be some standard settings for testing the performance of a CA system. Of course, the circumstances and goals of a study might often require a specific evaluation approach, but I'm quite sure, many researchers would welcome, if an easy to implement standard evaluation would be available. In my opinion, developing such a standard would be an exciting undertaking and probably highly appreciated by the research community.

While the H-MOG dataset, the first publicly available dataset dedicated to authentication use cases, is highly appreciated and I encourage its use during evaluation to further improve comparability. Two aspects reduce its otherwise great usefulness: It has been recorded in a controlled setting and does not include data of attacks. While data from a controlled setting has its advantages, it certainly deviates a lot from a real-world scenario, limiting the transferability of the results to a real application. It would also be valuable if the data would include attack scenarios, where the same device is taken out of the owner's possession by an impostor during the same session. Such scenarios would make the testing more realistic and would enable researchers to evaluate criteria like detection delay in a comparable fashion. In my opinion, it could also help communicate results, if the performance could be demonstrated on close to "real" attack scenarios.

As I have learned during this thesis, collecting and compiling such a dataset is a huge undertaking and effort of its own.[35] Nevertheless, it also would be a prestigious project if such a dataset would be made available and may be maintained with a list of studies related to it (cf. MNIST dataset[36]).

Last but not least, another cross-study comparing the various approaches under similar conditions would be highly valuable. The research done by Khan et al. (2014) in this field is very beneficial, but considering the many new approaches proposed in the last five years, an updated comparison seems suitable. Because of the reserve to publish source code, this would be an extensive study of its own, but due to the great variety of machine learning methods also a fascinating one.

[35] My own collection of data in a real-world setting with 10 subject over 14 days did not work out too well. Difficulties in the data recording due to the heterogenous Android smartphone landscape thinned the collected data down to 6 subjects. Issues with energy saving functions and differences in the activity recognition API among the subjects' smartphones slowed down the implementation of the preprocessing too much to be able to perform tests within the time frame of this thesis.

[36] The website of the famous MNIST dataset for handwriting recognition with a curated list of related studies is an excellent example: `http://yann.lecun.com/exdb/mnist/`.

6.2 Recommendations

Besides the general suggestions in the previous section, I gathered lots of ideas during working on this thesis to improve the presented Siamese network approach. Some of those have been already described in this thesis (see Section 5.7). Others could not be tested in my limited time frame. The latter are listed here as recommendations for further improvement of the model, prioritized by considering assumed impact and effort.

1. **Optimize hyperparameters per owner:** The random search for best hyperparameters of the OCSVM revealed that the optimal parameter values vary between the different owners (see Section 5.4). Tuning those parameters using a validation split of the respective owner's training data, instead of using the same hyperparameters for all owners, is probably a quick win regarding the model's performance. While this step would increase the complexity in a productive application slightly, it is perhaps still feasible.
2. **Train Siamese CNN with owner data:** In the presented approach, the deep feature generating Siamese CNN was trained using data from a separate set of subjects. This was done with the idea in mind to train a single model centrally and distribute it to the various devices in an application scenario. However, it would be interesting to see the impact on the authentication performance, if training data of the respective owner would also be included in the training of the Siamese CNN. Implementing this in a productive application would probably need some effort, but it might be justified depending on its impact.
3. **Extensive parameter search:** In this thesis, I performed a parameter search by following an un-nested search tree, ordered by my educated assumptions. With fewer limitations regarding resources like implementation time and computation power, I expect a more extensive automated parameter search with a broader search range to improve the results quite a lot.
4. **Dynamic contrastive loss margin:** In the given implementation, the margin parameter of the contrastive loss function was defined as constant. During literature research I came across several implementation favoring a dynamically adjusted value for the margin. E.g., one approach is to set redefine the margin after every batch or epoch to the value of the distance, which separates both classes' distributions. That is the same value I used for estimating the "accuracy" during the training of the Siamese networks.
5. **Test the Siamese LSTM variant:** During the editing period of this thesis, a new study was published, in which the CNNs of the Siamese network have been exchanged by LSTM networks (Deb et al., 2019, p. 5). Those networks are supposed to capture temporal aspects of time series better, and therefore might

be the right choice. The study reported promising results, but as it differs in many aspects from the approach implemented in this thesis, it is hard to estimate the impact of the different architecture alone.

6. **Pre-classify user activity:** The results of testing data from "sitting" scenarios versus "walking" scenarios or mixed scenarios suggest that the activity state of the user has quite some impact on the classification. This could be leveraged by detecting the current activity upfront (e.g., via Android API) and either train different models for the various activities or provide this information as an additional feature of the model's input data.

7. **Introduce an absolute coordinate system:** Some studies leveraged the gravity of the Earth to remove its vector of the accelerometer data and detect the spatial orientation of the smartphone. This orientation can be used to transform the sensor data into an absolute coordination system (see Figure 2.6). Some smartphones contain hardware and interfaces to provide this information directly. Thus, consuming it through the API might be more accurate and more accessible than deriving it from the inertial sensors manually. It might be possible that the orientation in space as additional information supports the neural networks in identifying unique patterns.

8. **Include additional data sources:** Gyroscope, accelerometer, and magnetometer have some advantages over other sensors of the smartphone. However, this data could be fused with additional data to improve the reliability of the system. Patel et al. (2016) provide a good overview and also mention less obvious signals, including Wifi access point sightings or background noise detected by the microphone. Also, the combination of a smartphone with other devices, e.g., a smartwatch, is possible (Al-Naffakh et al., 2018). Which data can be accessed and included depends on the requirements of the use case.

9. **Collect more data:** The amount of training data is, of course, very important for deep learning algorithms. The medium-sized H-MOG dataset limited the number of trainable parameters that could be practically used for this thesis. More data and the corresponding computation power could also enable the testing of more complex network architectures.

7 Conclusion

In this thesis, I have evaluated the feasibility of using inertial sensors of a smartphone to classify its current user as the smartphone's owner or an impostor to perform Continuous Authentication (CA). A framework was presented, describing how such an authentication process could like in an application, and an introduction into the leveraged sensors was given.

The extensive review of related work revealed a variety of different approaches and provided an overview of the multiple options available. Various machine learning models used in previous studies were introduced as well as the different preprocessing steps that have been applied. The related work proved the resourcefulness of previous researchers but also unveiled severe deficits, with the complicated reproducibility as the most critical one.

With the concept, I provided an outline of the later experiments, by reasoning the design decisions I took. The decision to leverage a deep feature learning approach was the most impactful one. This method allowed the combination of two benefits: The training of a generic model allows leveraging existing behavioral information, while the one class approach to classification does not depend on sharing data for training the individual authentication models. The tempting idea to skip manual feature engineering and learn the data transformation automatically also played a part in the decision. Some other researchers already have been investigating those possibilities in the past. Therefore, I decided to use the study by Centeno et al. (2018) as a basis. As the H-MOG dataset used in this study is publicly available, I had the chance to experiment with the reproducibility of their approach, which is based on an ensemble of a Siamese CNN for feature learning and an OCSVM for authentication.

Unexpected difficulties arose during the reimplementation of the Siamese CNN approach, rooted in missing details about the original implementation and inconsistencies. Taking quite some assumptions, I was able to reproduce results similar to those reported by the authors. But that was only possible by implementing steps not appropriate for a real-world application and by exploiting outliers, instead of identifying the user's movement pattern. While it is unclear how similar the reimplementation actually got to the original, I proposed a more valid approach, which

with an Equal Error Rate (EER) of 44.8% did not even close reached the classification performance reported by Centeno et al. (2018) and is very close to random. Therefore, I proposed a new variant of the Siamese network, which uses the FCN architecture for the Siamese subnetworks. After manual parameter optimization, the model reached an EER of 37.4% in its final evaluation, which is also far from being satisfying, but better than my valid reimplementation of the original model.

While discussing general issues of the research in CA, I argued for the release of the source code along with the related studies, before proposing various ideas for future research. This also includes a list of prioritized suggestions on how the proposed FCN based Siamese network could be tuned to achieve better results.

I want to end this thesis by pointing out the high responsibility when dealing with biometric identification features. Such information has intrinsically different characteristics and a completely different value, which must be taken into account by all stakeholders when used in practical applications.

> *If someone steals your password, you can change it. But if someone steals your thumbprint, you can't get a new thumb. The failure modes are very different.*
>
> **— Bruce Schneier**
> (Computer security expert and writer)

Appendix

A

A.1 Further related studies

Tab. A.1.: Studies related to CA not matching the relevance criteria for this thesis.

Study	Best Model	Reason for exclusion
Bo et al. (2013)	OCSVM	Uses touch inputs and application usage.
Frank et al. (2013)	k-NN, SVM	Uses touch inputs only.
Buriro et al. (2016)	MLP	Requires the user to draw his signature on the device.
Mahbub et al. (2016)	Ensemble	Uses face recognition, touch inputs, and geolocation only.
Fridman et al. (2017)	Ensemble	Uses touch inputs, application usage, and geolocation only.

A.2 Commonly computed Features

Tab. A.2.: Commonly computed features and number of resulting features (NF) per 3-axis sensor. Aggregated from Shen et al. (2018, p. 52), Al-Naffakh et al. (2018, p. 21), Parimi et al. (2018, pp. 3f).

Type	Feature Set	NF	Description
Time Domain	Mean	3	Average value
	Median	3	Most often occurring values
	Minimum	3	Lowest value
	Maximum	3	Highest value
	Range	3	Difference between highest and lowest values
	Variance	3	Spread of values around their mean
	STD	3	Standard deviation of the values
	Kurtosis	3	Tailedness of value distribution
	Skewness	3	Measure of symmetry of distribution
	SMA	3	Signal Magnitude Area (SMA) or signal energy
	Summed SMA	1	SMA of the three axis signals combined
	Quantiles	3·x	Separating partitions in the values distribution
	IQR	3	Interquartile Range (IQR), a measure of variability
	Cross-mean Rate	3	Fluctuation of the signal
Frequency Domain	Entropy	3	Dispersion of signal
	Peek occurrences	3	Number of peeks occurred
	Time between peeks	3	Average time between peeks
	Slope between peeks		(ul-Haq2018, p.28)
	Peek to peek signal value		(ul-Haq2018, p.28)
	Max./Min Latency		(ul-Haq2018, p.28) s
	ALAR		Absolute Latency to Amplitude Ratio
Other	Correlation coeff.	3	Relationship between two axes
	Cosine similarity	3	Pairwise cosine similarity measurements between axes
	Covariance	3	Pairwise covariances between axes
	DTW	3	Dynamic Time Warping
	Band Power	3	Dynamic Time Warping
	SNR	3	Signal to Noise Ratio

A.3 Parameters for OCSVM Approach

Tab. A.3.: Best parameters used for OCSVM baseline model and generating its input vectors, as provided by Centeno et al. (2018, pp. 3–5), along with open questions.

Step	Parameter	Value	Comment
Data	Sensor values	raw	3-axis of accelerometer, gyroscope, magnetometer.
	Sensor frequency	100 Hz	
	Window length	0.5 sec	Resulting in input vector length of 150.
	Normalization	MinMax(0,1)	Channel wise, per subject[a].
OCSVM	Train/Test subjects	30	60 were needed for the Siamese CNN.
	Train sessions	18	Per Subject, randomly selected, but with stratified task types.
	Train observations	6750	From owner only. Resulting in 135 samples for given window.
	Test sessions	6	Per Subject, resulting in 870 scenarios of 1 vs. 1.
	Kernel	RBF	Information on C or ν is missing.
	Gamma	0.0001–100	Used as threshold for ROC[a].

Open Question	Selected Solution / Assumption
Which method was used for resampling to 25 Hz?	Use mean over a sliding window of 4 samples width, and 4 samples step width.
Which step width was used during feature generation?	To keep the number of samples stable, I used a 0.5 s step width, which can be used with all tested window sizes (> 0.5 s).
Which 30 subjects were used for training and testing?	Select 30 subjects out of the 90 randomly.
Which 10 of the 100 subjects were excluded?	3 Subjects with incomplete sessions, plus 3 subjects with the most data and 4 subjects with the least data (“outliers” regarding the number of samples).
Which values for γ and ν were used?	Perform a random search with the remaining 60 of the 90 subjects.[b]
How was subject wise normalization performed?	Test two approaches: before splitting data into train and test sets, and after.

[a] Via correspondence with Mr. Centeno.
[b] The resulting values are presented in Chapter 5.6.

A.4 Parameters for Siamese CNN Approach

Tab. A.4.: Best parameters used for selecting/preprocessing data, Siamese CNN feature extraction model, and OCSVM authentication model, as provided by Centeno et al. (2018, pp. 3–5), along with open questions.

Step	Parameter	Value	Comment
Data	Sensor values	raw	3-axis of accelerometer, gyroscope, magnetometer.
	Sensor frequency	25 Hz	
	Window length	1 s	Resulting in input vector length of 225.
	Normalization	MinMax(0,1)	Channel wise, per subject[a].
Siamese CNN	Train subjects	60	Separated from subjects used for OCSVM.
	Train observations	6750	Per subject, resulting in 270 samples per subject.
	Train samples	270	Per subject
	Train pairs	8100	Implicitly given $(60 \cdot 270 \div 2)$. 50% positive, 50% negative pairs.
	CNN Layers	4 Conv., Max Pool.	Convolutional layers and Max Pooling layers are alternated.
	Max Pooling	2x2	
	Conv. Layers	32 (7x7), 64 (5x5), 128 (3x3), 22 (3x3)	Filter number of last layer is deduced: it is stated to be adjusted to result in a ~64 dimensional output vector.
	Distance Function	Eucl. Dist.	Euclidean distance.
	Loss	Contr. Loss	Contrastive loss function.
OCSVM	Train/Test subjects	30	Separated from subjects used for Siamese CNN.
	Train sessions	18 p. Subj	Randomly selected, stratified task types.
	Test sessions	6 p. Subj.	Resulting in 870 scenarios of 1 vs. 1
	Kernel	RBF	Information on C or ν is missing.
	Gamma	0.0001–100	Used as threshold for ROC[a].

Open Question	Selected Solution / Assumption
Which value was chosen for the margin of the contrastive loss function?	I used distribution plots of the loss of positive and negative pairs to tune the margin.[b]
How to apply 2D filters on 2D input vectors?	That is not possible. I assume 3D input blocks were meant.
How is the ordering of the sensor axis in the input data?	This is relevant for the 2D convolutional filters, but not provided. I use $Acc_{x,y,z}$, $Gyr_{x,y,z}$, $Mag_{x,y,z}$.
How was the hyperparameter optimization performed for the Siamese CNN?	By testing distance based EER on a validation set.
How did the validation set look like?	8 samples from 6 of 24 sessions per user.
Which activation functions were used for the convolutional layers?	I decided to use ReLU, as it is known to work well with CNNs, see, e.g., Nair and E. Hinton (2010).

[a] Via correspondence with Mr. Centeno.
[b] The resulting values are presented in Chapter 5.6.

A.5 Authentication Accuracy

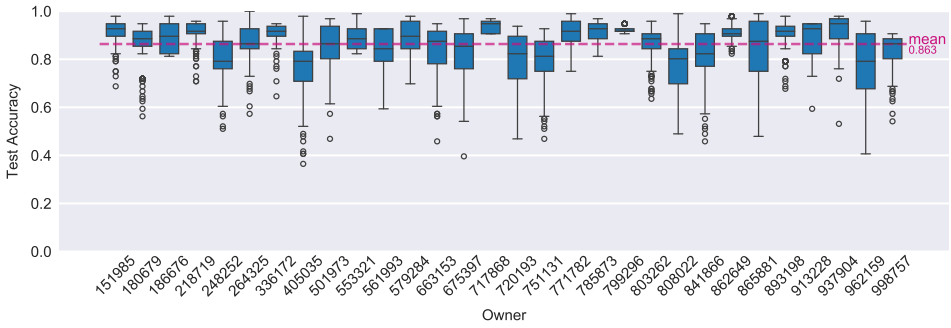


Fig. A.1.: Accuracy of samples per owner – OCSVM (naive approach).

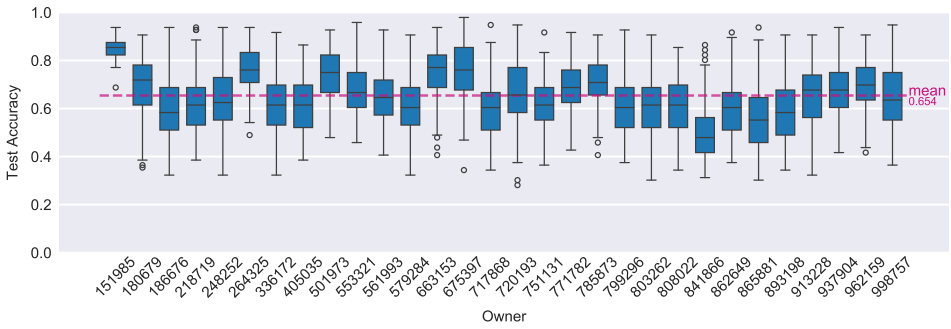


Fig. A.2.: Accuracy of samples per owner – OCSVM (valid approach).

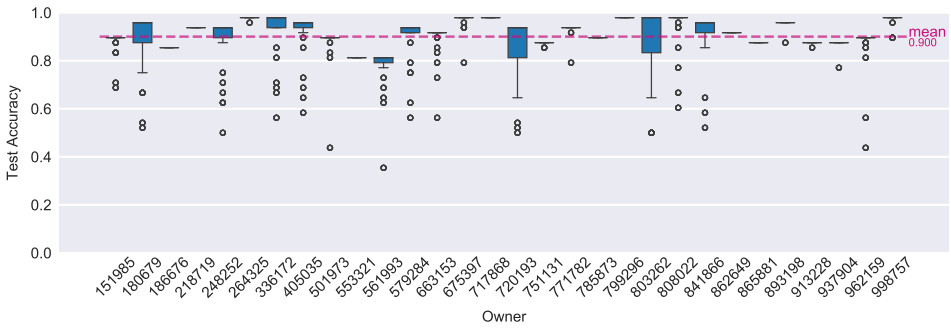


Fig. A.3.: Accuracy of samples per owner – Siamese CNN (naive approach).

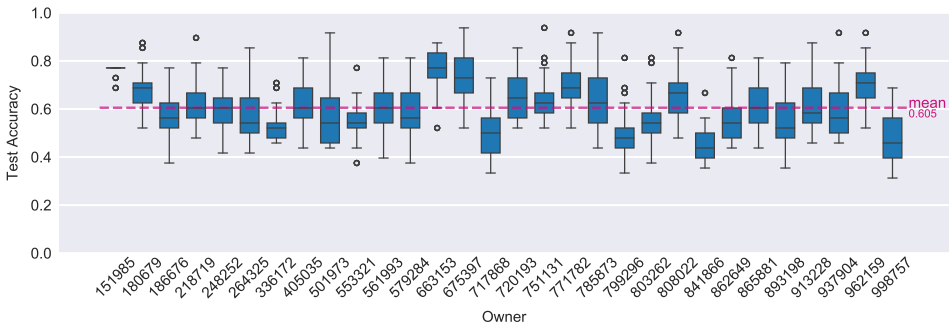


Fig. A.4.: Accuracy of samples per owner – Siamese CNN (valid approach).

A.6 Detection Delay

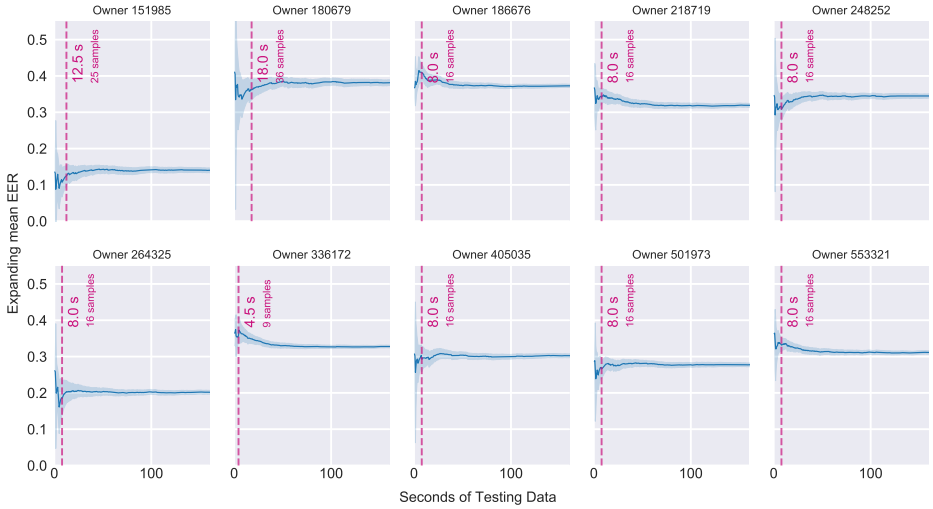


Fig. A.5.: Confidence intervals – OCSVM (valid approach).

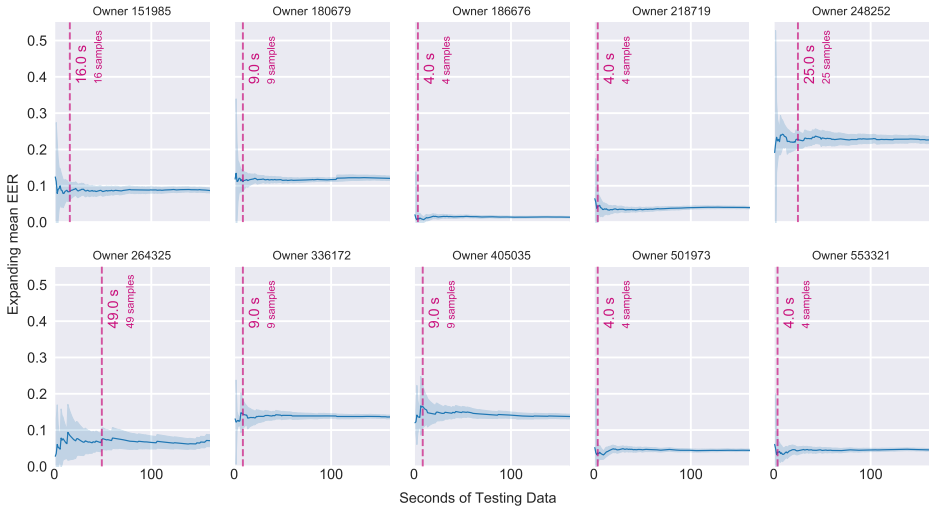


Fig. A.6.: Confidence intervals – Siamese CNN (naive approach).

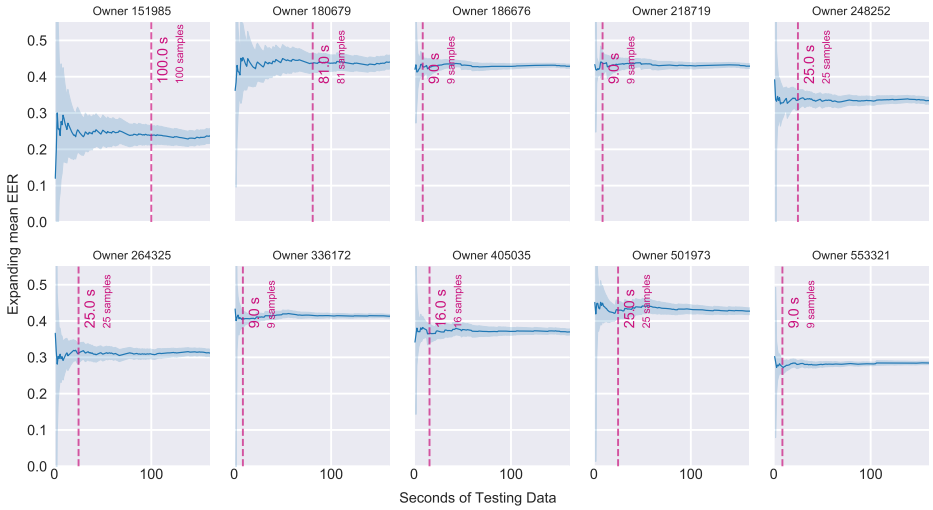


Fig. A.7.: Confidence intervals – Siamese CNN (valid approach).

A.7 Pair Distances during Training

Training history of the Siamese CNN with data normalized using the robust scaler.

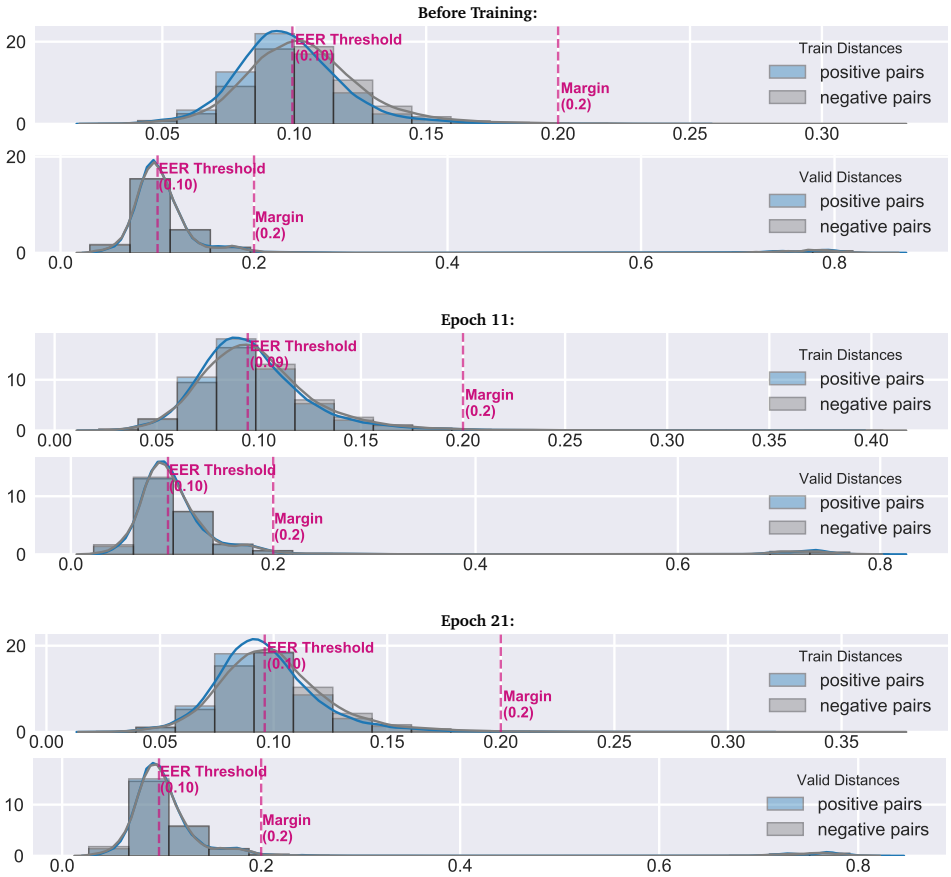


Fig. A.8.: Distribution of Euclidean distances between positive and negative pairs during training, for training (uneven rows) and validation set (even rows).

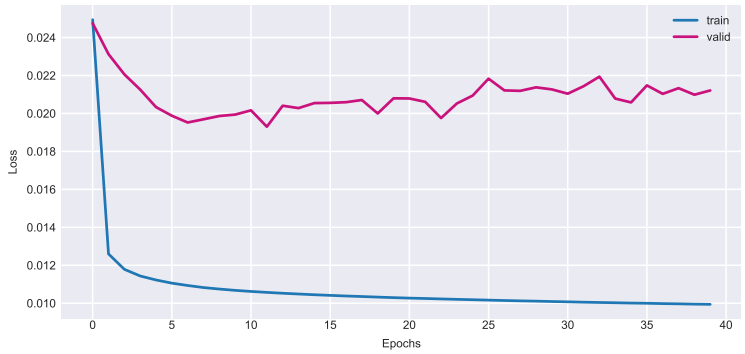


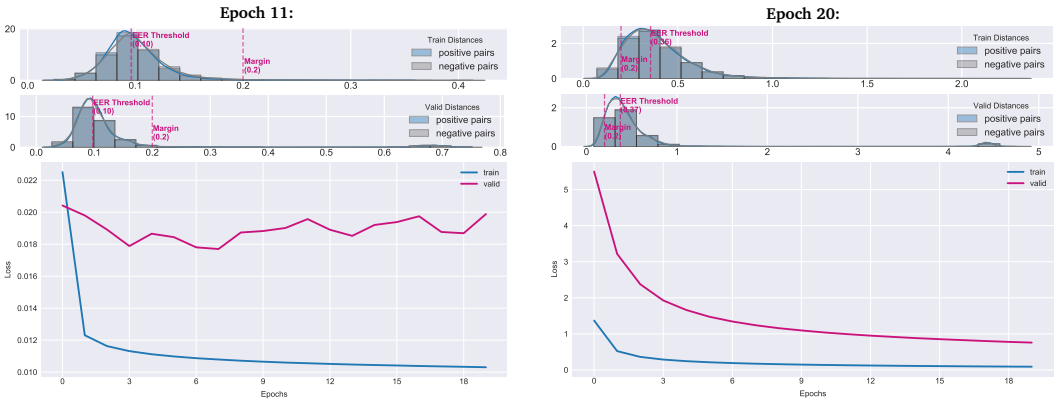
Fig. A.9.: Loss of training and validation sets (robust scaled) after epochs of training data.

A.8 Parameter Search Results

This section contains the results of the tested parameter for the Siamese CNN (see Section 5.7). For every tested variant, the final pair-distance plot and loss history of training and validation set after n epochs of training are displayed. For the full training history, see the HTML exports of the corresponding test runs located in `/reports/optimization/` of the repository (Büch, 2019).

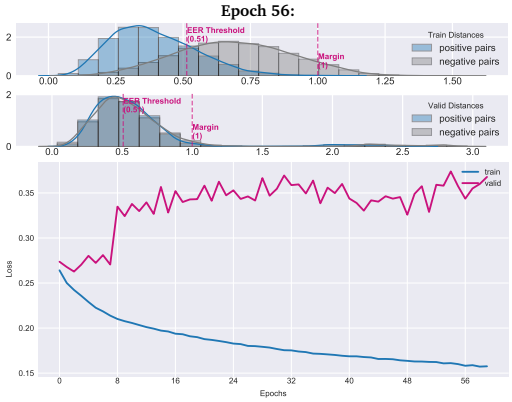
CNN architecture

Parameter	Value
CNN architecture	Original CNN (2D filters), Original CNN (1D filters), FCN (1D filters)
Features	{acc, gyr, mag}
Window size	1
Sampling rate	25 Hz
Scenarios	{sit, walk}



(a) Original CNN (2D filters)

(b) Original CNN (1D filters)

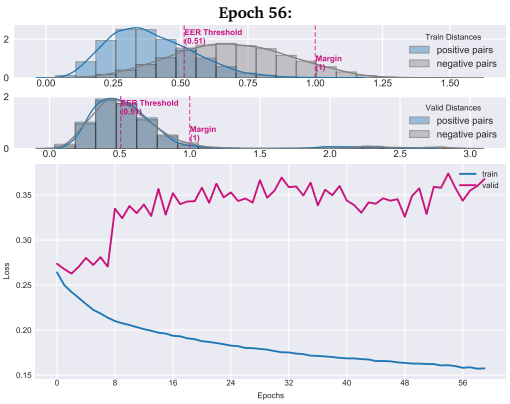


(c) FCN (1D filters)

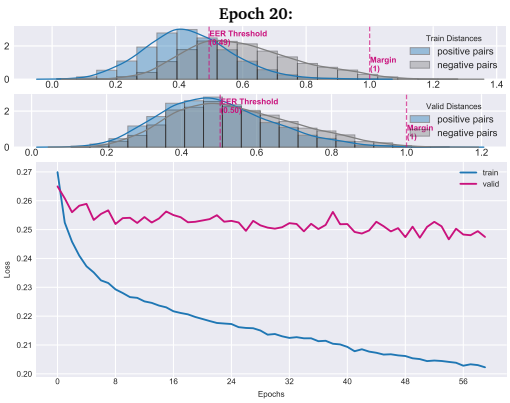
Fig. A.10.: Results of varying CNN architectures.

Features

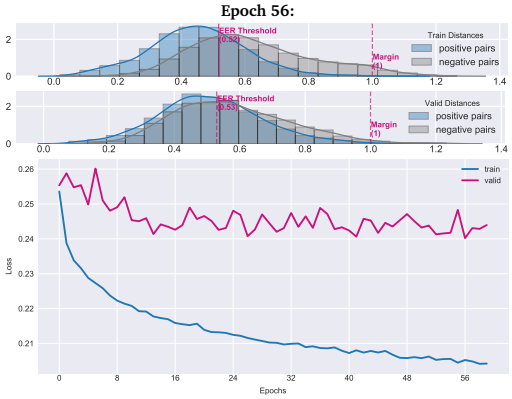
Parameter	Value
CNN architecture	FCN (1D filters)
Features	{acc, gyr, mag}, {acc, gyr}, {acc}
Window size	1 sec.
Sampling rate	25 Hz
Scenarios	{sit, walk}



(a) {acc, gyr, mag}



(b) {acc, gyr}

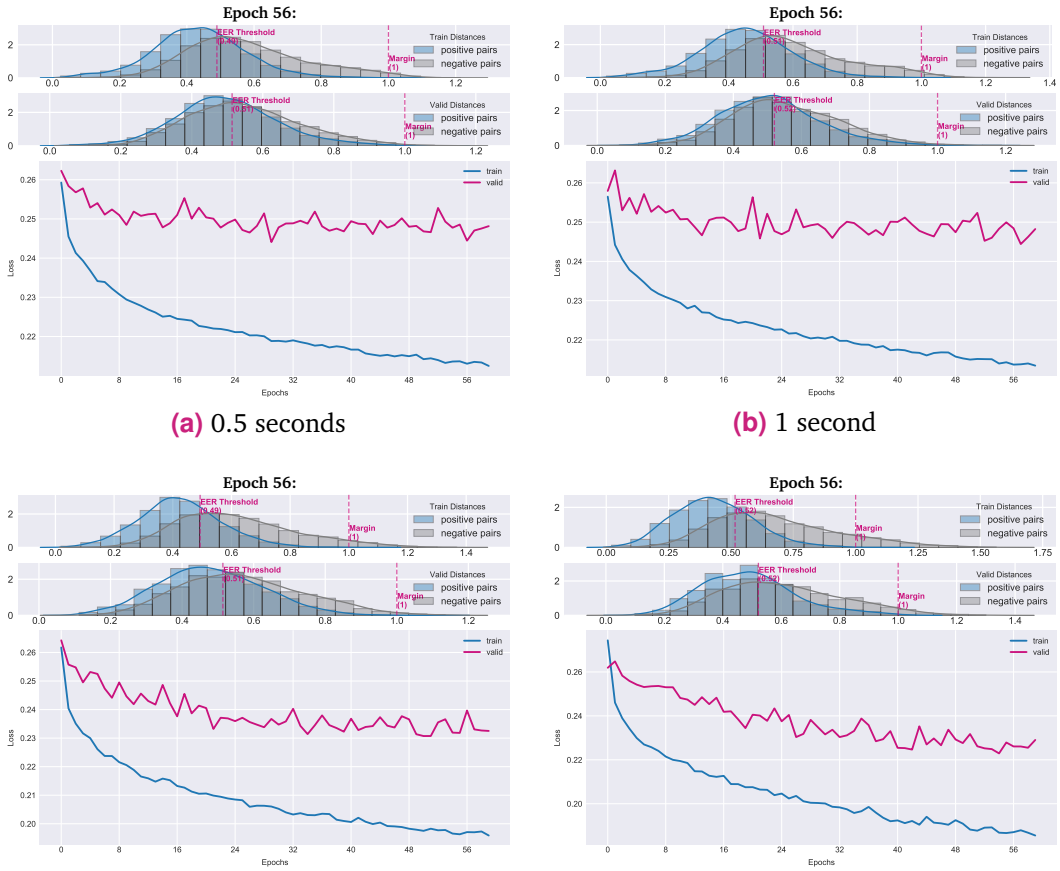


(c) {acc}

Fig. A.11.: Results of varying features.

Window Size

Parameter	Value
CNN architecture	FCN (1D filters)
Features	{acc}
window size	0.5, 1, 2, 5 sec.
Sampling rate	25 Hz
Scenarios	{sit, walk}



(a) 0.5 seconds

(b) 1 second

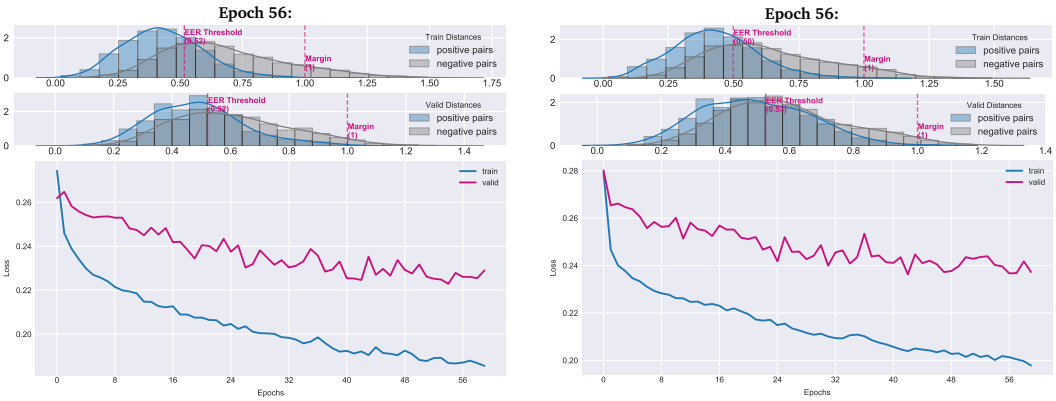
(c) 2 seconds

(d) 5 seconds

Fig. A.12.: Results of varying window sizes.

Sampling Rate

Parameter	Value
CNN architecture	FCN (1D filters)
Features	{acc}
window size	5 sec.
Sampling rate	25 Hz, 100 Hz
Scenarios	{sit, walk}

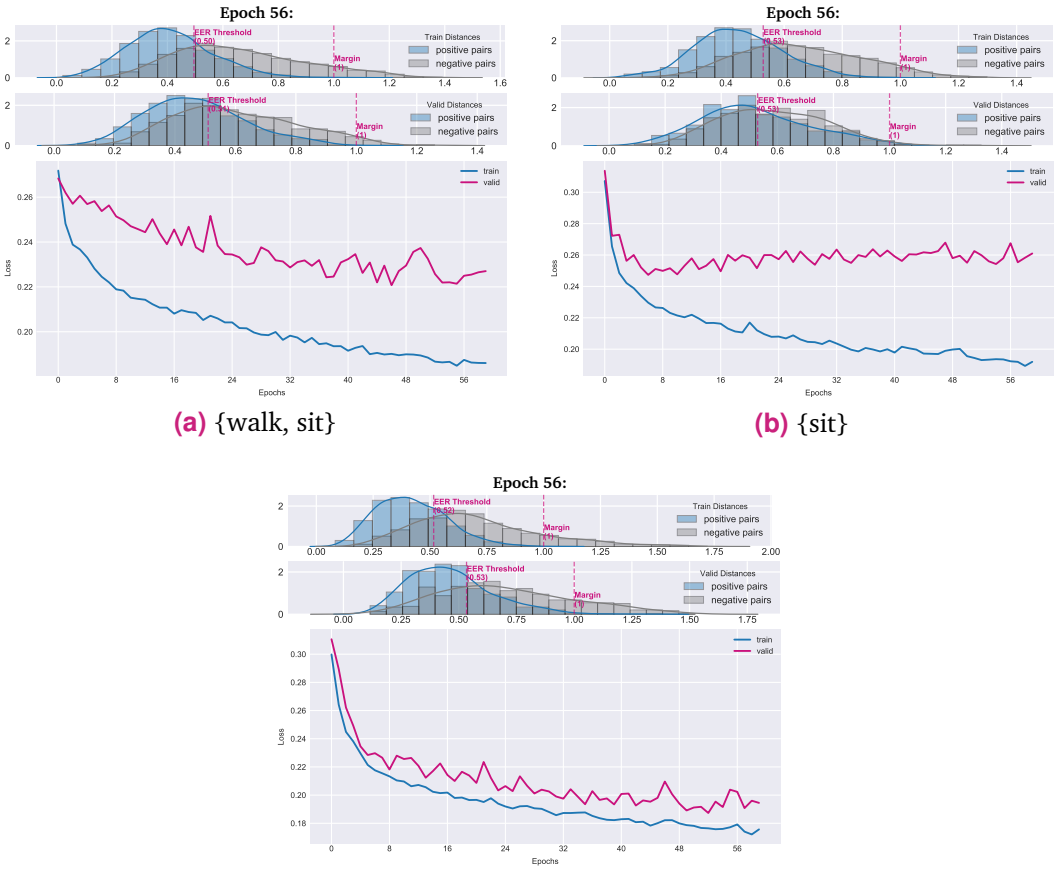


(a) 25 Hz **(b)** 100 Hz

Fig. A.13.: Results of varying sampling rate.

Scenarios

Parameter	Value
CNN architecture	FCN (1D filters)
Features	{acc}
window size	5 sec.
Sampling rate	25 Hz
Scenarios	{sit, walk}, {sit},{walk}



(a) {walk, sit}

(b) {sit}

(c) {walk}

Fig. A.14.: Results of varying scenario selection.

A.9 Minor Learnings

During the process of implementation, I learned lots of small things, too insignificant for the precious space of the main chapters, but also too useful to discard. Therefore, I decided to list them here in the Appendix:

Reading and writing large datasets
As expected, reading data directly from the original data format (CSV) was slow. I tried SQLite, which performed really well but to leverage its performance, I had to write most of the logic in SQL-statements, which was cumbersome and decreased code clarity. Then I experimented with the Dask library[37] and stored the data in Parquet format, but the learning curve of the library turned out to be harder than expected for more complex operations. Also, I ran into dependency and compatibility issues when I tried to use Dask on different operating systems. In the end, I compromised by using HDF format. That turned out to be quite fast, very easy to use, and has native support by Pandas. However, it doesn't allow lazy operations like Dask and therefore might not be the right choice if the data exceeds the size of memory.

Data collection on Android smartphones
Using Android smartphones for data collection is quite challenging, due to the fragmented market of multiple vendors: The variety of hardware components, Android OS versions, and software customization by the vendors decreases compatibility and stability of apps. Additionally, restricted rights for user or app makes it difficult to override Android system functionality, like running an app in the background. Battery saving functionalities, especially when customized by the vendor, complicate the usage additionally. Also, the Android activity API, which was leveraged to detect the current user's activity (walking, vehicle, still, etc.) seems to behave quite differently across smartphones. Some phones seem to report the activity in high frequency, while others so to report only changes in the activity. However, I do not know whether the data logger implementation causes this issue or, e.g., different Android versions.

Reproducing results
It is better to not take anything for granted, just because it is common sense. It matters what is explicitly written. For the own implementation, try to provide any information needed for reproduction, but noticed, how hard this is after working deep on the topic for a long time. Therefore, publishing the source code is essential, and I will appreciate studies with published source code more highly in the future.

Sanity checking of processing steps
In the beginning, I overlooked some implementation errors, which led to unexpected results and was easily misleading. Then I started writing tests for the individual intermediate steps, which significantly improved the situation. While I implemented the tests as simple debugging outputs in this thesis, I plan to work with proper unit testing for future projects, even in Jupyter Notebooks.

[37] `https://docs.dask.org/en/latest/`

Useful 3rd-party modules

Python is famous for its ecosystem of 3rd party modules, and while it is always a good idea to not depend on them too much, I discovered four modules, which I didn't come across before, and which I found quite useful:

- `tqdm` (`https://tqdm.github.io/`)
  Simple to use progress bars for terminal and Jupyter, also displaying time per iteration and remaining time.
- `dataclasses` (`https://docs.python.org/3/library/dataclasses.html`)
  Backport of the "dataclasses" introduced with Python 3.7. Very useful for maintaining data objects. I used it as a feature-rich alternative for NamedTuples.
- `pandas-profiling` (`https://github.com/pandas-profiling/pandas-profiling`)
  A module providing a statistical summary of pandas dataframes in the form of a neat, interactive report. (Only for small to medium sized dataframes.)
- `jupyterlab-toc` (`https://github.com/jupyterlab/jupyterlab-toc`)
  An extension of JupyterLab that displays all markdown headlines as a table of content in the sidebar. Very useful for navigating more extended notebooks.

Bibliography

Al-Naffakh, N., N. Clarke, and F. Li (2018). „Continuous User Authentication Using Smartwatch Motion Sensor Data“. In: *Trust Management XII*. Cham, Swiss: Springer, pp. 15–28. DOI: 10.1007/978-3-319-95276-5_2 (cit. on pp. 17, 18, 78, 82).

AlgoSnap Inc. (2015). *CrowdSignals.io*. URL: http://crowdsignals.io/ (visited on Jan. 21, 2019) (cit. on p. 14).

Ashbourn, J. (2015). *Practical Biometrics*. London, UK: Springer. DOI: 10.1007/978-1-4471-6717-4 (cit. on p. 6).

Bo, C., L. Zhang, X.-Y. Li, Q. Huang, and Y. Wang (2013). „SilentSense: Silent User Identification via Touch and Movement Behavioral Biometrics“. In: *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*. MobiCom '13. Miami, FL: ACM, pp. 187–190. DOI: 10.1145/2500423.2504572 (cit. on p. 81).

Bosch Sensortec (2018a). *BMI263 - Product Description*. URL: https://www.bosch-sensortec.com/bst/products/all_products/bmi263 (visited on Nov. 6, 2018) (cit. on p. 10).

– (2018b). *BMI263 - Product Flyer*. URL: https://ae-bst.resource.bosch.com/media/_tech/media/product_flyer/BST-BMI263-FL000.pdf (visited on Nov. 6, 2018) (cit. on p. 10).

Bromley, J., J. W. Bentz, L. Bottou, et al. (1993). „Signature Verification using a "Siamese" Time Delay Neural Network“. In: *International Journal of Pattern Recognition and Artificial Intelligence* 7, p. 25. DOI: 10.1142/S0218001493000339 (cit. on p. 37).

Büch, H. (2019). *ContinAuth*. GitHub repository. URL: https://github.com/dynobo/ContinAuth (visited on June 30, 2019) (cit. on pp. 39, 66, 88).

Buriro, A., B. Crispo, F. Delfrari, and K. Wrona (2016). „Hold and Sign: A Novel Behavioral Biometrics for Smartphone User Authentication“. In: *2016 IEEE Security and Privacy Workshops (SPW)*, pp. 276–285. DOI: 10.1109/SPW.2016.20 (cit. on p. 81).

Centeno, M. P., A. v. Moorsel, and S. Castruccio (2017). „Smartphone Continuous Authentication Using Deep Learning Autoencoders“. In: 15^{th} *Annual Conference on Privacy, Security and Trust (PST)*, pp. 1–9. DOI: 10.1109/PST.2017.00026 (cit. on pp. 5, 14, 15, 22–26, 31, 45).

Centeno, M. P., Y. Guan, and A. v. Moorsel (2018). „Mobile Based Continuous Authentication Using Deep Features“. In: *Proceedings of the* 2^{nd} *International Workshop on Embedded and Mobile Deep Learning (EMDL)*, pp. 19–24. DOI: 10.1145/3212725.3212732 (cit. on pp. 2, 4, 14, 15, 19, 21, 25, 37–39, 41, 44–47, 49, 51–54, 57, 58, 61, 62, 66, 70, 79, 80, 83, 84).

Crouse, D., H. Han, D. Chandra, B. Barbello, and A. K. Jain (2015). „Continuous authentication of mobile user: Fusion of face image and inertial Measurement Unit data“. In: *2015 International Conference on Biometrics (ICB)*, pp. 135–142. DOI: 10.1109/ICB.2015.7139043 (cit. on p. 27).

Dasgupta, D., A. Roy, and A. Nag (2017). *Advances in User Authentication (Infosys Science Foundation Series)*. Cham, Switzerland: Springer. DOI: 10.1007/978-3-319-58808-7 (cit. on pp. 3, 4, 7).

Deb, D., A. Ross, A. K. Jain, K. O. Prakah-Asante, and K. V. Prasad (2019). „Actions Speak Louder Than (Pass)words: Passive Authentication of Smartphone Users via Deep Temporal Features“. In: *CoRR*. arXiv: arXiv:1901.05107 (cit. on pp. 14–16, 19, 25, 26, 38, 77).

E.-u.-Haq, M., M. A. Azam, U. Naeem, S. ur Rèhman, and A. Khalid (2017). „Identifying Smartphone Users based on their Activity Patterns via Mobile Sensing“. In: *Procedia Computer Science* 113, pp. 202–209. DOI: 10.1016/j.procs.2017.08.349 (cit. on pp. 13, 14, 20).

E.-u.-Haq, M., M. A. Azam, U. Naeem, Y. Amin, and J. Loo (2018). „Continuous authentication of smartphone users based on activity pattern recognition using passive mobile sensing“. In: *Journal of Network and Computer Applications* 109, pp. 24–35. DOI: 10.1016/j.jnca.2018.02.020 (cit. on pp. 14, 16, 17, 21).

Fawaz, H. I., G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller (2018). „Deep learning for time series classification: a review“. In: DOI: 10.1007/s10618-019-00619-1. arXiv: http://arxiv.org/abs/1809.04356v4 [cs.LG] (cit. on p. 67).

Fraden, J. (2016). *Handbook of Modern Sensors*. Cham Heidelberg New York Dordrecht London: Springer. DOI: 10.1007/978-3-319-19303-8 (cit. on p. 11).

Frank, J., S. Mannor, and D. Precup (2010). „Activity and Gait Recognition with Time-delay Embeddings“. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI'10. Atlanta, GA: AAAI, pp. 1581–1586 (cit. on p. 14).

Frank, M., R. Biedert, E. Ma, I. Martinovic, and D. Song (2013). „Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication“. In: *IEEE Transactions on Information Forensics and Security* 8.1, pp. 136–148. DOI: 10.1109/TIFS.2012.2225048 (cit. on p. 81).

Fridman, L., S. Weber, R. Greenstadt, and M. Kam (2017). „Active Authentication on Mobile Devices via Stylometry, Application Usage, Web Browsing, and GPS Location“. In: *IEEE Systems Journal* 11.2, pp. 513–521. DOI: 10.1109/JSYST.2015.2472579 (cit. on p. 81).

García, S., J. Luengo, and F. Herrera (2015). *Data Preprocessing in Data Mining*. Cham Heidelberg New York Dordrecht London: Springer. DOI: 10.1007/978-3-319-10247-4. URL: https://doi.org/10.1007/978-3-319-10247-4 (cit. on pp. 15–17).

GitHub.com (2016). *Error in Contrastive loss function for the Siamese example*. URL: https://github.com/keras-team/keras/issues/1866 (visited on Apr. 17, 2019) (cit. on p. 54).

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep learning*. Cambridge, MA: MIT Press. URL: http://www.deeplearningbook.org (cit. on p. 23).

Hadsell, R., S. Chopra, and Y. LeCun (2006). „Dimensionality Reduction by Learning an Invariant Mapping“. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100 (cit. on pp. 53, 54).

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning*. New York, NY: Springer. DOI: 10.1007/978-0-387-84858-7. URL: https://doi.org/10.1007/978-0-387-84858-7 (cit. on pp. 20, 35, 36).

Hering, E. and G. Schönfelder, eds. (2018). *Sensoren in Wissenschaft und Technik*. Wiesbaden, Germany: Springer. DOI: 10.1007/978-3-658-12562-2 (cit. on pp. 7–10).

Hinton, G. E. and R. R. Salakhutdinov (2006). „Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786, pp. 504–507. DOI: 10.1126/science.1127647 (cit. on p. 23).

Horst, F., F. Kramer, B. Schäfer, et al. (2016). „Daily changes of individual gait patterns identified by means of support vector machines". In: *Gait & Posture* 49, pp. 309–314. DOI: 10.1016/j.gaitpost.2016.07.073 (cit. on p. 32).

Horst, F., S. R. Lapuschkin, W. Samek, K. R. Müller, and W. I. Schöllhorn (2018). „What is Unique in Individual Gait Patterns? Understanding and Interpreting Deep Learning in Movement Analysis". In: *Book of Abstracts of the Annual Congress of the European College of Sport Science*. Vol. 23, p. 33 (cit. on p. 32).

Illowsky, B. and S. Dean (2013). *Introductory statistics*. Houston, TX: OpenStax College, Rice University. URL: https://openstax.org/details/books/introductory-statistics (visited on May 9, 2019) (cit. on p. 50).

Jain, A. K., A. Ross, and S. Prabhakar (2004). „An Introduction to Biometric Recognition". In: *IEEE Transactions on Circuits and Systems for Video Technology* 14.1, pp. 4–20. DOI: 10.1109/tcsvt.2003.818349 (cit. on p. 6).

Kalantar-zadeh, K. (2013). *Sensors*. Boston, MA: Springer. DOI: 10.1007/978-1-4614-5052-8 (cit. on pp. 8, 9, 11).

Kayacik, H. G., M. Just, L. Baillie, D. Aspinall, and N. Micallef (2014). „Data Driven Authentication: On the Effectiveness of User Behaviour Modelling with Mobile Device Sensors". In: *Proceedings of the Third Workshop on Mobile Security Technologies (MoST)*. URL: https://researchportal.hw.ac.uk/en/publications/data-driven-authentication-on-the-effectiveness-of-user-behaviour (visited on Apr. 29, 2019) (cit. on pp. 14, 32).

Khan, H., A. Atwater, and U. Hengartner (2014). „A Comparative Evaluation of Implicit Authentication Schemes". In: *Research in Attacks, Intrusions and Defenses*, pp. 255–275. DOI: 10.1007/978-3-319-11379-1_13 (cit. on pp. 26, 33, 34, 76).

Kumar, R., V. V. Phoha, and A. Jain (2015). „Treadmill attack on gait-based authentication systems". In: *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–7. DOI: 10.1109/BTAS.2015.7358801 (cit. on p. 32).

Lee, W. and R. B. Lee (2015). „Multi-sensor authentication to improve smartphone security". In: *2015 International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 1–11 (cit. on pp. 14, 21).

– (2017). „Implicit Smartphone User Authentication with Sensors and Contextual Machine Learning". In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 297–308. DOI: 10.1109/DSN.2017.24 (cit. on pp. 14, 20, 21).

Li, Y., H. Hu, G. Zhou, and S. Deng (2018). „Sensor-Based Continuous Authentication Using Cost-Effective Kernel Ridge Regression". In: *IEEE Access* 6, pp. 32554–32565. DOI: 10.1109/ACCESS.2018.2841347 (cit. on pp. 4–6, 14, 15, 17, 25, 44).

Li, Y., H. Hu, and G. Zhou (2019). „Using Data Augmentation in Continuous Authentication on Smartphones“. In: *IEEE Internet of Things Journal* 6.1, pp. 628–640. DOI: 10.1109/JIOT.2018.2851185 (cit. on pp. 14, 17).

Mahbub, U., S. Sarkar, V. M. Patel, and R. Chellappa (2016). „Active user authentication for smartphones: A challenge data set and benchmark results“. In: *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–8. DOI: 10.1109/BTAS.2016.7791155 (cit. on p. 81).

Nair, V. and G. E. Hinton (2010). „Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair“. In: vol. 27, pp. 807–814. URL: http://www.cs.toronto.edu/~fritz/absps/reluICML.pdf (cit. on pp. 53, 84).

Neverova, N., C. Wolf, G. Lacey, et al. (2016). „Learning Human Identity From Motion Patterns“. In: *IEEE Access* 4, pp. 1810–1820. DOI: 10.1109/ACCESS.2016.2557846 (cit. on pp. 6, 26, 31, 44).

Neverova, N. (2016). „Deep learning for human motion analysis“. PhD thesis. Université de Lyon. URL: https://tel.archives-ouvertes.fr/tel-01470466 (visited on Dec. 7, 2018) (cit. on pp. 14, 15, 18, 19, 21).

NIST - National Institute of Standards and Technology (2018). *Computer Security Resource Center, Glossary - authorization*. URL: https://csrc.nist.gov/glossary/term/authorization (visited on Dec. 19, 2018) (cit. on p. 4).

Parimi, G. M., P. P. Kundu, and V. V. Phoha (2018). „Analysis of head and torso movements for authentication“. In: *2018 IEEE 4th International Conference on Identity, Security, and Behavior Analysis (ISBA)*, pp. 1–8. DOI: 10.1109/ISBA.2018.8311460 (cit. on p. 82).

Patel, V. M., R. Chellappa, D. Chandra, and B. Barbello (2016). „Continuous User Authentication on Mobile Devices: Recent progress and remaining challenges“. In: *IEEE Signal Processing Magazine* 33.4, pp. 49–61. DOI: 10.1109/MSP.2016.2555335 (cit. on pp. 5, 31, 78).

Reid, P. (2004). *Biometrics for network security*. Upper Saddle River, NJ: Prentice Hall (cit. on p. 7).

Reyes-Ortiz, J. L., D. Anguita, L. Oneto, and X. Parra (2015). *UCI - Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set*. URL: http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions (visited on Jan. 21, 2019) (cit. on pp. 13, 16, 17).

Reynolds, D. (2009). „Gaussian Mixture Models“. In: *Encyclopedia of Biometrics*. Ed. by S. Z. Li and A. Jain. Boston, MA: Springer, pp. 659–663. DOI: 10.1007/978-0-387-73003-5_196 (cit. on p. 20).

Roy, A., T. Halevi, and N. Memon (2015). „An HMM-based multi-sensor approach for continuous mobile authentication“. In: *2015 IEEE Military Communications Conference (MILCOM)*, pp. 1311–1316. DOI: 10.1109/MILCOM.2015.7357626 (cit. on pp. 14, 15, 21, 22, 25).

Schölkopf, B., R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt (2000). „Support vector method for novelty detection“. In: *Advances in neural information processing systems (NIPS)*, pp. 582–588. URL: http://papers.nips.cc/paper/1723-support-vector-method-for-novelty-detection.pdf (visited on Apr. 9, 2019) (cit. on p. 36).

Schölkopf, B., J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson (2001). „Estimating the Support of a High-Dimensional Distribution“. In: *Neural computation* 13.7, pp. 1443–1471. DOI: 10.1162/089976601750264965 (cit. on p. 36).

Schuckers, M. E. (2009). „Test Sample and Size“. In: *Encyclopedia of Biometrics*. Ed. by S. Z. Li and A. Jain. Boston, MA: Springer, pp. 1328–1332. DOI: 10.1007/978-0-387-73003-5_113 (cit. on p. 51).

Shen, C., Y. Li, Y. Chen, X. Guan, and R. A. Maxion (2018). „Performance Analysis of Multi-Motion Sensor Behavior for Active Smartphone Authentication“. In: *IEEE Transactions on Information Forensics and Security* 13.1, pp. 48–62. DOI: 10.1109/TIFS.2017.2737969 (cit. on pp. 14–17, 20–22, 25, 26, 44, 82).

Shepard, C., A. Rahmati, C. Tossell, L. Zhong, and P. Kortum (2011). „LiveLab: Measuring Wireless Networks and Smartphone Users in the Field“. In: *SIGMETRICS Perform. Eval. Rev.* 38.3, pp. 15–20. DOI: 10.1145/1925019.1925023. URL: http://livelab.recg.rice.edu/traces.htm (cit. on p. 14).

Shi, W., J. Yang, Yifei Jiang, Feng Yang, and Yingen Xiong (2011). „SenGuard: Passive user identification on smartphones using multiple sensors“. In: *2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 141–148. DOI: 10.1109/WiMOB.2011.6085412 (cit. on p. 14).

Shoaib, M., S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga (2016). *Complex Human Activities Dataset*. Pervasive Systems group, University of Twente. URL: https://www.utwente.nl/en/eemcs/ps/research/dataset/ (visited on Jan. 21, 2019) (cit. on p. 13).

Sitová, Z., J. Šeděnka, Q. Yang, et al. (2016). „HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users“. In: *IEEE Transactions on Information Forensics and Security* 11.5, pp. 877–892. DOI: 10.1109/TIFS.2015.2506542 (cit. on pp. 6, 14, 15, 17, 18, 21, 25, 26, 31, 32, 44, 46).

VanderPlas, J. (2016). *Python Data Science Handbook*. Sebastopol, CA: O'Reilly (cit. on pp. 20, 36).

Včelák, J., P. Ripka, and A. Zikmund (2015). „Precise Magnetic Sensors for Navigation and Prospection“. In: *Journal of Superconductivity and Novel Magnetism* 28.3, pp. 1077–1080. DOI: 10.1007/s10948-014-2636-7 (cit. on pp. 9–11).

W3C (2019). *Motion Sensors Explainer*. URL: https://www.w3.org/TR/motion-sensors/ (visited on Jan. 5, 2019) (cit. on pp. 8–12).

Wang, Z., W. Yan, and T. Oates (2016). „Time Series Classification from Scratch with Deep Neural Networks“. In: arXiv: http://arxiv.org/abs/1611.06455v4 [cs.LG] (cit. on pp. 67, 69).

Wikipedia contributors (2018). *Earth's magnetic field – Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/w/index.php?title=Earth%27s_magnetic_field&oldid=873551045 (visited on Dec. 21, 2018) (cit. on p. 9).

WISDM Lab (2012). *WISDM: WIreless Sensor Data Mining - Dataset Home About Datasets Resources Members*. Fordham University, Bronx, NY. URL: http://www.cis.fordham.edu/wisdm/dataset.php (visited on Jan. 21, 2019) (cit. on p. 13).

Yang, Q., G. Peng, D. T. Nguyen, et al. (2014a). „A Multimodal Data Set for Evaluating Continuous Authentication Performance in Smartphones“. In: *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems (SenSys)*, pp. 358–359. DOI: 10.1145/2668332.2668366 (cit. on p. 40).

– (2014b). *H-MOG Data Set: A Multimodal Data Set for Evaluating Continuous Authentication Performance in Smartphones*. URL: http://www.cs.wm.edu/~qyang/hmog.html (visited on Jan. 21, 2019) (cit. on pp. 14, 40).

Zhang, X. (2017). „Support Vector Machines“. In: *Encyclopedia of Machine Learning and Data Mining*. Ed. by C. Sammut and G. I. Webb. Boston, MA: Springer, pp. 1214–1220. DOI: 10.1007/978-1-4899-7687-1_810 (cit. on p. 21).

Zhu, J., P. Wu, X. Wang, and J. Zhang (2013). „SenSec: Mobile security through passive sensing“. In: *2013 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1128–1133. DOI: 10.1109/ICCNC.2013.6504251 (cit. on p. 14).

List of Figures

List of Tables

List of Equations

Acronyms

AMR Anisotropic Magnetoresistance. 9, 10

ANN Artificial Neural Network. 2, 18, 22, 23, 38, 98

CA Continuous Authentication. 1–3, 5, 6, 13–16, 18, 20, 23, 24, 27, 29–34, 38, 39, 69, 73, 75, 76, 81

CHAD Complex Human Activities Dataset. 13, 14

CNN Convolutional Neural Network. 2, 14, 18, 19, 25, 37, 45, 47, 49, 50, 52, 55–59, 61–64, 66, 67, 69, 70, 73, 77, 83, 84, 91, 92, 94–96, 107, 108

CSV Comma-separated values. 46

CV Cross Validation. 50, 51, 53, 58, 61, 76

CVG Coriolis Vibratory Gyroscope. 8

CWRNN Clockwork Recurrent Neural Network. 18, 38

DCWRNN Dense Clockwork Recurrent Neural Network. 14, 18, 21

DT Decision Tree. 21

EER Equal Error Rate. 6, 7, 14, 21, 22, 24, 33, 53, 54, 62, 64, 65, 67

FAR Falsse Acceptance Rate. 6, 7, 14, 25, 26, 33

FCN Fully Convolutional Network. 70, 71

FFT Fast Fourier Transform. 16

FPR False Positive Rate. 14

FRR False Rejection Rate. 6, 7, 14, 25, 26, 33

GMM Gaussian Mixture Model. 14, 18, 20, 21

GMR Giant Magnetoresistance. 9, 10

H-MOG Hand Movement, Orientation and Grasp. 14, 17, 26, 33, 37, 39–41, 46–50, 76

HDF Hierarchical Data Format. 39, 46, 48

HMM Hidden Markov Model. 14, 20–22

IMU Inertial Measurement Unit. 10

IQR Interquartile Range. 82

ISROS Inertial Sensordata from Real-World Office Setting. 39, 41

k-NN k-Nearest Neighbors. 21, 81

KRR Kernel Ridge Regression. 14, 21

LLT LiveLab Traces. 14

LSTM Long Short-Term Memory. 14, 18, 19, 25, 38, 77

MEMS Microelectromechanical System. 8, 9, 11

MI Mutual Information. 17

MLP Multilayer Perceptron. 22, 81

MR Magnetoresistance. 9

OCSVM One Class Support Vector Machine. 2, 14, 19, 21, 22, 25, 37, 38, 45, 47, 49–52, 55, 56, 59, 61, 62, 64–66, 69–71, 77, 83, 84, 107

PCA Principal Component Analysis. 18

RBF Radial Basis Function. 21, 37, 50

ReLU Rectified Linear Unit. 57, 70, 71, 84

RF Random Forest. 14

RNN Recurrent Neural Network. 18

ROC Receiver Operating Characteristic. 24

SED Scaled Euclidean Distance. 21

SMA Signal Magnitude Area. 82

SMD Scaled Manhattan Distance. 14, 21

SVM Support Vector Machine. 14, 21, 22, 36, 37, 81

TAR True Acceptance Rate. 14, 25

TPR True Positive Rate. 14

UBM Universal Background Model. 20, 21

Colophon

This thesis was typeset with LaTeX 2$_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.