Harnessing the Power of Generative Models for Mobile Continuous and Implicit Authentication

by

Ezzeldin Tahoun

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Mathematics
in Computer Science

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Authenticating a user's identity lies at the heart of securing any information system. A trade off exists currently between user experience and the level of security the system abides by. Using Continuous and Implicit Authentication a user's identity can be verified without any active participation, hence increasing the level of security, given the continuous verification aspect, as well as the user experience, given its implicit nature.

This thesis studies using mobile devices inertial sensors data to identify unique movements and patterns that identify the owner of the device at all times. We implement, and evaluate approaches proposed in related works as well as novel approaches based on a variety of machine learning models, specifically a new kind of Auto Encoder (AE) named Variational Auto Encoder (VAE), relating to the generative models family. We evaluate numerous machine learning models for the anomaly detection or outlier detection case of spotting a malicious user, or an unauthorised entity currently using the smartphone system. We evaluate the results under conditions similar to other works as well as under conditions typically observed in real-world applications. We find that the shallow VAE is the best performer semi-supervised anomaly detector in our evaluations and hence the most suitable for the design proposed.

The thesis concludes with recommendations for the enhancement of the system and the research body dedicated to the domain of Continuous and Implicit Authentication for mobile security.

Keywords: Machine Learning, Generative Models, Continuous Authentication, Implicit Authentication, Artificial Intelligence

Acknowledgements

I would like to thank my supervisor Prof. Urs Hengartner for his support, guidance, and leadership during my time in Waterloo. Also many thanks to Iman Akbari for the friendship, support, proofreading and help during my educational journey in UW. I would like to thank all the people who made this thesis possible, fellow researchers, innovators, and creatives. I would like to specifically thank my grandmother, my family and my friends for being of immense support throughout the journey. I am truly blessed to be surrounded by their love and I am grateful beyond words.

This research was enabled in part by support provided by SciNet (www.scinethpc.ca) and Compute Canada (www.computecanada.ca). We gratefully acknowledge the support of the Waterloo-Huawei Joint Innovation Laboratory for funding this research.

Dedication

This is dedicated to Dr. Adel Tahoun, Dr. Taghrid Ratba, Dr. Taghrid Tahoun, Dr. Ghada Tahoun and to my sweet little Lolo and Momo. I am truly forever in your debt.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Acronym	Meaning
AD	Anomaly Detection
AE	Autoencoder
AP	Average Precision
AAE	Adversarial Autoencoder
AUPRC	Area Under the Precision-Recall Curve
AUROC	Area Under the ROC curve
CAE	Contrastive Autoencoder
DAE	Denoising Autoencoder
DGM	Deep Generative Model
DSVDD	Deep Support Vector Data Description
DSAD	Deep Semi-supervised Anomaly Detection
EBM	Energy Based Model
ELBO	Evidence Lower Bound
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GT	Geometric Transformations
iForest	Isolation Forest
KDE	Kernel Density Estimation
k-NN	k-Nearest Neighbors
kPCA	Kernel Principal Component Analysis
LOF	Local Outlier Factor
LPUE	Learning from Positive and Unlabeled Examples
LSTM	Long short-term memory
MCMC	Markov chain Monte Carlo
MCD	Minimum Covariance Determinant
MVE	Minimum Volume Ellipsoid
OOD	Out-of-distribution
OE	Outlier Exposure

Table 1 continued from previous page

Acronym	Meaning
OC-NN	One-Class Neural Network
OC-SVM	One-Class Support Vector Machine
pPCA	Probabilistic Principal Component Analysis
PCA	Principal Component Analysis
pdf	Probability density function
PSD	Positive semidefinite
RBF	Radial basis function
RKHS	Reproducing Kernel Hilbert Space
rPCA	Robust Principal Component Analysis
SGD	Stochastic Gradient Descent
SGLD	Stochastic Gradient Langevin Dynamics
SSAD	Semi-Supervised Anomaly Detection
SVDD	Support Vector Data Description
VAE	Variational Autoencoder
VQ	Vector Quantization
XAI	Explainable AI
AES	Advanced Encryption Standard
ALPN	Application-Layer Protocol Negotiation
AVG	Average
CNN	Convolutional Neural Network
DASH	Dynamic Adaptive Streaming over HTTP
DL	Deep Learning
DNS	Domain Name System
DT	Decision Tree
GRU	Gated Recurrent Unit
HTTPS	Hypertext Transfer Protocol Secure
IAT	Inter-arrival Time
IDS	Intrusion Detection System

Table 1 continued from previous page

Acronym	Meaning
IP	Internet Protocol
ISP	Internet Service Provider
LSTM	Long Short-term Memory
ML	Machine Learning
MLP	Multi-layer Perceptron
MTU	Maximum Transmission Unit
NAT	Network address translation
NB	Naïve Bayes
NDA	Non-disclosure Agreement
NLP	Natural Language Processing
PCAP	Packet Capture
QUIC	Quick UDP Internet Connections
QoE	Quality of Experience
QoS	Quality of Service
RF	Random Forest
SAE	Stacked Auto-encoder
SMTP	Simple Mail Transfer Protocol
SNI	Server Name Indication
STD	Standard Deviation
SVM	Support Vector Machine
TC	Traffic Classification
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VPN	Virtual Private Network
VoIP	Voice over Internet Protocol

List of Symbols

Symbol	Description
$x^{(i)}$	Each data point is a vector of d dimensions, $x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]$.
x	One data sample from the dataset, $x \in \mathcal{D}$.
x'	The reconstructed version of x.
	The corrupted version of x.
z	The compressed code learned in the bottleneck layer.
$a_j^{(l)}$	The activation function for the j-th neuron in the l-th hidden layer.
$g_\phi(.)$	The encoding function parameterized by ϕ.
$f_\theta(.)$	The decoding function parameterized by θ.
$q_\phi(\mathbf{z}\|\mathbf{x})$	Estimated posterior probability function, also known as probabilistic encoder.
$p_\theta(\mathbf{x}\|\mathbf{z})$	Likelihood of generating true data sample given the latent code, also known as probabilistic decoder.

Chapter 1

Introduction

It is forecasted that there will be 17.72 billion mobile devices worldwide by 2024, and with 5G and 6G technologies on top of their personal computer (PC)-like computing resources, they will be more capable than ever [21]. As mobile devices are growing in numbers and capabilities by the day, they will be the main way humans interact with information technologies, given their portability and similarity to PCs. Nowadays, in our hyper-connected world, mobile devices are closer to the user, as well as their critical access and sensitive data, than any other class of electronics. It is of pivotal importance to properly authenticate (*i.e.*, validate the legitimacy of) every passive or active session of the mobile device in an effective manner to eliminate the risk of breaching the user's and the device's security. This needs to be done within a high level of convenience to the human using the device to mitigate workarounds (*e.g.*, : passwords sticky notes).

A variety of risks arise due to that very portability that makes these devices extremely convenient, and efficient tools for human technology usage, to the extent of outselling PCs for years [22]. These risks include the possibility of the devices getting stolen, lost, or misplaced easily, which would allow unauthorized personnel to run various sophisticated physical and cyber attacks on them, leading to an abundance of mobile security breaches. The impact of a mobile security breach is in most times more severe than that of a PC breach, as mobile devices hold just as many credentials, photos, and data as the PC, but additionally contain calls data, locations information, a popular second factor of authenti-

cation, such as One Time Password (OTP) offline generators or text Short Message Service (SMS) based OTP, used in many services nowadays and more [22].

1.1 Access Management

Authorization systems are mainly constructed around challenges of what the users are, have, do, or know, and more recently something relating to their context or situation [23]. This allows the user to prove their identity to the system and their eligibility and entitlements to certain actions and data. They have been evolving in implementations, but generally, these are the five authentication factors used nowadays and examples of their implementations:

- something the user knows (*e.g.*, password, pattern, PIN, secret question-answer)
- something the user has (*e.g.*, USB token, smart-card, software token, cookie)
- something the user is (*e.g.*, fingerprint, DNA fragment, iris pattern, voice pattern, hand geometry, heart rhythm)
- something the user does (*e.g.*, signature, gesture, handwriting, walk)
- something about the user (*e.g.*, current timezone, current location or position, current date and time, spatio-temporal authentication, reputation or web of trust, Turing test or Captcha to test whether the user is as capable or human as we assume, contextual or situational awareness)

These are the authentication factors mainly utilized, and generally, authentication systems are considered more robust if they use more factors (*i.e.*, Multi-Factor Authentication (MFA)). For instance, most two-factor authentication systems utilize a password, something the user knows, as well as a one-time passphrase sent to the user's phone, something the user has. A benefit of the OTP mechanism is its resiliency to replay attacks, as the

code is regenerated for every use case [23]. The most common of these authentication factors is the user knows authentication factor, which is commonly implemented in a shared secret ID and password fashion across our technological landscape today.

Continuous authentication refers to a mechanism where the authentication factor is being monitored continuously to assess the legitimacy of the user access repeatedly instead of more common one-time authentication systems (*e.g.*, phone only asks for a password when unlocking the screen). The frequency of these challenges or checks is to be minimized to shorten the period where an attacker can slip under the radar undetected, sometimes to a few seconds. Implicit authentication refers to authentication that can occur without the user of the system actively participating or even gaining knowledge of it, hence happening passively under the hood. Implicit authentication is commonly deployed continuously, as well, to check the security posture frequently since it does not have to undermine system convenience in the pursuit of tighter security. Continuous and Implicit Authentication (CIA) systems might be a great authentication factor, next to something the user knows (*e.g.*, PIN, password) for the mobile security use case. We evaluate that further in our work later on in this thesis.

A few questions that arise in our work are: How can we enhance mobile device security using breakthrough technologies like generative models that have appeared recently in the machine learning research domain? Is it possible to find a good balance between convenience and security and address both physical and cyber threats? How can a system continuously and implicitly collect data passively and use unique patterns to identify authorized users? Can we leverage the increasing computational power available to mobile devices and cloud technologies to enable a powerful security system? We take on all of these questions and delve deeper into this work to find a solution.

1.2 Continuous and Implicit Authentication

The basis of our solution, the Continuous and Implicit Authentication system, is the utilization of data streams continuously available on the phone at all times, including nonactive usage, and run it by an anomaly detection module for the sake of detecting an imposter,

conveniently without alerting the user to the process of validating their identity.

Imagine a phone theft occurs, and the phone can identify within seconds that the new holder is an imposter, and encrypts all its data safely, and alert authorities. Alternatively, the original user walks to their smart home or smart car, and everything unlocks automatically because the phone in the user's pocket can identify and authorize the person. This is the objective of this solution and its domain, and our contributions are a step in the right direction, making this objective more feasible, efficient, cheaper, and more probable for the near future.

1.3 Contributions

Our contributions in this work can be summarized as:

- Present a thorough review of motivations, threat models, and use cases in the continuous and implicit authentication domain
- Present a thorough review of existing works
- Present a thorough proposal and review of generative models capabilities for the continuous and implicit authentication use case, especially when using human kinematics
- Evaluate light machine learning models and compared them to the generative model proposed and the state of the art deep learning-based solution
- Find the proposed generative model to be lighter, faster, more accurate, and resilient than all other solutions, including state of the art
- Present a thorough mobile security collaborative architecture proposal for the solution around the proposed outlier detection generative model, in which edge and cloud training, real-time responses, and third party integrations can take place

- Utilized high availability and low privacy and power inertial data available on most smartphones for our work to allow for better convenience and eliminate the need for active usage of the device to enable authentication (non-cooperative and non-intrusive)
- Propose a new experiment setting that mimics real-world scenarios often overlooked by other researchers, which lowered performance metrics for all solutions but kept the generative model at the top
- Run extensive evaluations on the most popular public dataset in the domain with multiple experiment settings and machine learning models
- Implement the proposed solution core generative model and a data collection and real-time streaming application for Android and IOS systems.
- Open-source all processing modules, experiments, implementations, and mobile operating systems to enhance reproducibility and shorten development cycles for other researchers

The purpose of our work in this thesis is to assess the reliability and feasibility of a CIA system that could potentially enhance mobile security while maintaining high levels of privacy and convenience for the users. We aim to deliver maximum convenience hand in hand with security instead of selecting a point on the trade-off, using emerging technologies and techniques. To deliver maximum convenience and privacy within the mobile CIA system context, we limit our data to inertial sensors data (accelerometer, gyroscope, and magnetometer), as they are also continuously available whether the device is being actively used or not. We also limit our work to light machine learning models to perform anomaly or outlier detection evaluations to be trained and deployed on the phone continuously. We document threat models and use cases that have developed recently and affected all mobile device users. We evaluate how a CIA system can work with and collaborate with other third-party security products and processes in place to improve and simplify the overall security posture of the users, more on that in Chapter 2.

A significant component of our work is tailored towards utilizing generative models, more on our approach in Chapter 4, and specifically the new Variational Auto Encoder, as

a fast, reliable, convenient, and light anomaly detector to fit within our CIA system. We evaluate it against other common algorithms in numerous hyperparameter settings combinations across various experiment settings that covered optional deep feature extractions, extra data features, data scalings, and more. We then demonstrate that generative models also outperform other models in experiments that mimicked real-life scenarios instead of lab setups.

We base some of our work and evaluations on the HMOG dataset [24] and the promising results published by Centeno *et al.* [20] and re-implemented by Buech [13]. We elaborate on our extensive evaluations in Chapter 5, where we run more than ten different anomaly and outlier detection algorithms. We propose an experiment setting uncommon in today's literature that mimics real-life use cases and finds that it deeply hinders reported state-of-the-art metrics, commonly reported in the domain, including our generative models-based results. We also document the implementations and engineering we put into the solution for ease of reproducibility in Chapter 6.

We contribute to the relevant research domains, carefully reviewed in the Chapter 3, extensive open-source evaluations, methods to annihilate the need for manual feature engineering other than deep learning, a detailed methodology in architecting the solution, implementation of data collection, and real-time streaming, mobile application, and implementation of the proposed Variational Auto Encoder for the Android and IOS mobile operating systems. To allow other researchers to understand our contributions, we make our results comparable to most existing works by utilizing the most common performance metrics, dataset, and data processing techniques.

Finally, in Chapter 7, we discuss findings, propose questions, and highlight future opportunities to enhance the overall continuous and implicit authentication research domain.

Chapter 2

Background

This chapter explains the domain of our work and describes the concept, threat models incorporated in our design, use cases, contextual awareness, threat response, performance metrics, and data types.

2.1 Concept

Our work investigates the problem of outlier detection to spot unauthorized users of a mobile device implicitly based on how they handle the device at any particular time. By handling the device, the user leaves behind a trace of unique movements to them, based on the idiosyncrasy of their habits, hand measurements, and natural range of motion while typing or using the device. The movements are captured by the inertial phone sensors: accelerometer, gyroscope, and magnetometer. Assuming that a limited number of users use the smartphone, most commonly, the idea is to trigger when the activity in the session (any particular minute in our case) seems to be off the standard behavior. Standard behavior in this context is defined by the owner or authorized user's behavior saved during a pre-determined duration, where multiple scenarios are performed and measured.

Although the most commonly used biometrics are *physiological* (like face, fingerprint, hand, iris, DNA), we believe those that are *behavioral* (like keystroke, signature, and voice)

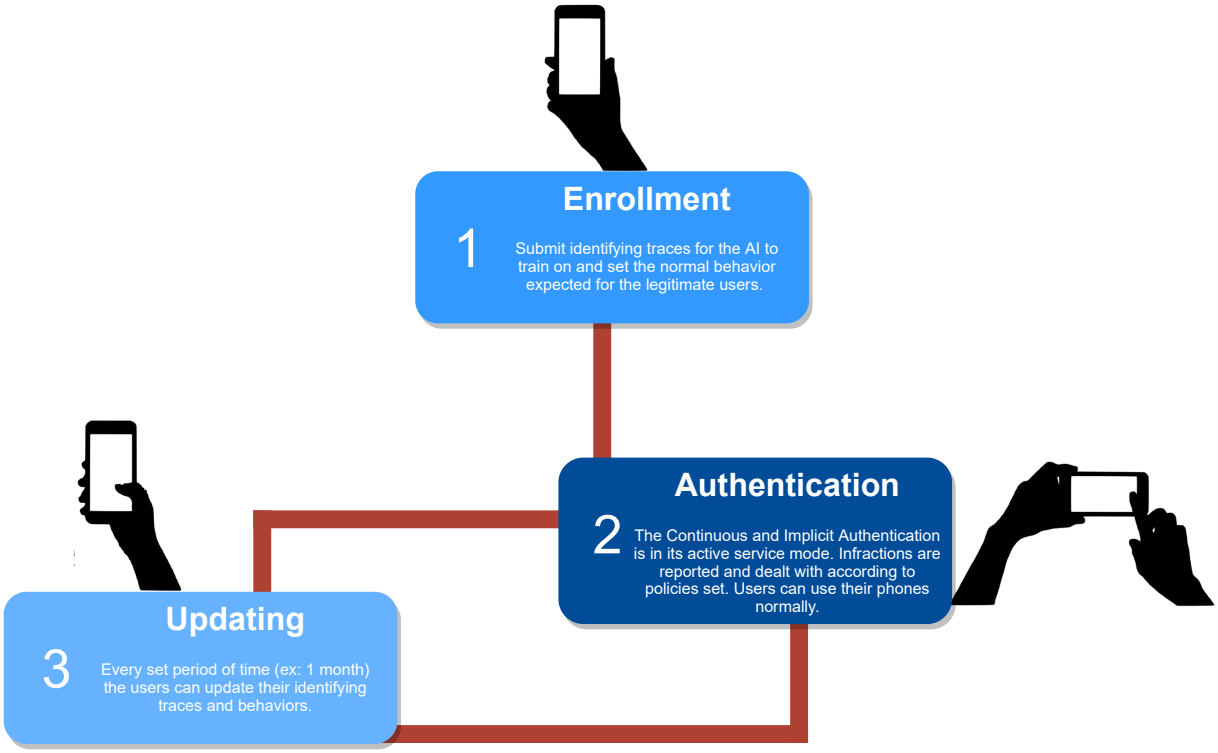

Figure 2.1: Logical stages of the system in deployment.

are more fitting to Continuous and Implicit Authentication and hence are more resilient to the owner being forced or tricked to hand over their authenticated device.

In a real-world deployment of this solution, the following three fundamental stages need to be considered: (i) enrollment, (ii) authentication (iii) updating. These stages are explained in more detail below and can be seen in Figure 2.1.

Enrollment is generally a very simple scenario involving inertial data collection where the user is presented with certain activities to conduct (*e.g.*, using the phone while sitting, using the phone while walking, using the phone while standing, using the phone while running, using the phone while biking, and using the phone while driving). As the user acts out the activities, the data from the inertial sensors are ingested and streamed to the AI module to be utilized as training data for this user's profile. The users can register as many profiles as they wish, although commonly smartphones have one legitimate user.

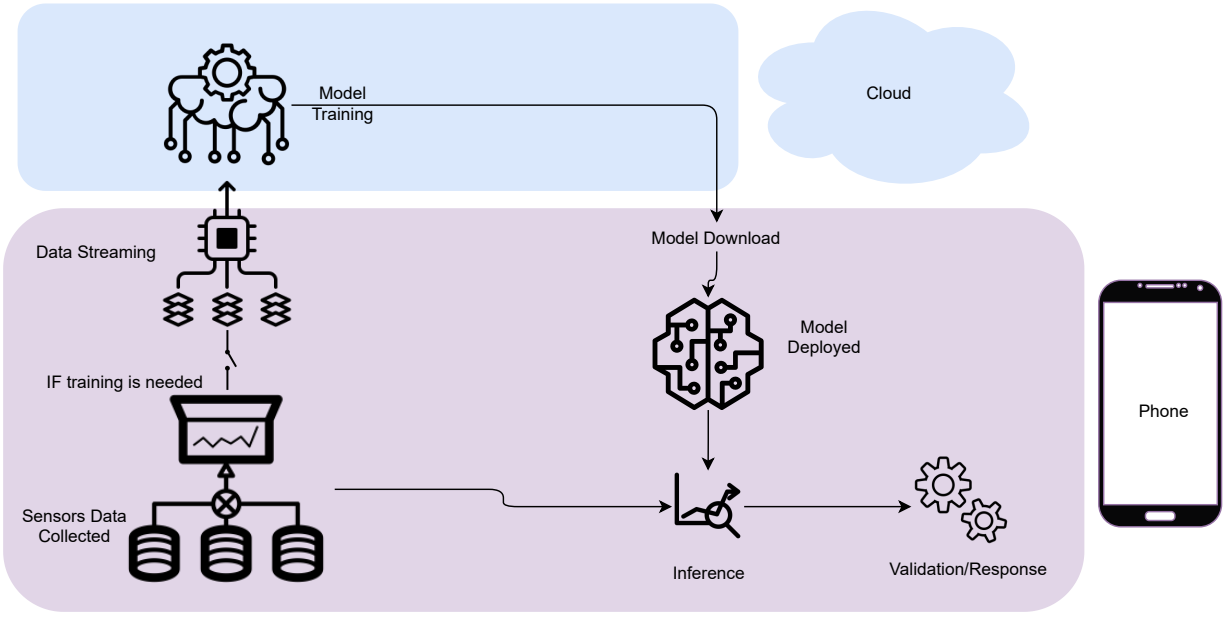

Figure 2.2: System architecture, showing the training taking place on a cloud resource.

The model is internally evaluated against a pre-determined set of malicious data and a threshold is set for acceptance in a functional testing sub-stage. If the model passes, the model is ready to get out of enrollment and graduate to the next stage. If the model does not pass, then the enrollment is determined not good enough and the user is asked to redo the activities.

The next stage of the deployment is authentication. This is when the core activities of the solution take place, which is detecting deviations from normal behavior and responding according to a set policy that sets sensitivity thresholds. Responses might include notifying local authorities or hiding sensitive apps and data from the current session's user until the user performs a secret bypass pattern to restore the device to its normal state. This stage is where the solution is designed to be most of the deployment life cycle. Suppose the user is falsely classified as malicious, and their sensitive data and apps are being hidden from view. In this case, the user can either initiate an updating stage transition to enhance the model's performance by proving they are legitimate users by entering a special secret bypass pattern of movements or taps and then updating the profile by acting out new training

scenarios, or the user can enter the secret bypass pattern to make an exception and cancel responses for the current session. The bypass pattern can take the form of any set of events. For instance, tapping the top right corner of the screen five times and then tapping the bottom left corner two times or both corners simultaneously three times. The secret bypass pattern allows the user to have a way to authenticate if the system has reasonable suspicion against the current session as a fallback. We advise against utilizing the secret bypass pattern frequently, and hence the system can enforce a series of authentication challenges and an update stage transition if the secret bypass pattern is utilized with high frequency and is abused.

Occasionally, the updating stage will be initiated as scheduled or in an active learning setting via immediate feedback. It is recommended to schedule monthly or quarterly updating sessions for the system to learn new behaviors that the users have adopted, consciously or subconsciously. The sessions are similar to enrollment from user experience, but only the deviations are recorded and reflect certain changes on the user profile normal behavior model. As this stage might present an attack surface, this stage is only enabled after other authentication factors are passed (*e.g.*, password, PIN, face recognition, fingerprint, keystroke dynamics). Immediate feedback can be set to achieve an active learning setup where the model can get feedback on its detections in near real-time by the user just performing a secret bypass pattern when falsely classified as an anomaly.

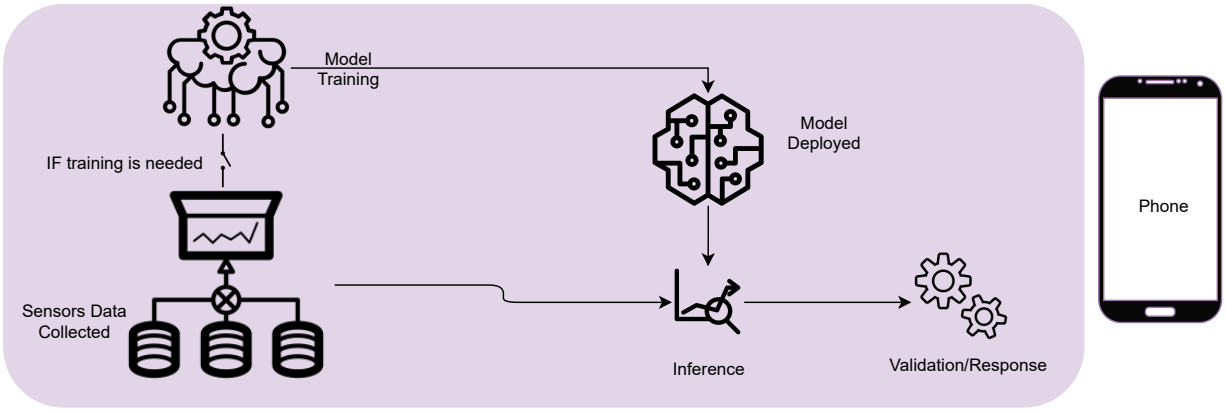

Figure 2.3: System architecture, showing the training taking place on the smartphone.

It is important to note that during our design, we consider two cases, as per Figure 2.2 and Figure 2.3, for model training (1) at edge training and (2) at cloud training, but we always fixed inference at the edge since the threat models indicated the possibility of losing connection to the internet at critical times, stressing the need for an always-on and always alert continuous and implicit authentication system. The edge of the phone, in this case, might or might not run training, depending on preference, compatibility, and capabilities. The compatibility and capabilities explored by us are further detailed in the Chapter 6.

2.2 Threat Models

In this section, we build on the concept put forward in the previous section, and quantify the threat models endangering mobile users' security today.

The four threat models we are considering feature the device being used by someone else other than the owner. The ability to detect an unauthorized user implicitly and continuously allows us to offer protection against the loss of sensitive and private data. The owner or authorized users are the devices able to choose which data or device functionalities are considered sensitive.

The scenario is defined as the threat actor gaining *physical* access to the device, as well as bypassing the initial authentication system (*e.g.*, PIN, fingerprint, Face ID) by any means. This may include previous possession, negligence, software bugs, brute-force attacks, dictionary attacks, or even acquiring the device and credentials by force.

The sensitive information and access available to the unauthorized user in this situation includes but is not limited to:

- Personally identifiable information (PII): *e.g.*, address, date of birth, name, initials, signature, pictures
- Critical information: *e.g.*, passwords in a password manager, intellectual property, trade secrets, personal secrets, strategy, business plans
- Critical access: *e.g.*, valuable assets, digital currency, bank accounts, infrastructure, email accounts, social media accounts, private keys

The threat model can take a few different forms and variants. In Table 2.1, the following specific threat models are described: threat model (A), Compromised authentication factor, is where an unauthorized user is able to use the correct PIN, password, pattern, or biometrics to log in due to prior knowledge (*e.g.*, by-shoulder surfing the legitimate user previously logging in). Threat model (B), Handover/Sharing, is when the authorized user hands over their device, unlocked, to an authority figure, friend, or spouse. Threat

Threat Models			
Label	**Name**	**Description**	**Example**
A	Compromised	Compromised authentication factor leading to successful bypass of other authentication factors by a threat actor	Unauthorized user is able to enter the correct PIN, password, or pattern to login to the device and act maliciously inheriting the privileges of the compromised credential's user.
B	Handover	Authorized user handing over or sharing their device with someone unauthorized	Owner giving their unlocked device to an authority figure, attacker forcing login, friend, or spouse. The unauthorized actor is able to act maliciously with the same privileges the authorized user has.
C	Unattended	Authorized user leaves phone unlocked and unattended, resulting in theft or unauthorized access	Owner forgets device on the table, and leaves the room. A threat actor picks it up and is able to act maliciously with the same privileges the authorized user has.
D	Locked device loss	Locked device is lost	Owner gets pick-pocketed while walking. A threat actor has access to the device in its locked state.

Table 2.1: Threat models considered

model (C) is when an authorized user forgets their device, is unlocked, on a table, and leaves the room. A threat actor then picks it up and is able to act maliciously with the same privileges the authorized user has. Lastly, a fourth and important threat model is (D), and it does not depend on the active usage of the phone at all. Instead, it focuses on the passive phone authentication state, meaning that the device loss can be detected when the phone is unlocked by capturing moves the phone registers while locked in the hands, pockets, or bag of someone else.

2.2.1 Use Cases Scenarios

Compromised Authentication System

Threat model (A) (*cf.*, Table 2.1) is when a phone is physically with someone else that has compromised the authentication procedure. In this case, they do not need you to leave the phone unlocked behind or give it to them unlocked, as they can unlock it as many times as they want by knowing how to bypass the primary authentication. That is, by obtaining the means of submitting the correct answer to the authentication system (*e.g.*, fingerprint, password, PIN, pattern, biometrics) or being able to exploit a software vulnerability to bypass the authentication system used to protect the phone.

In the case described above, the only security system left to protect the phone is one that can detect deviating behavior from the normal (associated with the real owner) and be discrete, implicit, and continuous in its detecting fashion, as well as being subtle in its response to that detection. We do not tackle the response procedure in this work very deeply, but we are very interested in detecting the threat models as accurately as possible and minimizing the false positives to ensure convenience is not heavily sacrificed for security. Cases where we can see threat model (A) play are when a colleague shoulder surfs you entering the PIN and using that later when you are not around to access sensitive information. Within a minute, the system can pick up the new behavior solely based on inertial sensors data and the way the co-worker handles the phone and subtly disable access to important data.

Handing over an Authenticated Device

There are many examples around the second threat model (B), specifically by handovers to government officials, which has spiked in the past few years. In some of these cases, customs and border officers would ask for the smartphone of the people going through the border or checkpoints to view their receipts, purchases, contacts, text messages, social media, or emails. The digital search is conducted without a need for a warrant, much like the physical luggage search, during a border crossing.

The Canadian Border Services Agency (CBSA) says it is allowed to search cell phones, smartphones, computers, tablets, removable media, drives, cameras, smartwatches, and any other digital device. The search is said to be to address concerns for reasons as broad as the admissibility of goods, identity confirmation, breaking laws or regulations, among others. The CBSA says that the officer will ask the person to write their password on a piece of paper and that they are obligated to provide the password when asked. The CBSA goes on and indicates that failure to grant access to the device will result in detention of the device under section 101 of the Customs Act or seizure of the digital device under subsection 140(1) of the Immigration and Refugee Protection Act. The use case is further documented and reads that the office will put the phone on airplane mode, hence disabling any internet-based security systems, meaning that a security system to detect this action needs to be able to process the data locally at all times. [25]

The United States Department of Homeland Security (DHS) Customs and Border Protection (CBP) also has customs and procedures involving the search of all electronic devices at any border crossing. CBP has published a directive [26] on electronic devices border searches in which it is indicated that the authorities are allowed to search devices such as portable computers, tablets, disks, drives, tapes, mobile phones, and other communication devices, cameras, music players, and other media players. The directive says the officers will disable the network connectivity and will conduct a search, with or without suspicion of their contents. This behavior can be detected if a security system that performs implicit and continuous authentication is running locally without the need for an internet connection.

In this use case, once the behavior is detected by handling the phone differently for one

minute, after which the operating system can hide certain data and applications automatically until a specific secret authorization step is performed, for example, tapping the top right corner of the screen five times.

Threat model (B) can also take the form of a forced login by a criminal motivated to access and transfer financial assets such as digital currencies or valuable business or political information they can sell to news media, competitors, and traders. Figure 2.4 shows how this system will be able to hide data and apps, which might even save human lives in certain circumstances. Even if the phone was unlocked and authenticated, anomalous user behavior before and during the primary authentication is noticed by the device. As we mention later in this chapter, context can be very useful, as it gives our module information about location or time. For instance, the system can be set to hide access to your Bitcoin wallet when the user is passing a small isolated street during a vacation abroad. Moreover, the system can hide access to the user's confidential emails, files, and apps when passing through border control.

Threat model (B) also happens in many social scenarios when you would like a friend, spouse, or colleague to view your screen momentarily, and they decide to take the liberty and use your phone in their possession for something else.

Misplacing an Authenticated Device

Threat model (C) has a variety of use cases, most popular of which is when a user leaves the phone for a nearby opportunistic threat actor. The phone can be left behind intentionally or unintentionally, but it is exposed to theft or unauthorized access from nearby individuals. In this use case, for instance, a spouse or friend can take control of the phone when it is left unlocked on a coffee table and explore the contents, either maliciously or not. Within a minute, the phone detects the new user and triggers an action plan like hiding data or applications or simply forcing a reboot or lockout.

In case of theft, the use case would be easily detected and differentiated from the spouse use case by context, and using heuristics or rules sets of common post-theft patterns (*e.g.*, throwing SIM card away, logging out of accounts), which is something our solution can be expanded to detect. One example of a rule that can be used is that if the user is logged

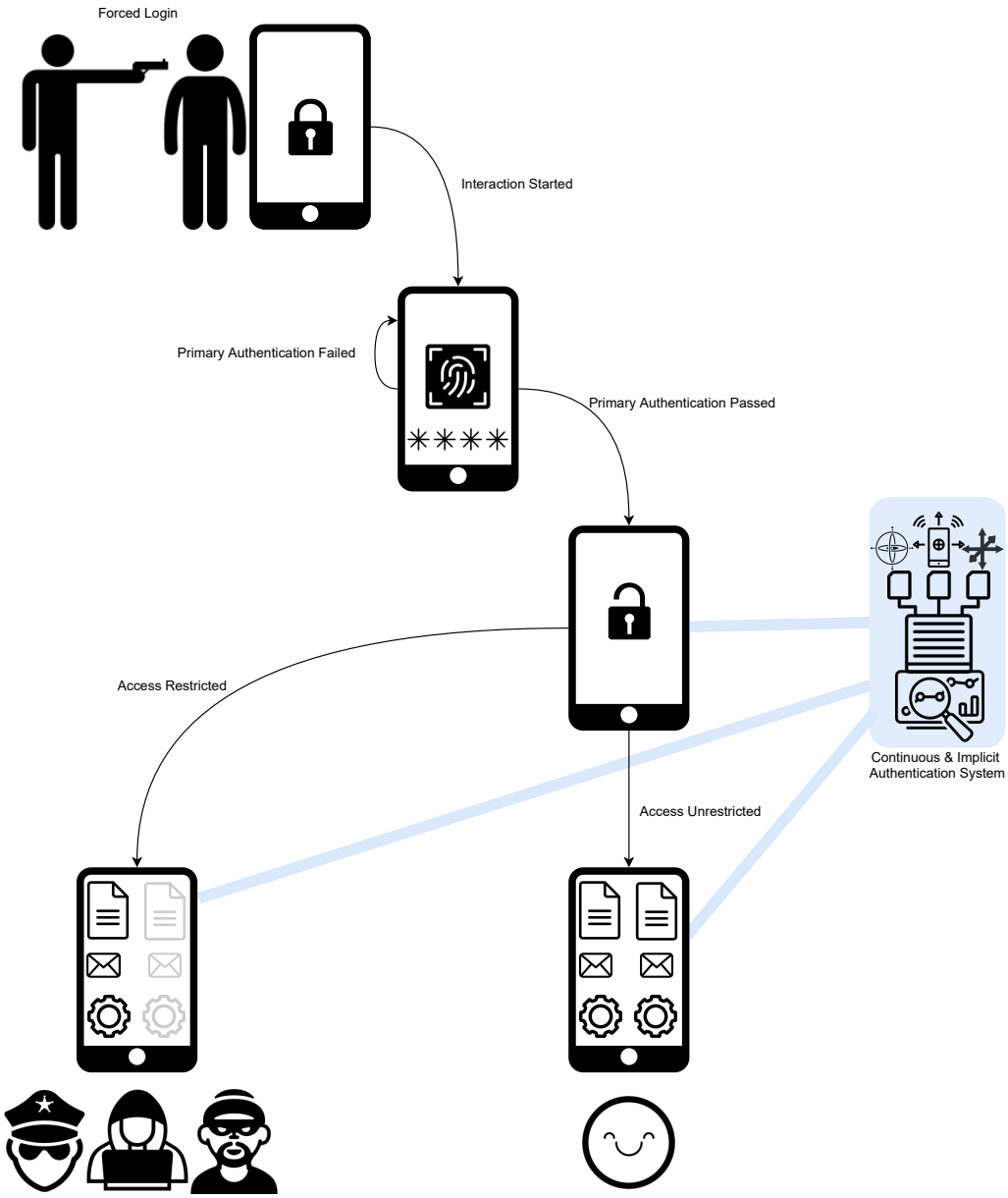

Figure 2.4: An example forced login scenario, showing how the proposed system can hide data, and apps on the phone to prevent losses and damages.

out of their Google or Apple account and the SIM card is removed, trigger theft procedure, assuming different inertial behavior is detected as well.

Locked Device Loss

Threat model (D) focuses on the passive use of the phone to alert the owner and their enterprise of a lost or stolen device. This can work hand-in-hand with the "find my device" capability that most modern smartphones today have. Notice that by having the system running while the phone is not used actively, the system can act as an authenticator itself. For instance, it can go as far as sending an authentication token for integrated systems like smart cars, smart gates, or smart homes.

We consider all four threat models defined and aim to design a solution to protect the user in all of these use cases.

2.3 Contextual Awareness

In this section, we build on the concept and threat models put forward in the previous sections, and propose the utilization of contextual awareness to better the understanding and sense of the environment and the situation at any given moment before taking action against a detected threat.

The Continuous and Implicit Authentication System, also referred to as the solution can interact with other modules that can provide further contextual and situational awareness to the model. We have not been able to test that in our evaluations for the lack of an appropriate dataset. However, there are strong reasons to believe that situational awareness can play a significant role in most use cases.

For instance, we can simply define context as the location of the device. If the device is at a border or a police station, the system can be set to go in high alert mode and disable the profile updating stage functionality or disable high-profile functionalities temporarily. The system can only allow low-profile data and applications or display customized mock interfaces and pages to unauthorized users. This is useful for preventing suspicion by the

threat actor, who can accuse the legitimate user of non-compliance in the scenario where they are law enforcement authorities.

Context can also be set not only by geographical location but also by neighboring devices or beacons that give our solution a feeling of trust and security when the device is at a virtual home, that can be pre-set as a safe zone [27]. The safe zone can be identified based on certain smart home or Internet of Things enabled electronics such as TVs, displays, speakers, virtual assistants, power outlets, cameras, doorbells, baby monitors, home security systems, thermostats, lighting systems, kitchen appliances, home cleaning electronics, fitness devices, lawnmowers, sprinkling systems, smart luggage, or smart car. Alternatively, detection of the safe zone can be based on the electronics on the user at most times like personal electronics such as tablets or laptops or smart gadgets such as smartwatch watches, fitness tracker, or headphones. Devices such as telehealth and telemedicine smart wearables, pacemakers, insulin pumps, blood sugar monitors, shoe inserts, necklaces, ECG and EEG monitors, RFID implants, or smart tattoos are also good indicators of a safe environment.

Many mobile and WiFi sensing works have been able to show successful results in indoor localization, people counting [28], activity classification, health monitoring, humidity estimation, sign language recognition, metal detection, smoking detection, traffic monitoring, sleep detection, gesture recognition, emotion recognition, attention monitoring, keystrokes recognition, drawing in the air, and more [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 32].

We envision integrations that enable the detection of a forced login by also checking for rising skin temperature, rising heartbeat rate, or negative emotions. The integrations open up a wide range of possibilities for the use cases in Table 2.1. These indicators can be leveraged via the use of other features, gadgets, and models.

An integration with an Enterprise Mobile Management (EMM) system will also prove pivotal, as notable alerts can be forwarded to the EMM and hence can reach the corporate Security Operations Center (SOC), correlated and analyzed by security analysts and via a Security Information and Event Management (SIEM) as well as responded to in the appropriate manner. EMM systems are able to integrate with a Virtual Private Network

(VPN) and mobile threat defense systems, to name a few, and are commonly tasked with managing and protecting mobile devices in an increasingly mobile enterprise setting [44]. This means that an EMM can block a device from accessing the corporate VPN if not compliant with certain risk scores, for instance. Our system can send a token over the network confirming behavioral authorization to other devices (such as a smart, safe box, for instance), not just other software that exists on the same device.

2.4 Threat Response

In this section, we build on the concept, threat models, and contextual awareness put forward in the previous sections, and materialize what the response could be, the vehicle it would use to deliver its impact, and how it would take place in our solution.

NIST has recommended many responses to different threat models, as well as other researchers in the field [44]. We will assume the response can be easily configured by the user, or Mobile Device Management (MDM), EMM, or an organization SOC, to either restrict access to certain functions and data or to completely lock the device until further notice by secure explicit authentication. The latter can take the forms of unlocking from a cloud account, an email, OTP, a wired enabled hardware authentication factor, a wireless hardware authentication factor using technologies like Bluetooth, NFC, RFID, WiFi, a secret master key, or even secret sharing like the one proposed by Atwater and Goldberg [45].

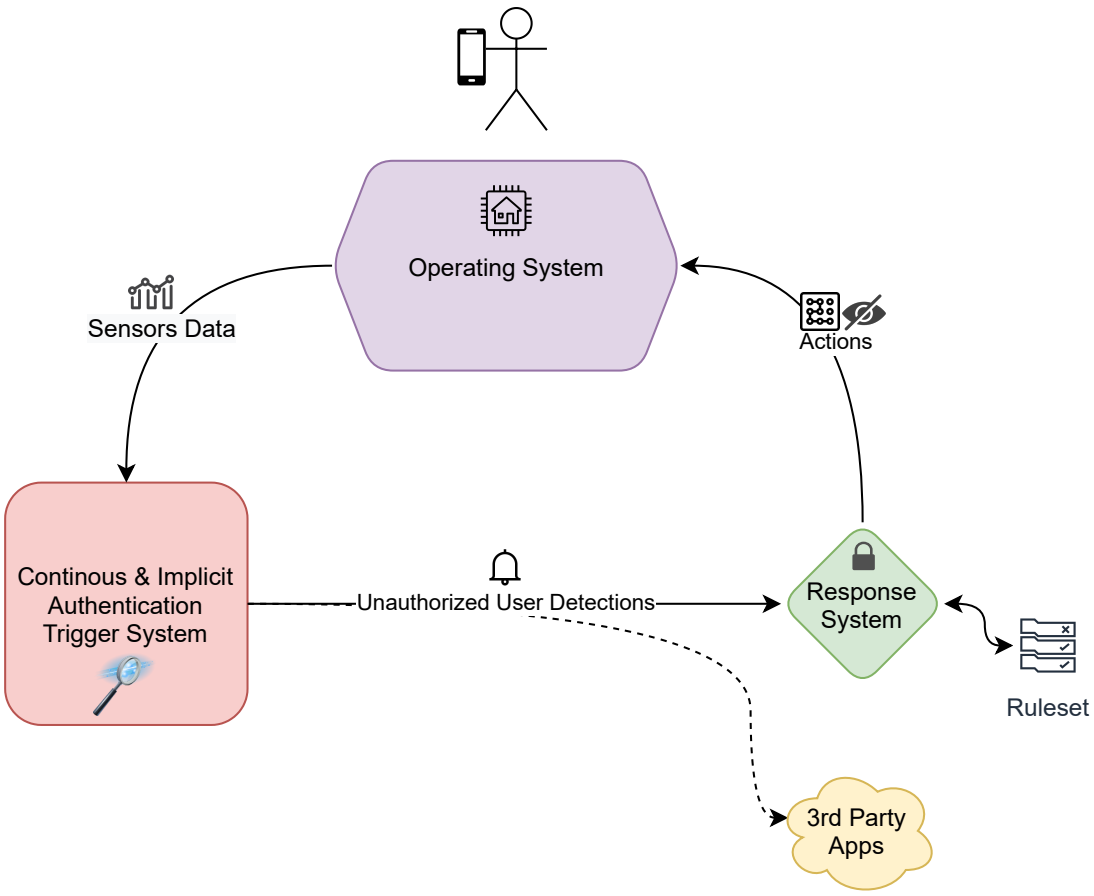

Figure 2.5: The information and actions flow between the continuous and implicit authentication system, the operating system, third party applications, and the response system.

Varying possible responses to a "not owner" signal from the Continuous and Implicit Authentication system are presented below.

- Lock and encrypt the device and immediately, require two authenticators (*e.g.*, password, PIN, pattern, face, fingerprint, token-based, network-based, domain-based, digital certificate) to unlock and decrypt the device.

- Wipe the device after a certain amount of time or failed trials to authenticate, suspecting the device being lost, stolen, or accessed by unauthorized actors to mitigate the risk of confidential data recovery
- Revocation of enterprise access
- Removal of certain apps, files, or data (*e.g.*, emails, SMS text messages, private keys)

We believe a policy or a playbook can be leveraged here to configure automated responses per user requirement to a wide range of threat models. In Security Orchestration, Automation, and Response (SOAR) products, playbooks are commonly used to respond to threats in the security monitoring and response industry. For instance, the network administrators are able to define a playbook that is triggered by a certain anti-virus alert, to block the IP addresses involved or disable certain machines temporarily. As previously mentioned, it is highly advised to integrate this system with detection tools for validation and to maximize benefit across different apps, platforms, and security frameworks. A multi-layered approach is critical to a successful security program [46]. Occasional testing is recommended for these policies, and if it is found that there is a breach due to a certain policy, it is safe to assume it is weak or outdated and hence should be reviewed and updated.

Sample playbook for response to detected threats could be defined be as follows:

- Our solution could act as a layer in an MFA mechanism to thwart against threat model (A). In this case, it can trigger an extra authentication step, such as OTP, secret question, or biometrics
- It would also be able to detect a handover threat model (B), in which case it would trigger a pre-set configuration such as restrict access to certain applications, settings, installs, or sensitive data
- In threat model (C), the phone can beep or send an alert to the smartwatch or wearable gadget assumed to be with the owner. The phone can also be set to reboot or restrict access to certain information and applications, as well.

- In threat model (D), the phone can decide to wipe its backed up sensitive contents completely upon verification that this is a phone theft situation.

We envision the Continuous and Implicit Authentication (CIA) trigger system, seen in Figure 2.5 to be able to share its detections with other security systems that can benefit from the threat scoring of particular activity time stretches. The 3rd party apps would be able to use the data to correlate it with suspicious activities and compound the threat scoring to come up with a decision about the transactions or access approved or utilized during the time period. For instance, a banking app can utilize the scoring from the CIA module based on the inertial sensors in the smartphone and the internal model of the user of the phone. The authentication state can change and result in further authentication factors for the user, such as email OTP. Standards like FIDO2 [47] and PIV [48] can guide the interactions between the module and web applications and other devices. The integrations will allow it to expand the possible responses beyond the phone itself.

2.5 Data Types

In this section, we build on the concept, threat models, contextual awareness, and threat response put forward in the previous sections, and discuss what data types we aim to utilize in our solution and why.

Off-the-shelf smartphones in the market today come packed with sensors that acquire a variety of data points around the device at any given point. Below is a list of a few sensors that are typically available:

- accelerometer
- gyroscope
- magnetometer / compass
- barometer
- ambient light sensor / photometer
- proximity sensor
- battery temperature sensor / thermometer
- touchscreen sensors
- biometric (facial, iris/retina and fingerprint recognition)
- heart rate sensor
- air humidity sensor
- camera
- microphone
- GPS

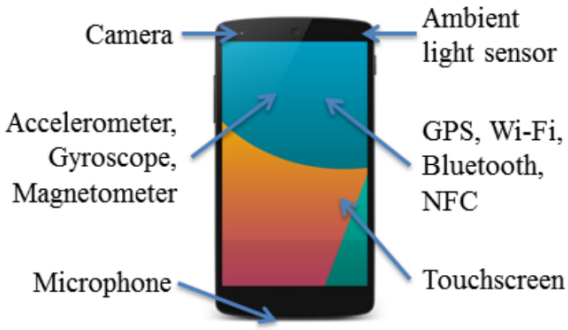

Figure 2.6: Some of the sensors present on many phones. Source: [1]

An accelerometer measures the device's acceleration. A gyroscope measures the device's angular velocity. A magnetometer measures the effect of the magnetic field to cope with the device's orientation. A barometer measures changes in the atmospheric pressure to identify elevation. The ambient light sensor detects the surrounding light, allowing for reconfiguring the screen brightness automatically. Proximity sensors measure the distance of surrounding objects. The battery temperature sensor generates data that can be harvested to identify the temperature of the phone as well as the surroundings.

Touchscreen sensors measure touches and taps to identify gestures, swipes, multi-touch, and clicking. The biometric recognition sensors allow for identity authorization using something the users are, as opposed to know or have, as we discussed in the previous section [4].

Sensors like heart rate and air humidity help better fitness applications. The camera allows for visible light collection in the photo or video formats, the microphone allows for registering and collecting all sounds in the surroundings, and GPS allows the device to utilize a network of satellites to triangulate the location on earth, where it currently exists.

Each of these sensors can help us identify an environment, user, pattern, or activity if we analyze the data and utilize data science technologies like data mining or outlier detection.

Not all sensors were created equal, however, and they incur different power costs, operating system permissions, as well as data relevancy, noise to signal ratio, and quality. For

(a) (b)

Figure 2.7: (a) The Wii™ remote with the MotionPlus™ attachment (b) The Apple IPhone™ 6.

instance, the most expensive three sensors from a power and permissions point of view are camera, microphone, and GPS [4].

Camera, microphone, bio-metrics, ambient light sensor, barometer, temperature, and proximity, are all examples of multimedia sensors. Accelerometer, magnetometer, and gyroscope are considered motion or inertial sensors as they give us the best data on rotational and accelerations forces across all axes. They have been used in devices ranging from the WII controller Figure 2.7 to capture movements of players, to the Apple iPhone 6 Figure 2.7 to capture orientation, movement, and more. Any device that is equipped with these inertial sensors probably is fitted with an electronic component commonly referred to as an Inertial Measurement Unit or IMU, more on which at the end of this chapter.

These analog inertial sensors do not depend on active usage of the smartphone, have a low power fingerprint, and are not protected behind permission walls like other sensors such as camera or microphone, and they might have a huge potential in fingerprinting how a user moves the device while using it, if we harvest and analyze their data properly [4].

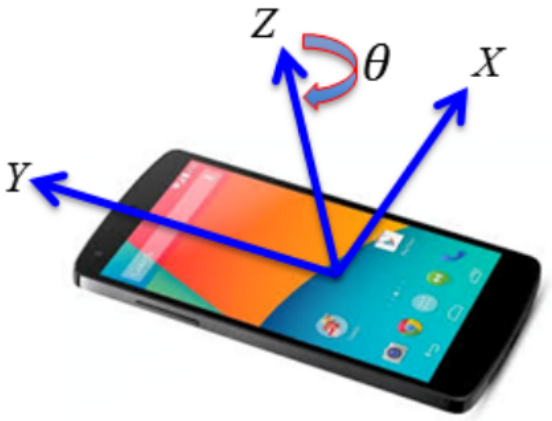

Figure 2.8: Axes of sensors on many phones. Source: [1]

2.5.1 Inertial Sensors

Let's start by understanding how these sensors work before we take a deep dive into analyzing their data to potentially improve authentication systems for mobile devices.

Many accelerometers and gyroscopes are based on microelectromechanical system (MEMS) technology; their components are inexpensive, low-power, light-weight, minuscule, and of good accuracy and startup times [13, 2].

Frames

To understand the measurements by these sensors we need to define a few coordinate frames, namingly:

- body frame (b-frame)
- inertial frame (i-frame)
- navigation frame (n-frame)
- earth frame (e-frame)

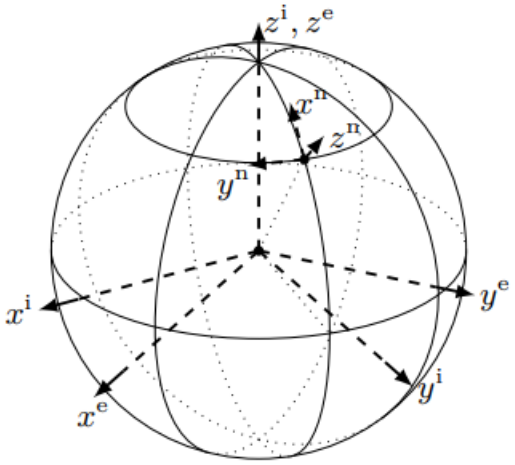

Figure 2.9: Coordinate frames, n-frame at a certain location on earth and the e-frame rotating with earth and the i-frame. Source: [2]

The b-frame is a coordinate frame that is originated at the center of the accelerometer triad, with axes aligned to the casing, and all the inertial measurements are resolved in this frame. The I-frame is, simply, a stationary frame originated at the center of the earth with axes aligned with respect to the stars. The n-frame is a local geographic frame, and we usually observe the orientation and position of the b-frame with respect to this frame. The e-frame coincides with the I-frame but with axes fixed with respect to earth, meaning it rotates with earth.

Accelerometer

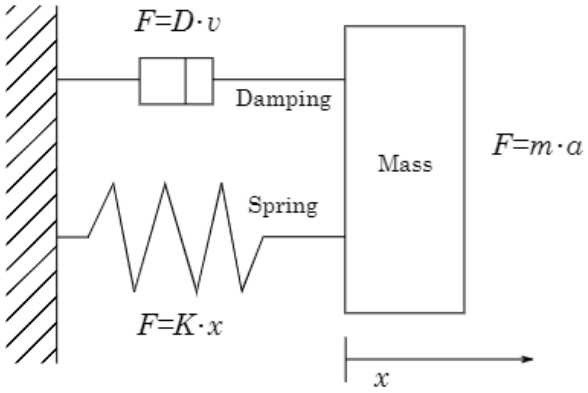

Figure 2.10: Schematic of an accelerometer. Source: [3]

The accelerometer measures the external specific force f, consisting of both of the sensor's acceleration and the earth's gravity, acting on the sensor in the body frame b [2]. Acceleration can be described as $dv(t)/dt$, where $dv(t)$ is the specific change in velocity over time t and dt is the time of that velocity change. If we factor in that velocity is nothing but the derivation of distance over time, we end up with $d^2x(t)/dt^2$.

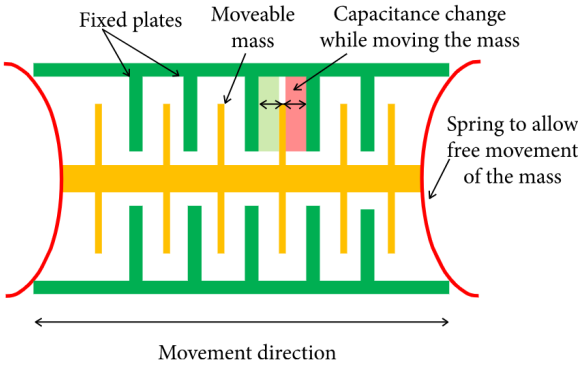

Figure 2.11: Accelerometer structure. Source: [4]

A three-axis accelerometer allows us to cover three-dimensional acceleration, as a single accelerometer usually detects acceleration on one axis. Measuring the deflection and time

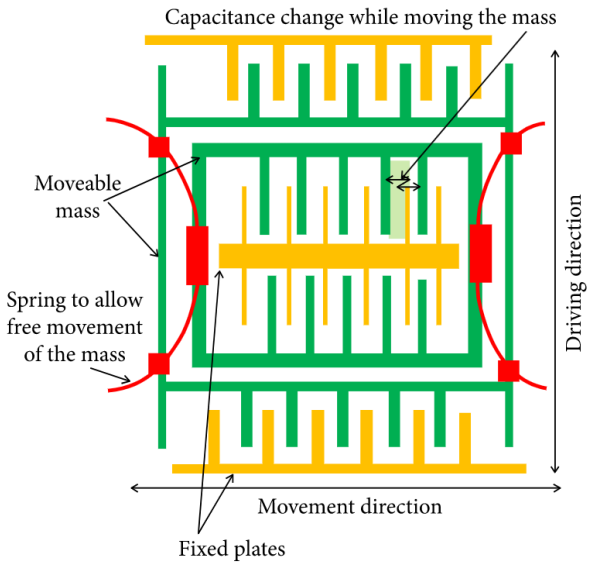

Figure 2.12: Gyroscope structure. Source: [4]

of a spring-mounted mass, as seen in Figure 2.10 as well as Figure 2.11 during the exposure to an accelerating force allows the accelerometer to compute these values [3, 4].

Gyroscope

The gyroscope measures the angular measure angular displacement per time of the body frame with respect to the inertial frame, expressed in the body frame. Three-axis gyroscopes allow us to cover three-dimensional angular velocity, as a single gyroscope usually measures angular velocity on one axis.

As Coriolis vibratory gyroscopes (CVGs) are based on vibrating masses that get deflected by the Coriolis force when it is exposed to a torque, as seen in Figure 2.13 and Figure 2.12. The measured deflection is used to calculate the movement angular velocity. Gyroscopes are among the sensors with the highest power consumption due to the constant high-frequency vibrations [5], and hence we will evaluate our solution without the data from the gyroscope to see what is the observed effect on performance.

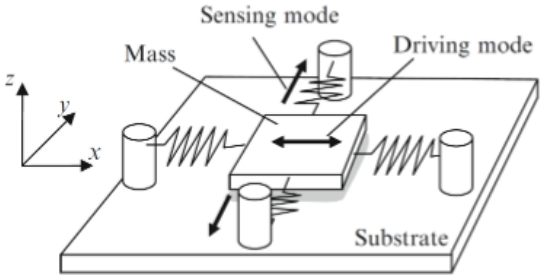

Figure 2.13: Schematic of a vibrating gyroscope. Source: [5]

Magnetometer

The magnetometer measures the strength of the magnetic field. This allows the device to obtain its absolute direction related to the greater earth's geomagnetic field [4]. Some of them are built to utilize measurements of detected voltage across a metallic element.

One of the most-known implementations for the magnetometer sensor is referred to as the search-coil magnetometer [6]. It is based on Faraday's induction law and hosts a coil wound around a magnetic iron core. The concept is based on the observation of a measurable voltage change proportional to the rate of change that is observed at the leads induced when the magnetic flux through the coiled conductor gets altered in any manner. Its schematic is to be seen in Figure 2.14.

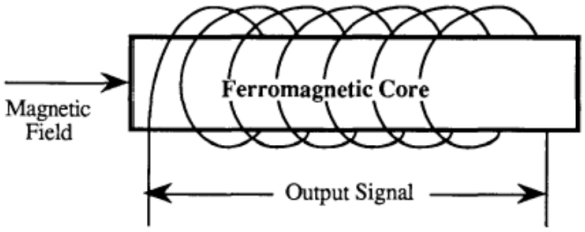

Figure 2.14: Schematic a search-coil magnetometer. Source: [6]

Most commonly, the types seen in smart devices today are Anisotropic Magnetoresistance (AMR) or Giant Magnetoresistance (GMR), as they are small, cheap, and low-power. As is the case with the accelerometer and gyroscope, the three axes are needed in all use

cases, and hence a magnetometer is always built with the ability to read magnetic fields on all three axes to cover all possible orientations.

Considerations

Magnetometers are not usually standalone modules in today's devices. The Inertial Measurement Unit (IMU) is a package that commonly holds a three axes magnetometer, accelerometer, and gyroscope. They are commonly fitted with self-calibration features and are commonly available in devices today, as they are cheap and extremely helpful in many applications like gesture recognition and image stabilization [13].

Inaccuracies find their way into such data streams due to calibration errors, drift, or noise. Some of the manufacturers of such electronic components fit solutions to handle the problem of noise, using noise reduction technologies. These solutions can go as far as making new high-level fusion sensors data such as orientation sensors that can utilize all three accelerometers, magnetometer, and gyroscope processed and noise-reduced data streams to enable applications to use more robust data points. Such fusion sensors (*e.g.*, relative orientation sensor, absolute orientation sensor, geomagnetic orientation sensor, gravity sensor, linear acceleration sensor) can also be calculated by third-party applications, using the underlying physical sensors discussed: magnetometer, accelerometer, and gyroscope, to account for their use cases (*e.g.*, gaming, augmented reality) [49].

2.5.2 Security Issues with Mobile Sensing

Despite their effectiveness in user behavior profiling, the leakage of mobile sensor data can put the user's privacy and security in jeopardy. For instance, malware can target a specific group of people with a common trait or behavior, which can be identified using their motion sensor data. A more alarming example is cyber-criminals purchasing the user's sensor data from a data broker [50] or a third-party mobile application developer, in order to locate an individual within their house [30], detect their current activity [51], identify the number of people living with the individual [52], and detect their sleeping patterns [53, 54] [55].

Thus, we have to be cautious with the data generated by our devices. Harvesting these valuable data sources can reveal too much information about the user. Hence, we limit the amount of data needed to one sensor data, but we also understand that the data from this sensor alone may be sufficient to obtain information about a user's location, body features, gender, age, activities, health condition, personality traits, and emotional state. The data can even be used to uniquely identify a user based on their biometric movement patterns and to reconstruct sequences of text entered into a device, including passwords, for instance [56].

In Chapter 7, we propose the exploration of our work's intersection with differential privacy in order to protect accelerometer data in transit. Additionally, we develop our solution architecture (*cf.*, Chapter 2) to allow for mobile sensor data to never leave the phone. The training for our method can occur at the device, thanks to our light machine learning shallow generative neural network proposed in Chapter 4, adding a layer of security and privacy to the solution.

2.6 Performance Metrics

In this section, we build on the concept, threat models, contextual awareness, threat response, and data types put forward in the previous sections and discuss how we would interpret and score our results to highlight certain properties of the solution's models. Many metrics exist in this domain, but few are adopted by the research community in continuous authentication literature.

There are many authentication systems that use machine learning. However, there is yet to be a consistent approach and metric for reporting performance. Evaluation is of pivotal importance to machine learning problems. Choosing the metrics to use is very important since a solution will be as good as its performance metric.

Recent works have shown that maximum accuracy (ACC), equal error rate (EER) as well, as Area Under the Receiver operator characteristic curve (ROC) curve (AUROC) hide the details of the inherent trade-offs a system must make when implemented, and hence are all, although very abundant in the body of research, inherently flawed [57].

Predictions from binary classifiers are in the form of true or positive. In the case of outlier detection, true means outlier, and false means benign, or part of the expected distribution. These predictions can be categorized into true-positive (TP), false-positive (FP), true-negative (TN), and false-negative (FN). These four values often are grouped into a confusion matrix (CM) or used to calculate other performance metrics. CM is a table of contingency counts.

It is important to note that performance is not consistently reported in one metric in authentication systems. For instance, False Acceptance Rate (FAR), some times referred to as False Positive Rate (FPR) and False Rejection Rate (FRR), sometimes referred to as False Negative Rate (FNR), are used in many authentication works, where they are not meaningful in many cases since they present a point on a trade-off. FNR presents how often a legitimate user is denied, FPR presents how often an illegitimate user is authorized, True Negative Rate (TNR) presents how often an illegitimate user is denied, and True Positive Rate (TPR) how often a legitimate user is authorized. True rejection rate (TRR) is the probability of the system to correctly reject impostors, and True acceptance rate (TAR)

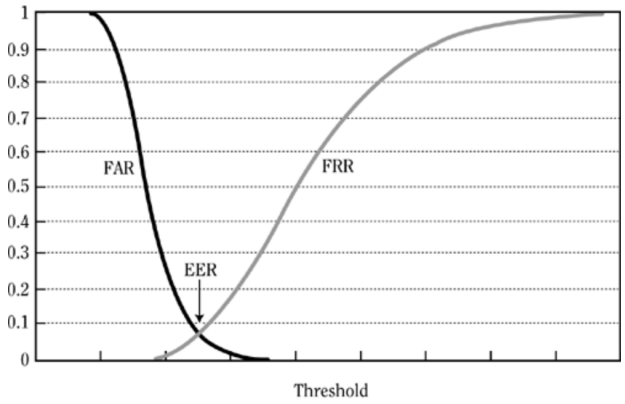

Figure 2.15: FAR, FRR and the EER point. Source: [7]

is the probability of the system correctly identify legitimate users. We can not tell based on these metrics whether the model is able to discriminate successfully or did the authors adjust the system parameters to inflate the metric. TPR and FPR, as well as FPR and FRR, are based on a compromise between two kinds of misclassifications, and hence all are inherently flawed.

One of them most reported metrics in the research domain is ACC.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

It is the relative frequency of correct classification. Often the value of accuracy that is reported is the maximum across thresholds. That presents a challenge to researchers because only a single threshold is represented in this performance metric. Researchers cannot know how the accuracy will change if the threshold is set in a way that satisfies an FPR requirement at an inconvenience to the TPR, for instance. ACC does not identify the type of user, be it authorized or unauthorized, when misclassification is made.

The most popular metric is Equal Error Rate (EER); it is the point that TPR equals 1-FPR on the Receiver Operating Characteristic (ROC) curve, as seen in Figure 2.15. The ROC curve is the curve of TPR and FPR by varying thresholds. Ideally, the curve should grow toward the top-left, meaning that the model makes correct predictions. The area under ROC (AUROC) is the probability that the scores of random legitimate users

are higher than an illegitimate user. But all single-number summaries hide the details of which errors occurred and how.

Precision, or Confidence, identifies what ratio of positive findings from the model are correct.

$$precision = \frac{TP}{TP + FP} \tag{2.2}$$

Recall, or sensitivity, measures what ratio of positive findings from the model are correct.

$$recall = \frac{TP}{TP + FN} \tag{2.3}$$

"Harmonic mean" of precision and recall (F1) is defined as a combination of the two metrics:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.4}$$

There has been work defining some metrics to use in authentication systems, but they all suffer from a lack of use and hence are very hard to use for comparing results and approaches to existing work [58, 57]. It is still best to report metrics like EER and ACC to allow other researchers to compare results and understand the impact new additions and ideas can bring to the system. It is advisable to report more metrics to allow researchers to grasp the fundamentals of the system behavior. It is also of pivotal importance to open source the code base, dataset, and model architecture to allow for easy reproducibility.

Another two quality metrics we aim to maximize our usability which is minimizing FNR and the time the system is blocking or hindering regular device usage and energy consumption which is minimizing the use of energy involved in the authentication process. Mobile devices, as well as the Internet of Things and others, are generally built as resource-constrained devices, so we aim to be considerate and help maximize the lifetime of the device benefiting from our solution.

In our work, we will use all the popular metrics and will allow for easy reproducibility by limiting ourselves to open-source datasets and open-source our work and models.

Chapter 3

Relevant Work

In this chapter, we look at various existing contributions, studies, and analyses found in our research domain, tackling different aspects of our solution. From well-known datasets to new machine learning algorithms that have profoundly impacted the literature, we explore the steps needed for our overall solution, the CIA system, and compare different methods. We underline the most relevant studies to our work, which also propose innovative Continuous and Implicit Authentication mechanisms based on smartphone sensor data. A summary of these works can be found in Table 3.2. Our literature survey spans various sensor modalities, threat vectors, data sets, and evaluation metrics. This makes it difficult for researchers to compare and cross-examine these methods. However, insightful comparisons, with fixed variables and controlled experiments, are presented in our extensive evaluations later in Chapter 5, to allow for hypothesis validations and further investigations.

3.1 Datasets

In the interest of practical studies, we start by choosing the categories of datasets that are general enough to produce reliable analyses. For instance, we exclude human activity recognition (HAR) related datasets [59, 60, 61] as they are usually collected in strictly controlled environments and hence are not useful for our real-world scenario Continuous and Implicit Authentication use case.

The LiveLab Traces [62] dataset consists of smartphone sensors, apps, and calls data from 35 users. However, the captured duration per user varied greatly as some users are tracked for a few days while others are tracked for a year. Surveys, such as those released by Abuhamad *et al.* [63] and Gonzalez *et al.* [64], have featured a plethora of continuous authentication works and datasets that have been geared towards features such as touch, leap motion controllers, orientation, pressure, camera, cyclic rotation metric, force sensing, compass, microphone, speaker, light sensor, gravity sensor, elevation, healthcare wearables, piezoelectric and electromagnetic energy harvester. These features and their respective datasets are used by multi-modal authentication systems that use keystroke dynamics, voice, gait, or motion-based sensors. However, the primary shortcoming of most of these datasets is their size and diversity, as they are typically gathered from fewer than 100 users. Thus, their results might be unreliable for real-world deployments and render most of the research work in this domain hard to use in a real environment. It is notable for highlighting that commercial-grade Continuous and Implicit Authentication systems do not currently exist due to performance and reliability limitations. To overcome the practical limitations, more research and development efforts are required in which the compilation of more diverse public datasets and better technologies need to be employed to push this domain further.

Multiple studies [65, 66, 19, 67] have chosen to collect their own custom datasets, sometimes from up to 1500 users [66] or 30 sensor modalities [67], but unfortunately have not made them public. Privacy concerns are among the primary reasons for authors refusing to release the users' data. However, this hinders the reproducibility and comparability of the resulting works. Along with the high cost of gathering such datasets, this compels us to avoid such an approach in our work.

The scarcity of high-quality labeled mobile sensor datasets has been one of the main obstacles to research in this area. In 2015, *Crowdsignals.io* [68] was born as a "crowdfunding campaign to fund the largest ethically collected set of high-quality, labeled mobile and sensor data for use by the research community". Although the project exceeded its funding target and generated incredible interest and support, its dataset is yet to be published.

The most fitting dataset to our use cases is a public dataset that has been the de facto standard in the field of continuous authentication since its release in 2015. HMOG [24]

is a public dataset released in 2015, which has been used extensively in the Continuous Authentication literature [16, 20, 19, 24, 69, 70]. HMOG includes data collected from smartphone accelerometers, gyroscopes, magnetometers, tap coordinates, finger-covered areas, or pressure from 100 subjects during 24 sessions. In the study group, predefined realistic scenarios have been played out by the users, which makes it more reliable than similar datasets [71]. All sessions in HMOG include active usage and are generally less than 15 minutes long. A slightly larger dataset was released publicly in 2019 by Belman *et al.* [72]. However, it has not been used as extensively in the domain, and this makes comparability of its results more difficult. Thus, we find HMOG to be one of the best candidates for our evaluations. The high diversity of scenarios, data balance, validation from existing literature, and appropriate features in the dataset make it a better alternative to the other public datasets in this domain.

With that, we conclude this section with the selection of the de facto HMOG as a suitable dataset for its inclusion of inertial data, its publicity, and the comparability it brings to our work. Moreover, we delve into the domain further and look at how inertial data type has been utilized in the smartphone Continuous Authentication use case.

3.2 Mobile Sensing

As discussed in Chapter 2, inertial data captures the motion, orientation, and location of the device in its surrounding space. The approaches that utilize this kind of data for non-intrusive user authentication leverage patterns from the user's behavior such as their gait, touchscreen interactions, hand waving, keystroke patterns, voice, or signature moves and thus create a behavioural profile.

One of the first authors that collected their own extensive continuous authentication dataset and used a one class distance-based classifier, Zheng *et al.* [73] used the inertial data coming from the device's accelerometer and gyroscope and augmented it with touchscreen data, acceleration, pressure, size of the touch area, and time intervals between interactions. They are able to analyze and profile how each user touches their phone while they put in their PIN codes and classify the current session as either the true owner or an

imposter with up to 3.6% EER. Another early work that used deep learning is Trojahn *et al.* [74]. They utilized both keystroke and handwriting analytics to authenticate users of a smartphone, using data of when the users are entering their password repeatedly. They utilized models such as the Multi-layer Perceptron (MLP) [75], Bayesian Net classifiers [76], and Naive Bayes [77] for their classification.

A few years later, Neverova *et al.* [78] used the large-scale data collected in Google's Abacus project and time-based deep feature extraction, using RNNs and CNNs for user authentication. They classified the data using a Dense Clockwork RNN model. As mentioned in Section 3.1, the Google Abacus dataset is reported to include data from 1500 users captured in real-life situations, but unfortunately, it has not been released to the public. Soon after, Shen *et al.* [19] showed great success on HMOG and have collected a dataset of more than 27,000 data instances from 10 subjects, extracting wavelet, frequency, and time-domain features and evaluating several algorithms such as Support Vector Machines, Hidden Markov Model, and K-Nearest-Neighbor.

Work	Approach	Model	Data
Yang *et al.* [79], 2013	Hand waving using linear accelerometer	SVM [80]	Sampling interval, acceleration
Shrestha *et al.* [81], 2015	Hand waving using ambient light sensor	SVM [80]	Time, light, gesture duration
Draffin *et al.* [82], 2014	Keystroke biometrics	Neural Network Classifier [83]	Location pressed on key, length of the press, touched area size, and drift
Derawi *et al.* [84] 2010	Gait biometrics using smartphone sensors	DTW [85]	Time interpolation, Average cycle length
Mantyjarvi *et al.* [86], 2005	Gait biometrics using accelerometer	Cross Correlation [86]	Acceleration, 10 bin fast Fourier transform (FFT) histograms
Kambourakis *et al.* [87], 2014	Behavioral profiling	MLP, Random Forest, and K-NN [75, 88, 89]	Time, speed, distance
Feng *et al.* [90], 2013	Keystroke biometrics	Decision Tree [77], Bayes Net [76] Random Forest [88]	virtual key combinations, holding time, pressure
Frank *et al.* [91], 2013	Touchscreen interactions	SVM and K-NN [80, 89]	Touchscreen Data
Sae-Bae [92], 2014	Line signature drawn with fingertip	DTW [85]	Touchscreen interactions
Kunz *et al.* [93], 2011	Speaker verification during ongoing phone call	HMMs [94]	Voice
Shahzad *et al.* [95], 2012	Support Vector Distribution Estimation	coordinates of each touch point, accelerometer values and time stamps	Touchscreen interactions
Clarke and Mekala *et al.* [96], 2007	Dynamic signatures by typing words	PDALok	various biometrics
Das *et al.* [97] , 2008	Speaker's identification based on speech dynamics	DTW [85]	Audio

Table 3.1: Table of important studies in smartphone behavioral user authentication. Adopted from [18]

There exist multiple important behavioural biometric studies in the scientific literature, which are summarized in Table 3.1 adopted from Ul-Haq *et al.* [18]. The approaches these studies took span gestures, keystroke, touchscreen, handwriting, voice, and gait. The primary limitations in gestures-based studies include the fact that they require the user to interact actively in the authentication process, and once a device is unlocked, an imposter can not be detected. On the other hand, solutions based on keystroke dynamics are limited in that they require a lot of data compared to other modes, disruptions during typing will give false signals, they get affected by the variations in user behaviour (*e.g.*, different moods) and switching keyboards might change the learned patterns. Touchscreen-based studies also have shortcomings in those interactions in every orientation are different, and the current user activity affects the interaction significantly. Methods based on handwriting are mainly not developed for the sake of Continuous Authentication capability, and the phone may not be steady all the time to confidently detect a change in the pattern. Voice-based works are hampered by the surrounding environment's noise. Gait-based works are vulnerable to changes created by different outfits that may alter the walking pattern, as well as the requirement to fix the sensors on the body at all times to a good position [18].

Multiple works [79, 91, 70] have been aimed at authenticating a user while doing a specific activity like typing a password, making a call, or picking the phone from the table. It mainly consists of human activity recognition and then authentication. Though that might be an easier machine learning problem and might yield good better results, it suffers from a lack of continuity. For instance, once a user is authenticated, the phone would not know how to classify the activities in their various types that are happening until that same activity happens again later. This subdomain in user authentication benefits from activity recognition advancements, covered in Appendix A.2, in the last few years that made detecting a user's current activity accurately a reality. User activities that motion sensors can capture can be categorised into *simple* and *complex*. Simple activities include walking, sitting, sleeping, going upstairs or downstairs, or laying down. Conversely, activities like driving a car, changing clothes, riding a bike, and exercising are typical examples of complex activities. For instance, using GPS and accelerometer data, Martin *et al.* [98] propose a method that detects periods of walking, biking, car, bus, and rail transportation methods in real-time. GPS data is successfully used by Martin *et al.* [98] to achieve 96% accuracy with

a random forest classifier on data transformed with Principal component analysis (PCA) and recursive feature elimination (RFE). However, GPS stays an infeasible solution for practical usage in CIA use cases due to its high battery power consumption and necessity to have user's permissions to operate and utilize its data. Also, using accelerometer and gyroscope data, Anguita *et al.* [99] used a Support Vector Machine (SVM) model to detect walking at a 95% accuracy, climbing upstairs at a 72% accuracy, standing at 92% accuracy, sitting at 94% accuracy, laying down at 100% accuracy, and going downstairs at 79% accuracy. Using similar features, Ronao *et al.* [100] explored human activity recognition with deep learning and artificial neural networks, with an accuracy of ˜95%.

3.3 Data Preprocessing

Data preprocessing refers to a stage where certain techniques are used to transform data into a more desired form to enhance learning for an artificially intelligent agent. As the studies in the Continuous and Implicit Authentication domain use very different mechanisms to perform data preprocessing, we cover the most important approaches in the upcoming paragraphs.

One of the main drawbacks of mobile device data sensors and inertial data sensors specifically is the signal-to-noise ratio. As discussed in Chapter 2, the data will almost always get contaminated in the collection phase, and researchers aim to improve the quality of the data being fed to their models using noise reduction modules. For instance, Deb *et al.* [67] utilized the Fast Fourier Transform to map inertial data measurements from the time to frequency domain.

In an interesting study, Shen *et al.* [19] used kinematic information extraction to filter measurements of inertial sensors and touch events. The authors employ a Kalman filter and then decompose signals using wavelet functions and threshold analysis. They then reconstruct the original signal by using inverse wavelet functions. This approach handles noise that appears in the whole spectrum as opposed to Reyes-Ortiz *et al.* [101] where the authors separated movement and gravitational elements of the data from the accelerometer data and used a butter-worth low-pass filter with a cutoff, 0.3 Hz, to capture

the low frequencies of gravity. Similarly, Haq *et al.* [102] used a smoothing filter of three samples length at a certain frequency, 50Hz.

Several works in this domain construct new features by applying operations to raw features, not to add any new information to the data but to transform the data into a more suitable representation and form for the models to learn. Such operations can include sliding time windows and statistical and temporal metrics of the data. The relevant constructed features in the Continuous Authentication domain include magnitude, min, max, entropy, and an average of the gyroscope sensor data, as well as the entropy and magnitude of the accelerometer sensor data. Cycle-based approaches allow for features to be aggregated without fixed windows but rather based on cyclic events. For instance, human steps in walking can be used in gait models. Cycle detection is usually the most difficult to perform accurately in real-world scenarios since the phone might be held or attached loosely rather than fixed or strapped properly to the body [13].

Sarova *et al.* [24] developed a set of metrics to capture grasp resistance, grasp stability, tap, keystroke, and touch events in HMOG. Further sampling, scaling, cropping, and jittering are all techniques that have also been used on top of HMOG [24] by Li *et al.* [69]. In our experiments, we also propose our own preprocessing method, which is covered in Chapter 5.

Dimensional reduction is another technique for data preprocessing. By transforming the data into a lower-dimensional space, the computational effort needed to analyze an event can be reduced while retaining the most important components to generate insights. In the effort to represent the data in the new space, the pre-processor is compelled to discard the noise and low-importance features while keeping the important semantics and patterns. Our approach is designed to find the most meaningful subset inherently given any data instance, as explained in Chapter 4.

Among other works, Neverova *et al.* [66], utilized PCA to perform dimensionality reduction. Other work like Shen *et al.* [19] utilized mutual information and fisher custom score thresholds to reduce their features by more than 80%, which leads to 38 final features, including kurtosis and skewness.

Some of the works in the literature use deep features instead of manually computing

metrics to better represent their raw data. Neural Networks can learn filtering, feature construction, and dimensionality reduction effectively. Neverova [66] evaluated CNN, plain Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Clockwork Recurrent Neural Network (CNN) to analyze their ability in generating deep features for the purpose of Continuous Authentication, as well as proposed Dense Clockwork Recurrent Neural Network (DCWRNN).

Centeno *et al.* [20] used a Siamese CNN to construct features that differentiate imposters from authenticated users from raw sensor data. These deep features would then be fed to an OCSVM for classification. Their work is by far the most regarded in the domain, and its re-implementation is covered thoroughly in Chapter 5 and Chapter 6 of this thesis.

As part of the data preprocessing, a model can try to understand the environment in which the data was collected. This can be done through third-party application collaboration like that described in Chapter 2 or alternatively through evaluating the user activity or status at the temporal neighborhood. Some works [102, 103] have explored environment sensing by detecting the context first then doing authentication later. However, their results indicate little benefits in their implementations and experiments. We propose a variant of this technique in Chapter 7.

We will be experimenting with different methods of data processing in Chapter 5 that include deep feature extraction using a variety of models.

3.4 Machine Learning Models

Numerous works have tackled the problem of Continuous Authentication and proposed a variety of machine learning models and data pipelines to push towards higher performance scores and lower latencies. This section provides an overview of machine learning models which are considered to be most relevant to our work.

We could approach the problem with either the one-class approach, binary class approach, or multi-class approach. A one-class model only trains on the device owner's data and aims to classify this user given sensory data at any given point. The multi-class model

would have access to a few users, of which one is the device owner; this would make sense if we were able to enumerate and collect data for all the possible imposters. Such a case where multi or binary class would make sense is in a closed environment such as a strict campus or facility.

However, due to the uncertainty of who would be the imposter in our use cases and threat models, detailed in Chapter 2, we, along with most of the works in this domain, choose to focus on the one class approach. Where we can not assume we have data on the imposter's behavioral profile, but rather only the device owner.

Most Relevant Studies that used HMOG				
Work	**Features**	**AI Model**	**Performance**	**Time**
Centeno *et al.* 2018 [20]	Deep	Siamese CNN and OCSVM	97.8% Accuracy	1s
Zhu *et al.* 2013 [70]	Constructed	n-gram	FAR 13.15%, FRR 15.29%	4.96s
Li *et al.* 2018 [69]	Constructed	OCSVM	4.66% EER	5s
Shen *et al.* 2018 [19]	Constructed	HMM	FAR 5.13%, FRR 6.74%	8s
Centeno *et al.* 2017 [16]	Raw	Autoencoder	4.5% EER	20s
Volaka *et al.* 2019 [104]	Deep	Deep Neural Network	15% EER, 88% Accuracy	-
Sitova *et al.* 2015 [24]	Constructed	SMD	7.16% EER walking, 10.05 EER sitting	80s
Li *et al.* 2018 [105]	Constructed	KRR	3.0% EER	-
Amini *et al.* 2018 [106]	Constructed	LSTM	72.29 ACC	-

Table 3.2: Best One Class solutions that reported results on HMOG

Centeno *et al.* work [20] shows great promise, as they achieve very high accuracy and low latency with the help of a Siamese CNN network for deep feature extraction along with an OCSVM for impostor classification. Buech [13] re-implemented the work reported by Centeno *et al.* [20] and proposed improvements to the evaluation criteria. Their improved evaluation criteria make the experiment closer to real-life scenarios. For instance, normalization is performed only on the benign subset of the training dataset. We also adopt this approach and build upon the state-of-the-art, as detailed in Chapter 5.

Artificial neural networks have been used in deep feature generation and authentication in this domain. Shen *et al.* [19] used a three-layer MLP against a Hiden Markov Model and an OCSVM. An autoencoder has been used for outlier detection by Centeno *et al.* [16], with very promising results. The authors trained the autoencoder with samples from the owner in a single-class approach. The model learns to reconstruct the benign data better than the data it never saw, which is assumed to be anomalous. The distance between the input vector and its reconstruction is expected to be lower for samples from the real device owner compared to a sample from an impostor. Binary classification is achieved by applying a threshold to the model's confidence in the input's anomaly [13].

Volaka *et al.* [104] used a Deep Neural Network trained on touch and motion features. The model used by the authors consisted of 3 dense layers and 128 nodes and was trained for 200 epochs. In their evaluations, they achieved 15% EER and 88% accuracy. As seen in Chapter 5 we outperform their results, using much less computational power, thanks to our use of generative neural models.

3.4.1 Generative Models

VAEs [107, 108, 109] and GANs [110] are among the most popular and established generative neural network models. The regularities in the training data are expressed by a probability distribution in a lower-dimensional latent space, which is created by the generators. The use of generative models for anomaly detection has been explored extensively in machine learning literature, making it a very fruitful area of research in the past few years.

The details of VAE models and their usage is covered in Chapter 4. We take this opportunity to highlight GANs, which we encourage as an avenue for future works (*cf.*, Chapter 7). Exploring the utilization of GANs along with VAEs, like in Adversarial Autoencoders [111, 112, 113, 114] can create many opportunities in future research, such as the work put forward by Ibrahim *et al.* [115] or Ueda *et al.* work [116], for the CIA use case.

GANs pose the problem of learning the target distribution as a zero-sum game. The generator network is trained in competition with an adversary, the *discriminator*, that challenges it to generate samples whose distribution is similar to the training distribution. AnoGAN [117] was one of the first works to propose GANs for anomaly detection, which started the emerging research field. It aimed to use GANs trained on benign data only. The generator will then learn how to make new benign samples, while the discriminatory will know how to tell benign from not. When an outlier instance is encoded, its reconstruction will be benign, and the difference between the input and the reconstructed instance will highlight the anomaly. One of the limitations of the proposed method was that it suffered in terms of test-time performance. Moreover, the anomaly score, set to be a convex combination of the reconstruction loss and the discrimination loss, is hard to interpret as compared to more familiar metrics such as probabilities. The objective of the GAN was also not modified to take into account inverse mapping learning.

Further evolution of these methods included EGBAD [118] which outperforms AnoGAN, and GANomaly [119] which outperforms both in accuracy and speed. Mattia *et al.* [120] published a survey evaluating all GAN-based anomaly detectors and discussing their inherent differences and strengths. The use of adversarial generative models is beyond the scope of our work as there are genuine concerns regarding their robustness and computational complexity, as systematically shown by works such as Skvara *et al.* [121].

VAEs are a popular class of probabilistic AutoEncoders [122]. Xu *et al.* [123] use VAE to detect anomaly data for seasonal KPIs in web applications since VAE prevents over-fitting by allowing noise and randomness of latent variables. Chen *et al.* [124] uses the VAE-LSTM architecture for anomaly detection and robust prediction of time series. Luo *et al.* [125] use the VAE-SVDD with a BiGRU hidden layer and a latent variable that is big enough to use temporal correlations. VAE-based anomaly detectors are redefining

the state-of-the-art performance and speed in many applications *e.g.*, web, image, and intrusion detection [126, 127, 128]. Park *et al.* show VAE anomaly detectors, with LSTM, on multi-modal sequential data [129]. Another work in this area, Cerrie *et al.* [130], is geared towards new physics mining at the Large Hadron Collider.

We aim to utilize VAEs in this use case and experiment with a few implementations of VAEs in Chapter 5. We do not aim to test GANs due to robustness and complexity issues [121].

3.5 Experiment Settings

A wide range of experiment and evaluation settings, as well as datasets, are being used in the literature for showcasing the effectiveness of CIA frameworks, which makes it impossible to have fair comparisons between results. This is concerning for the research in this area, as it is commonplace for authors to compare results against other works without further information into the evaluation configurations.

The performance metrics also vary between different works, although EER and accuracy are by far the most popular, and that adds to the comparability of these works. We delved into performance metrics in the previous chapter Chapter 2 and furthermore discussed what metrics are more fitting in this domain. Some works report metrics for subsets of their data, like certain scenarios individually, as seen in Sitova *et al.* [24] while some use general metrics for their entire datasets *e.g.*, Centeno *et al.* [16, 20].

Another problem is that the setup settings vary greatly between these works. While some of them might have 10-fold cross-validation random, one vs. one scenario, [20], others randomly draw from a pool of attacks and sample benign instances along in a one vs. all fashion [69]. Some even reported values that were improved by calculating the mean of the individual sample's score across a sliding window [19, 65].

All of these issues pertain to the works using the same public dataset, as seen in Figure 5.18. As discussed in Section 3.1, many publications *e.g.*, Neverova *et al.* [66], and Deb *et al.* [131], work with data that is not available for others to reproduce. This adds

another layer of complexity that makes comparability in the domain very difficult without all of these settings somehow fixed and standardized.

Shen *et al.* [19] took the effort to compare many one-class models and offer us some comparable results from which we can draw insights and conclusions, but unfortunately, through correspondence, we were unable to get the codebase or the datasets they used. We aim to also run many models in a comparable fashion to allow others to draw insights and conclusions from our work. Conversely, we are committed to releasing code and data used in our work in the interest of transparency and reproducibility.

We are going to explore a few experiment settings in Chapter 5, with an aim to be comparable to other works as well as find a setting that is highly realistic, given our aim to bridge the gap between academic works like those presented and real-world use cases and commercialization that is yet to exist in this domain.

Chapter 4

Approach

In this chapter, we discuss our approach to solving the problem discussed in Chapter 2. This chapter details the thought process, decisions, architecture, considerations, and design of the system proposed due to the study presented in this thesis. Our proposed solution aims to act as a trigger module based on a change in the typical behavior of phone usage and movements. Previous research efforts have tried to detect the act of handing over itself, *i.e.*, threat model (B), or the act of leaving behind itself, threat model (C), as trigger events that can be fingerprinted and detected in their recurring variants [132]. We find the detection of the user behavior deviating from being the most pivotal to the resiliency and robustness to the coverage of the use cases and threat models domain. Nonetheless, our solution can work in tandem with other modules and systems, like the ones detecting the very events of handing over or leaving behind, to enhance the overall detection coverage, capabilities, performance, and abilities. This chapter discusses the data, the models, and the techniques we considered to evaluate the solution.

4.1 Data

We choose to work with inertial data only, hence effectively limiting our algorithms to data streams coming only from the accelerometer, gyroscope, and magnetometer in the forms

commonly installed and fitted on off-the-shelf smartphones, as covered in Chapter 2. This ensures continuous availability and low privacy concerns, low resource footprint, low power footprint, and no limitations imposed by the operating system. This will allow us to track deviations in minor movements in the user demeanor and make it possible for the module to identify a person's unique behavior, given that the model is powerful enough.

4.2 Classical Machine Learning Models

Our selection of machine learning models limits the options to anomaly detection or outlier detection, or novelty detection algorithms. Anomaly detection refers to the task of detecting anomalous instances given the distribution of data. An observation belonging to the distribution is commonly referred to as an inlier, while any outlying instance is commonly referred to as an outlier or an anomaly. The three approaches to this task are:

- Supervised
- Unsupervised
- Semi-Supervised

Supervised models are trained on observations with a ground truth label for both inliers and outliers. The supervised approach is taken when ground truth labels are given, and it is assumed that the anomalies will follow the same distribution as in the training data set.

Unsupervised models are trained on observations without any ground truth label for both inliers and outliers alike. The models can differentiate outliers during the training period and fit to detect the difference learned. Unsupervised techniques are preferred when the anomalies are defined as points that do not belong to high-density regions.

Semi-supervised models are trained and fit on labeled observations describing normal behavior only. The approach is taken when *outliers* are defined as data points that differ from the distribution of the benign training data, given as ground truth. Any observations differing from the benign training dataset are considered anomalies, even if they form a

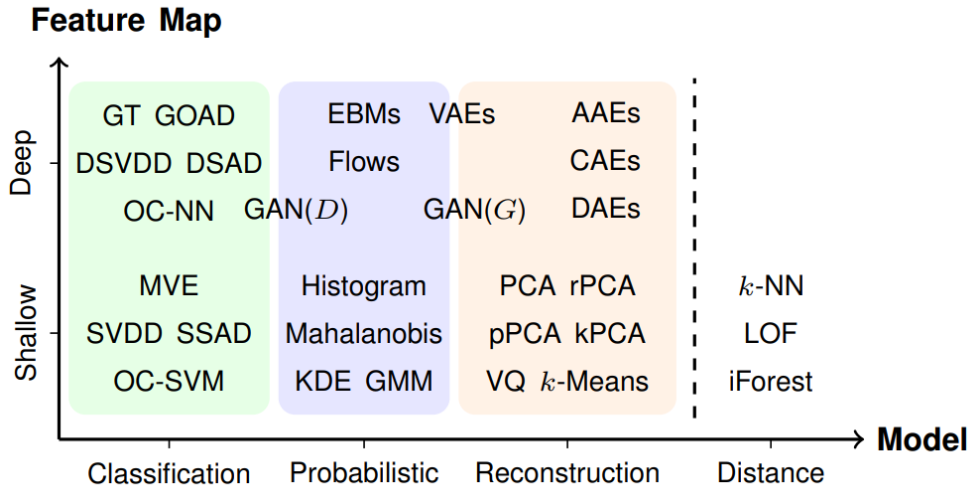

Figure 4.1: Anomaly detection approaches arranged in the plane spanned by two major components (model and feature map) of our unifying view. Based on shared principles, we distinguish One-Class Classification, probabilistic models, and reconstruction models as the three main groups of approaches that formulate shallow and deep models. Purely distance-based methods complement these three groups. Adopted from [8]

high-density region or cluster. The semi-supervised approach is the best fit for this problem since we would have the opportunity to gather ground truth labels from the benign user of the phone at some point.

Although state-of-the-art and many works in past literature base their findings on Deep Learning techniques, we do not aim to make the system lightweight, fast, and efficient in the interest of practicality. Due to their recent success in many anomaly detection problems, we also aim to leverage generative models such as Variational Auto-Encoders (VAE). VAE have a wide range of varieties. We decide to utilize the most commonly used models in the domain and choose one based on timing, performance, and stability across a few experiments. The models we chose to evaluate our VAE, β-VAE, KNN, OCSVM, ABOD, CBLOF, FeatureBag, HBOS, IForest, LOF, PCA. This selection allows us to sample the most esteemed algorithms across linear, proximity-based, probabilistic, ensembles, and neural networks, as seen in Figure 4.2. Other familiar categorization of anomaly detection algorithms showed in Figure 4.1, supports that this selection is diversified and will allow us to capture and evaluate broad categories of algorithms in our use case. In the following

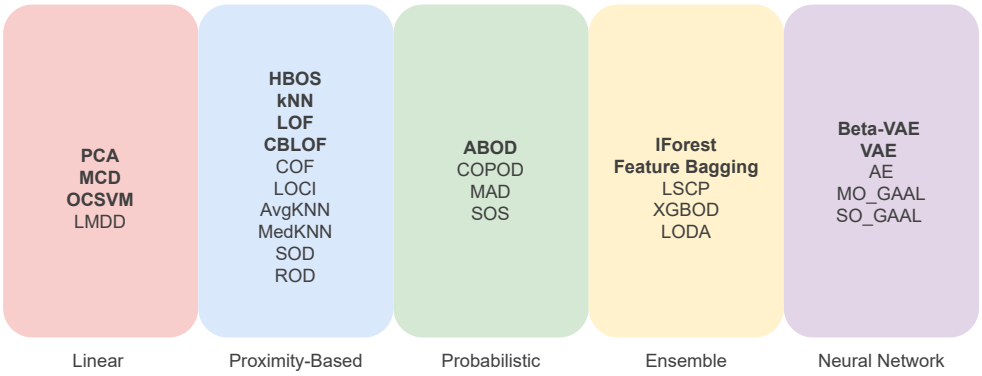

Figure 4.2: Anomaly detection approaches arranged in the plane categorised by their underlying type. The 5 categories in the anomaly detection models literature are linear based, proximity based, neural networks based, probabilistic based, and ensembles. Models we used in our evaluations are bold-ed for emphasis.

sections, we provide a brief introduction to some of these models.

4.2.1 Linear, Probabilistic, Density, and Ensemble-based Models

Linear

PCA (Principal Component Analysis) [133] is a linear transformation often used for dimensional reduction to allow for easy data exploration and analysis. The technique used highlights the variance co-variance structure of a few variables through a set of new variables, which are the original variables' functions. The principal components are linear combinations of these random variables that are not correlated, sorted by variance starting from the first, and conserve the total variation in the original variables.

OCSVM (One-class Support Vector Machine) [134] is a linear algorithm that aims at learning a decision boundary to group the data points. Once the model is trained, each data point is classified based on the normalized distance of the data point from the determined decision boundary.

MCD (Minimum Covariance Determinant) [135] is an estimator of multivariate location and scatter. It is to be applied to Gaussian data but can also be helpful on data drawn from a uni-modal, symmetric distribution.

Proximity-based

HBOS (Histogram-based Outlier Score) [136] is a proximity-based technique that models uni-variate feature densities using histograms with a fixed or a dynamic bin width. Once the computations are in, all histograms compute an outlier score for each instance presented in the data. It can achieve linear time complexity, $O(n)$, if used with a fixed bin width.

KNN (K-Nearest neighbors) [137] is a proximity-based model. For each data point, the whole data set is examined to extract the k data points with the most similar feature values. These are then defined as the k nearest neighbors. Once the nearest neighbors are found, the data instance is classified as anomalous if and only if the majority of those nearest neighbors were previously classified as anomalous; otherwise, the data point is benign.

LOF (Local Outlier Factor) [138] is a proximity-based method that captures precisely the relative degree of isolation of an object from its surrounding neighborhood. The outlier factor is local because only a restricted neighborhood of each object is taken into account. The approach is loosely related to density-based clustering.

CBLOF (Cluster-Based Local Outlier Factor) [139] is a proximity-based method. The outlier score is computed by the distance of each instance of the data to its respective cluster center multiplied by the instances belonging to its cluster.

Probabilistic

ABOD (Angle-based Outlier Detection) [140] is a Proximity-based technique built for high-dimensional data anomaly detection tasks. It is a parameter-free approach based on the variance of angles between pairs of data instances. Using the variance in the angles between a data point to the other points as an anomaly score, ABOD is one of the most popular techniques for anomaly detection on high-dimensional data sets.

Ensemble

Forrest (Isolation Forest) [141] is an ensembling technique that is based on the usage of numerous isolation trees, a tree structure constructed effectively to isolate every single instance. The idea is based on the susceptibility of anomalies to isolation; hence anomalies are found closer to the root of the tree; whereas ordinary points are isolated at the deeper end of the tree. Isolation Forest builds an ensemble of isolation trees and defines *anomalies* as those instances with the shortest average path lengths on the trees formed.

FeatureBag [142] is an ensemble technique applying Feature Bagging for outlier detection. The technique is based on combining results from multiple outlier detection algorithms that are put into action with different sets of features. Every outlier detection algorithm uses a small subset of randomly selected features from the original feature set. The anomaly scores computed by the individual algorithms are then combined to find better quality anomalies in the data set.

Limitations

Linear models like PCA are limited to data encodings that can only exploit linear feature correlations [8]. Deep models need significant amounts of training data to achieve acceptable performance; hence we will refrain from making our VAE and β-VAE deep: exceeding three layers in the architecture. KNN needs to have the number of neighbors K manually chosen; it also suffers from the "curse of dimensionality" and needs data scaling as it does not perform well on imbalanced data [143]. Classification models like SVMs are not appropriate for non-linear problems, have overlapped classes, or have many features. Many of these models also are very slow, as we will see in the experiment chapter.

4.3 Generative Models

A recent addition to the domain of outlier and anomaly detection is the use of generative models. The generative model is an umbrella term that covers a vast spectrum of

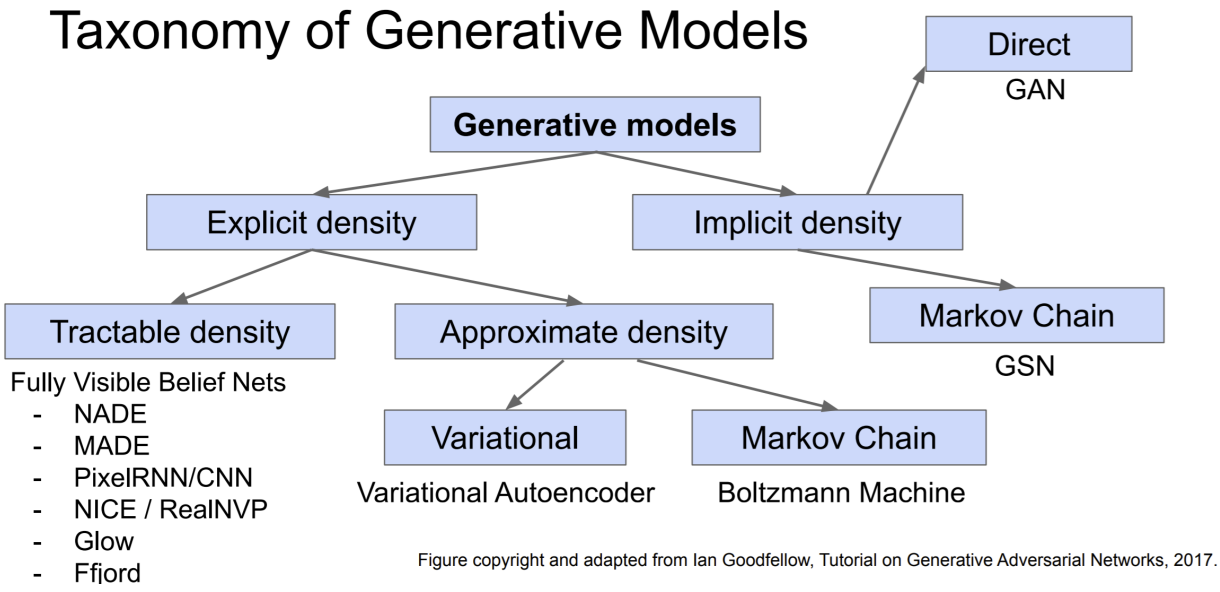

Figure 4.3: Taxonomy of generative models. Source: [9]

ML models. VAEs [109] are among the most popular ones and have shown great performance in recent ML literature for a large variety of tasks ranging from fairness in ML to better image generation. It is recognized that generative models can yield better performance for abnormal event detection due to their inherent Gaussian distribution modeling properties [144]. Other examples of popular generative models are Generative Adversarial Networks (GAN) [145], PixelRNN [146] and PixelCNNs [147] which leverage Recurrent Neural Network (RNN) [148] and Convolutional Neural Network (CNN) [149] respectively, as seen in Figure 4.3. VAE optimizes variational lower bound on likelihood and hence produces good latent representations and allows inference queries. PixelRNN and PixelCNN are explicit density models that optimize exact likelihood and make good samples but inefficient sequential generation. GANs are a game-theoretic approach and generate the best samples but can be tricky and unstable to train, with no inference queries. [9]

VAEs are stochastic inference algorithms deeply rooted in Bayesian statistical models. However, they are based on the older Auto-Encoder model, which has two neural networks: an *encoder* and a *decoder*. In the Auto-Encoder Figure 4.6, the encoder network aims to learn a smaller representation of the data it trains on, and then the decoder reconstructs

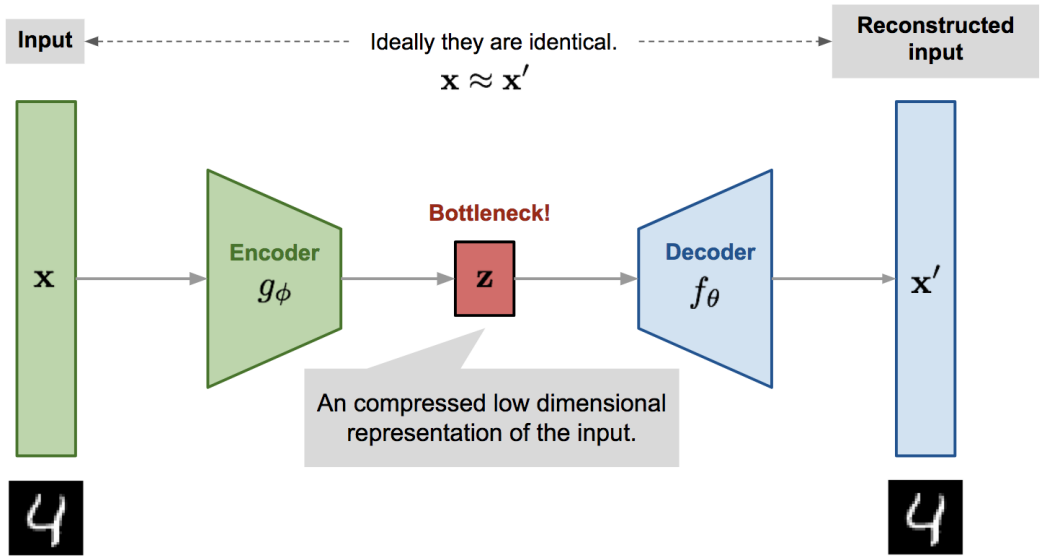

Figure 4.4: Auto-Encoder model architecture. Source: [10]

the original data from the smaller representation the encoder created, outputting the reconstruction of the input data to the encoder. The encoder learns a compressed representation called code, bottleneck, vector, latent features, latent variables, latent space vector of the input. The decoder learns how to reconstruct the input based on the latest features. The network as a whole tries to minimize reconstruction error: by minimizing the input and output differences. VAEs have been proved across many works and data sets to be the best novelty detection generative model due to their unique properties [121].

4.3.1 Auto-Encoder

The Auto-Encoder, seen in Figure 4.5, can serve as a good anomaly detector since it has excellent feature extraction capabilities. Imagine training an Auto-Encoder on benign samples and minimizing reconstruction errors on it. After training, the Auto-Encoder sees malicious input and cannot reconstruct with low error rates anymore since it only knows how to recognize the patterns of the benign data. An anomaly score could be set to mimic the reconstruction error. It serves as a deviation-based anomaly detector in what is referred to as a semi-supervised learning fashion.

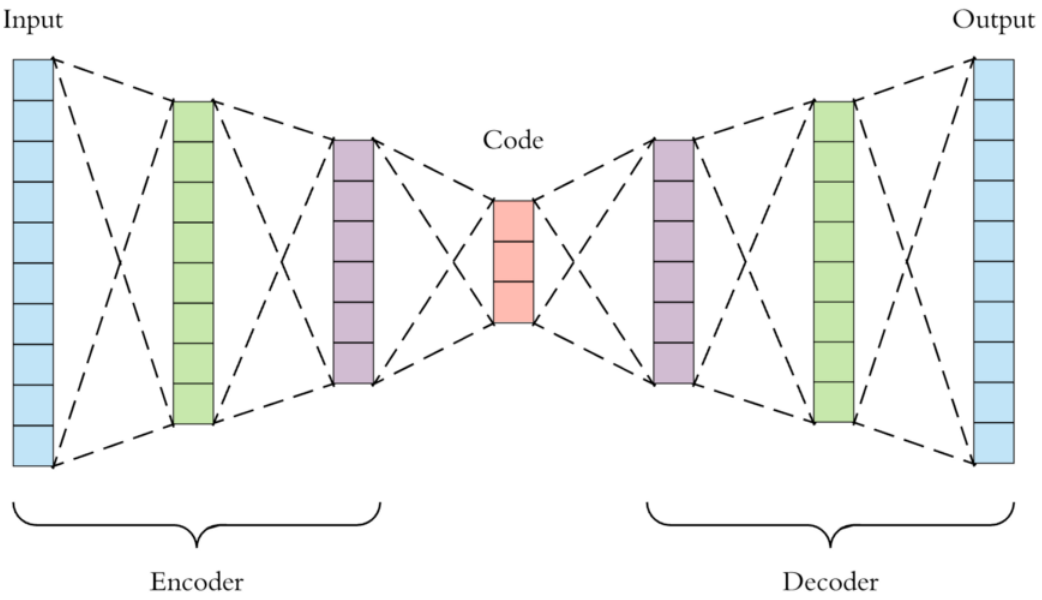

Figure 4.5: Auto-Encoder architecture featuring the encoder and decoder multi layered networks. Source: [11]

4.3.2 Variational Auto-Encoder

VAE is the graphical Bayesian inference probabilistic variant of the Auto-Encoder. As opposed to the Auto-Encoder where the encoder and decoder implement two complementary deterministic transformations, in VAE, it is *a distribution* that is being learned, and the output is *a draw* from that underlying distribution. The VAE's small representation is referred to as the *latent variable* and usually fits a *prior distribution*. The standard choice for that distribution is the *Gaussian* distribution [150], although other distributions such as the Mises-Fisher distribution are used in some variations of the VAE. The encoder is its posterior distribution, and the decoder is its likelihood distribution. Prior is a belief in some quantity, typically on a set of parameters, without observations at the data. When data is involved, the belief is updated and is called a posterior.

As part of a class of likelihood-based directed graphical generative models, a VAE maximizes the likelihood of the training data according to a generative model [151] [152]. We know an Auto-Encoder minimizes reconstruction loss, but a Variational Auto-Encoder

also ensures that latent vectors are sampled from a Gaussian mixture and take a fixed range of values, referred to as latent loss. Unlike AEs, VAEs can also generate data much like GANs without the overhead of introducing a min-max game or the Nash equilibrium challenge. VAEs are very useful with complex distributions as they try to find a smaller set of latent features that have much easier probability density distributions to model and fit them to a Gaussian distribution, for instance [12] [153].

By providing a probabilistic measure, the VAE can provide an *anomaly score*. This means that they eliminate the need to infer a cut-off threshold for the anomaly score but instead have a continuous anomaly score that follows a Gaussian distribution. A high anomaly score indicates that this sample deviates significantly from the training data and hence must be more anomalous in nature [151]. Hence the intuition behind using the VAE for anomaly detection is that the anomalous data will deviate from the mean of the distribution *i.e.*, the benign data, which has been represented in a low dimensional space. Judging based on reconstruction probabilities, the model can spot the anomalous data easily.

Usually, a forward pass in a VAE encodes an instance into the mean and standard deviation of the latent variable. It then samples from the latent space normal distribution and decodes the sample into a mean and standard deviation of the output variable. The model finally draws a sample from the output variable's distribution to produce the reconstruction. We can then perform backpropagation after every mini-batch to update our encoder and decoder parameters (θ and ϕ), based on our loss function to maximize the likelihood of our training data. Given x is the input, the reconstructed x' from a VAE trained on MNIST [154] looks like what we can see in Figure 4.7, when z, the compressed code learned in the bottleneck layer, is varied, showcasing the 2-dimensional latent space capturing interpretable and independent factors, more on that is seen in Figure 4.8 where we can see that one dimension captured smile and the other captured head pose. This suggests that z dimensions are great feature representations as it captures those interpretable semantics. The encoder is great for feature representation, and since we can define a threshold for the reconstruction probability based on the latent space, it can be helpful for classification or anomaly detection, as it allows inference of $q(z|x)$, the estimated posterior probability function, also known as the probabilistic encoder. All mathematical symbols are carefully

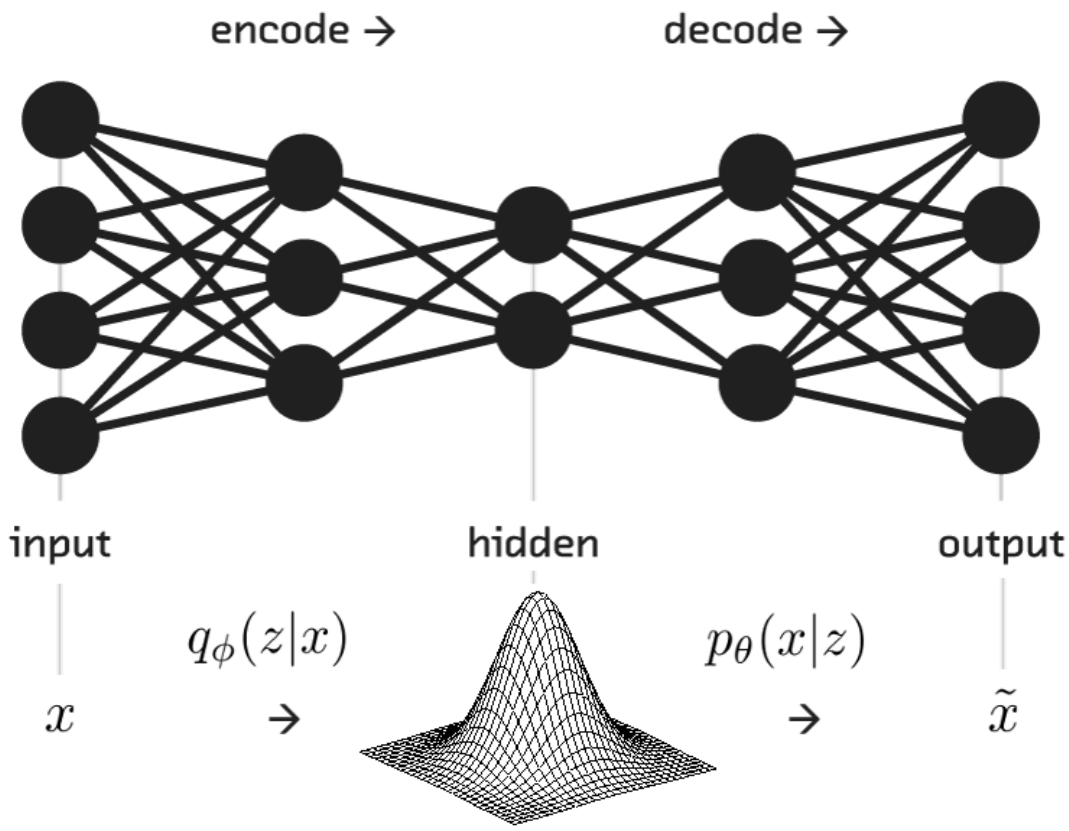

Figure 4.6: Variational Auto-Encoder representation. Source: [10]

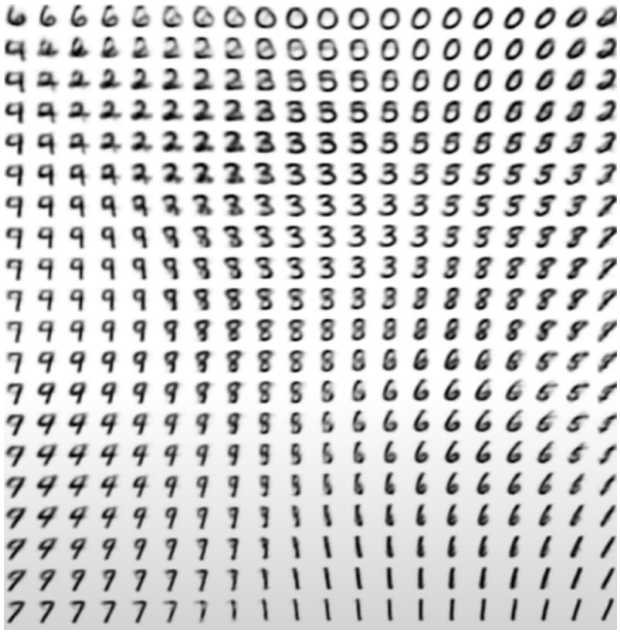

Figure 4.7: Reconstructed samples from a VAE trained on MNIST. Source: [12]

defined in **??**.

As mentioned before, a VAE does not map the input to a fixed vector like an Auto-Encoder does, but rather to a distribution p_θ where θ stands for the parameter of that distribution with an optimal, hence real, value θ^*. Given z representing a latent encoding vector, or compressed code learned in the bottleneck layer, and $\mathbf{x} \in \mathcal{D}$ representing our original input from the dataset, we are able to define the prior, often approximated with a learnt $q(z)$ or simply $\mathcal{N}(0, \boldsymbol{I})$ as:

$$p_\theta(\mathbf{z}) \tag{4.1}$$

the posterior as:

$$p_\theta(\mathbf{z}|\mathbf{x}) \tag{4.2}$$

the likelihood of generating true data sample given the latent code, also known as probabilistic decoder as :

$$p_\theta(\mathbf{x}|\mathbf{z}) \tag{4.3}$$

To generate a sample that looks similar to the real input $\mathbf{x}^{(i)}$, we sample $\mathbf{z}^{(i)}$ from the

Figure 4.8: Reconstructed samples from a VAE trained on faces. Source: [12]

prior distribution $p_{\theta^*}(\mathbf{z})$ and generate $\mathbf{x}^{(i)}$ from the conditional distribution $p_{\theta^*}(\mathbf{x}|\mathbf{z} = \mathbf{z}^{(i)})$. This allows us to define what is the real or optimal parameter of the distribution θ^* as such:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{n} p_\theta(\mathbf{x}^{(i)}) \tag{4.4}$$

Since it will be the parameter that maximizes the probability of generating $\mathbf{x}^{(i)}$, we can also use log probabilities to change the right hand side into a summation. Trying to

increase the likelihood of our training data, we define

$$p_\theta(\mathbf{x}^{(i)}) = \int p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z} \tag{4.5}$$

This involves the simple Gaussian prior and the decoder neural network. The equation above is not tractable, as we cannot compute every $P(x|z)$ for every z. Calculating $p_\theta(\mathbf{x}^{(i)})$ is a challenge as it requires to check for all the possible values of $\mathbf{z}$. We can reduce the value space to facilitate faster search by utilizing an approximation function to the posterior, $q_\phi(\mathbf{z}|\mathbf{x})$, with a parameter ϕ, to output a likely $\mathbf{z}$ given $\mathbf{x}$. Estimated posterior $q_\phi(\mathbf{z}|\mathbf{x})$ should very similar to the real posterior $p_\theta(\mathbf{z}|\mathbf{x})$. Hence we work out the log of the data likelihood to the following equation, equipped with our neural networks

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \tag{4.6}$$

If we look closely at the above equation, we can see the first element $\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})$, simply tries to reconstruct the input data, and can be provided by the decoder network and computed through differentiable sampling by the using the reparametrization trick. The second element $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$ makes approximate posterior distribution close to the prior, and is between two Gaussians and hence has a nice closed-form solution. The third element $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$ is intractable due to $p(\mathbf{z}|\mathbf{x})$ being intractable posterior, but we know this term will be greater than or equal to zero by definition. The first two elements then form a tractable lower bound that we can take the gradient of and optimize.

We define the loss function as

$$\begin{aligned} L_{\mathrm{VAE}}(\theta,\phi) &= -\log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \\ &= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) \\ \theta^*,\phi^* &= \arg\min_{\theta,\phi} L_{\mathrm{VAE}} \end{aligned} \tag{4.7}$$

The goal of a VAE in training is to minimize the KL divergence of $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ and $p_\theta(\mathbf{z}|\mathbf{x})$ and as well as the reconstruction probability of x. It generalizes more easily

than an Auto-Encoder because it works with probability distributions [12]. Numerous works have proved Variational Auto-Encoders' high effectiveness in anomaly detectors and showcased their use [155, 156, 157, 158].

β-VAE [159] is a variation of the VAE with the goal to discover disentangled latent factors. When each variable in z is sensitive to only one specific generative factor and relatively invariant to other factors, the representation is defined as factorized or disentangled. Disentangled representations are good for interpretability and generalize to a variety of use cases. We will also be evaluating β-VAE for our use case.

$$L_{\text{BETA}}(\phi, \beta) = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + \beta D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \tag{4.8}$$

The above equation represents the β-VAE loss function, and as we can see, the β parameter can be considered a hyperparameter that, when it exceeds the value of one, applies a stronger constraint on the latent bottleneck and limits the representation capacity of the latent space z. In conditionally independent generative factors, keeping them disentangled is the most efficient representation. Hence the higher β is, the more efficient the latent encoding becomes and the better the disentanglement. However, a high β may create a trade-off between reconstruction quality and the extent of disentanglement.

4.3.3 VAE based Anomaly Detection

Due to isotropic Gaussian priors on the latent variables, VAEs give representations with disentangled factors [160]. Modeling factors as Gaussians allows each dimension in the representation to be as far as possible from others. Higgins *et al.* [160] added a regularization coefficient that controls the influence of the prior and defined priors as sequential models. Unlike an Auto-Encoder, prior gives significant control over how a VAE models a latent distribution, as expected from a Bayesian model. Precise modeling can capture better representations, as shown in work presented by Chung *et al.* [161].

As seen in Algorithm 1 during training, one needs to use both the encoder, f_ϕ, and decoder,g_θ, models to minimize the reconstruction loss as well as latent loss. During testing, only the encoder is used to get the extracted bottleneck, and then the latent features

Algorithm 1 VAE anomaly detection. Adapted from [153, 12]

Input: Benign Training Data $\mathbf{X}$, Real-World Data $\mathbf{x}^{(i)}$ $i = 1, \ldots, N$, *Threshold* α

Output: reconstruction probability $p_\theta(\mathbf{x}|\mathbf{x'})$

$\phi, \theta \leftarrow$train a VAE using the benign Dataset $\mathbf{X}$

for $i \leftarrow 1$ ***to*** N **do**

  $\mu_{\boldsymbol{z}}(i,l), \sigma_{\boldsymbol{z}}(i,l) = f_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})$

  Draw L *samples from* $\boldsymbol{z} \sim N(\mu_{\boldsymbol{z}}(i), \sigma_{\boldsymbol{z}}(i))$

  for $l \leftarrow 1$ ***to*** L **do**

    $\mu_{\boldsymbol{x'}}(i,l), \sigma_{\boldsymbol{x'}}(i,l) = g_\phi(\boldsymbol{x}|\boldsymbol{z}^{(i,l)})$

    $reconstruction\ probability(i) = \frac{1}{L}\sum_{l=1}^{L} p_\theta(\boldsymbol{x}^i|\mu_{\boldsymbol{x'}}(i,l), \sigma_{\boldsymbol{x'}}(i,l))$

    if $reconstruction\ probability(i) \leq \alpha$ **then**

      $\boldsymbol{x}^{(i)}$ *is an anomaly*

      else

        $\boldsymbol{x}^{(i)}$ *is benign*

are sampled following a Gaussian distribution, and its standard deviation and mean are fed to the decoder to get the reconstruction probability for this data point, which then can be checked if above a certain threshold, and the data point is judged anomalous or benign [153, 12].

VAE anomaly detection has been utilized in the cybersecurity context by Bernieri *et al.* [162], as well as others as seen in Chapter 3. However, their application in cybersecurity literature has been limited, and there is still a lot of room for their expansion to different use cases. The low footprint of VAE in cybersecurity comes as an opportunity to evaluate the technology in some of the interesting problems in cybersecurity that rely on anomaly detection. According to Yao *et al.* [163], VAE-AD performs better than other anomaly detection approaches such as AE and Kernel Principal Component Analysis (KPCA). We recognize the importance of feature extraction in anomaly detection, and Auto-Encoder has been a great feature extractor, arguably the best in the past years. We understand, however, that AE has limitations in its ability to generalize well and is not able to easily extrapolate beyond the dataset used in training. These issues are addressed in VAE's

capability to capture the underlying distributions of the input data. Thus, they can work more effectively with smaller training data sets and can yield better anomaly scores than other feature extractors such as AE and PCA.

4.4 Summary

Anomaly detectors use a very large variety of models, including density-based, Bayesian networks, fuzzy-logic, deviations from association rules, HMMs, cluster analysis, NN, AE, LSTMs, SVM, sub-space, or tensor-based outlier detection. The superiority of the VAE model, however, lies in the generative property, making them cheap, efficient, not data-hungry, and generalizable. Exploiting the advances in variational Bayes and in probabilistic inference in general, VAEs and their variants conditional VAEs carry the potential to outperform most ML-based anomaly detectors [164]. Hence we include VAE, with KNN, LOF, HBOS, CBLOF, ABOD, FeatureBag, IFeature, PCA, MCD and, OCSVM in our evaluations.

Chapter 5

Experiments

This chapter describes the implementation of approaches presented by Centeno *et al.* [20] which is revisited by Buech [13] as well. In this method, a Siamese Convolutional Neural Network is used along with an OCSVM on the HMOG dataset, in a manner that allows for a near real-world test case. We present the implemented evaluations for linear models like PCA, OCSVM, MCD, HBOS, KNN, LOF, and CBLOF, Probabilistic like ABOD, and ensemble like iForest, and FeatureBag on a few novel and classic experiment configurations reported, span a variety of parameters, scenario steps, and performance metrics, to satisfy our use cases and comparability requirements. It is important to note that in the realistic experiments, noted as VALID experiments, the accelerometer was the only data type used to avoid any fingerprinting of surroundings in the magnetometer or gyroscope data. We selected our models based on our methodology that was refined and detailed in Chapter 4. This chapter benefits from the related works and uses the concepts visited in previous chapters upon which we base our experiments, implementations, and conclusions. Implementations details are captured carefully along with curated designs and codebase in Chapter 6.

5.1 Data Processing

5.1.1 Initial Data-set Explorations

To evaluate our work, we use the HMOG [24] dataset, which provides the smartphone data collected from many users while they were performing various activities (*cf.*, Section 3.1). We train and evaluate our models on these sessions. The current state-of-the-art [20] uses a Siamese Convolutional Neural Network to extract deep features from the inertial sensors data (magnetometer, accelerometer, and gyroscope) per every one-minute session and passes it to a One-Class Support Vector Machine to detect if the current user is the real owner.

The dataset, as outlined in Chapter 3, is used as a de-facto in our research domain, due to its thorough data collection procedure detailed by its authors [14, 24] and publicly available for researchers to use [165]. The authors developed a data collection tool for Android phones to record real-time touch, sensor, and keypress data invoked by user's interactions with the phone. Data from three usage scenarios on smartphones were recorded: (1) document reading, (2) text production, (3) navigation on a map to locate a destination.

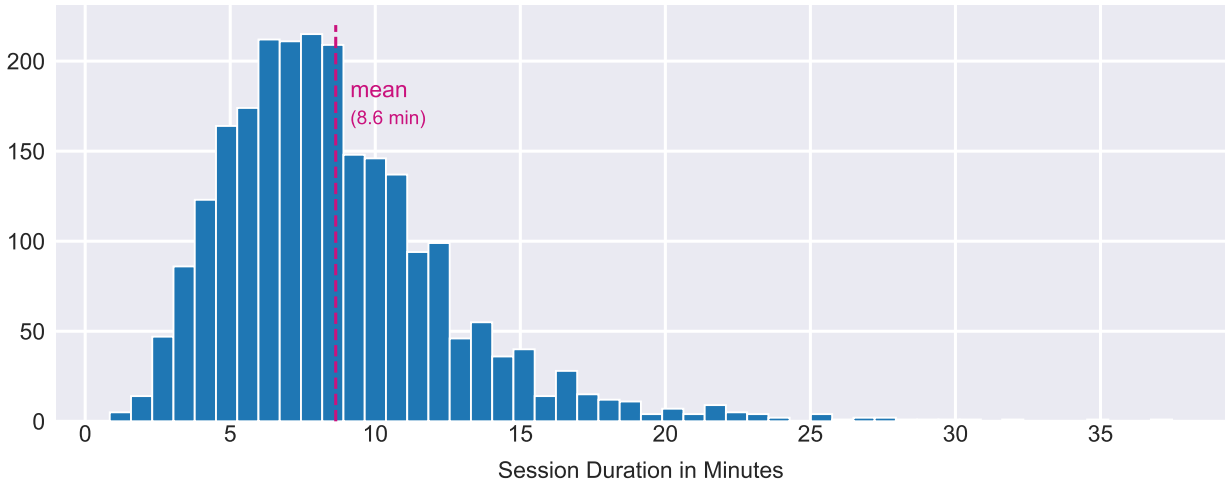

Figure 5.1: HMOG sessions duration. Source: [13]

Category	Content
Accelerometer	Timestamp, acceleration force along X/Y/Z-axis
Gyroscope	Timestamp, rate of rotation along X/Y/Z-axis
Magnetometer	Timestamp, ambient magnetic field along X/Y/Z-axis
Raw touch event	Timestamp, finger count, finger ID, raw touch type, X/Y coordinate, contact size, screen orientation
Tap gesture	Timestamp, tap type, raw touch type, X/Y coordinate, contact size, screen orientation
Scale gesture	Timestamp, pinch type, time delta, X/Y focus, X/Y span, scale factor, screen orientation
Scroll gesture	Starting and ending timestamp, X/Y coordinate, and contact size; speed along X/Y-coordinate, screen orientation
Fling gesture	Starting and ending timestamp, X/Y coordinate, and contact size; speed along X/Y-coordinate, screen orientation
Key press	Timestamp, press type, key ID, screen orientation

Figure 5.2: Nine categories of touch or sensor data are recorded in HMOG. Source: [14]

They had 100 volunteers log into the data collection tool, where each subject was randomly assigned a reading, writing, or map navigation session. For each session, the subject would either sit or walk to finish the tasks. Each session lasted about 5 to 15 minutes, with a mean of 8.6 minutes as depicted in Figure 5.1. Each subject was asked to perform 24 different sessions: four sessions reading while sitting, four sessions reading while walking, four sessions writing while sitting, four sessions writing while walking, four sessions navigating a map while sitting and four sessions navigating a map while walking.

In total, each subject contributed about 2 to 6 hours of behavior traits [165]. The data recorded included streams of accelerometer, gyroscope, magnetometer, raw touch event, a tap gesture, scale gesture, scroll gesture, fling gesture, and keypress on the virtual keyboard, as in Figure 5.2. We decided to limit ourselves to initial data features only (accelerometer, gyroscope, and magnetometer) as discussed in Chapter 4, for reasons pertaining to power, privacy, and security.

We mainly utilize the software put forward by Buech [15] in their re-implementation of Centeno *et al.* [20], data processing step, to achieve the necessary preprocessing and address several issues with the data set, to enhance its effectiveness in our experiments. Buech [15] identified sessions 9, 10, 11, 12, 13 and 14 of subject 733162 were missing accelerometer data and subjects 526319 and 796581 had only 23 sessions instead of 24, as seen in Figure A.1 and Figure A.6. We hence eliminate subjects 526319, 796581, and 733162 from our working dataset moving forward. Li *et al.* [69] also reported two subjects in HMOG to have *unusual* data, but did not explain further. We assume these are subjects 526319 and 796581. Centeno *et al.* [20] excluded 10 subjects from the HMOG dataset for their work. Others [24, 66, 19] did not mention any such data cleaning steps, like excluding any of the subjects' data, in their use of HMOG. In an effort to be comparable with Centeno *et al.* 's work [20] seven additional subjects that shown irregularities in data sizes are removed. Following Buech's [15] reasoning, we remove subjects 256487, 389015, and 856401 for having too much data compared to the mean, as seen in Figure A.6 and 219303, 539502, 737973 and 986737 for having too little.

Another issue that is worth noting and will be visited later in our experiments, is the fact that magnetometer data had some significant outliers, especially on the z-axis, as shown by Buech [13]. These anomalies were highly related to subjects, and hence can be used as a shortcut for any model to learn how to spot a certain user. We suspect that's due to the environment in which the data was captured. We discuss that in more detail later on in our evaluations. As for other sensors, accelerometer data was skewed due to gravity on the y and z axes, and gyroscope data was evenly distributed around zero. We scrutinize the usefulness of magnetometer in our evaluations moving forward, and will aim to evaluate our models without that, for a more realistic test case and results. Based on additional data explorations, we also made the empirical observation that the inertial

data only is distinctive enough between subjects to allow for an authentication system to perform reasonably well. As displayed in Figure A.4 and Figure A.5, this is fairly visible in the visualizations even when eye-balling the data. It is important to note that the accelerometer alone, also seems to be distinctive enough, as seen in Figure A.2, which will be validated in our evaluations as well. Further HMOG statistics and observations are to be found in Appendix A.3.

5.1.2 Data-set Preparations

The CSV files from the dataset had to be converted into tables in a single file of the Hierarchical Data Format (HDF) to enhance the performance and speed of our experiments on the supercomputer. It is worthwhile to note that HDF files do not support lazy operations. Hence, it is not possible to load a data instance only when needed, and the whole dataset needs to completely fit in memory. If the computer resource does not have enough memory to support all of HMOG to be loaded into RAM, then a library like Dask [166] might be the right choice.

Buech [15] proposed an initial data set transformation process that has been adopted in our work, as well. They collect the inertial sensors (accelerometer, gyroscope, and magnetometer) data streams at 100Hz frequency, and joined them from all HMOG spreadsheets. Unnecessary attributes and features were removed, and the results are saved in the Hierarchical Data Format, for faster data reading operations, as seen in Figure A.3.

As the accelerometer, gyroscope, and magnetometer sensors data is not uniformly sampled, it would generate time shifts between the entries if the data would be joined on row indices. So the data is transformed into a uniformly sampled time-series and joined under one table, after re-sampling each sensor data to ten milliseconds and linearly interpolating the gaps. Sequences with incomplete entries at the beginning and end of their span were shortened.

Information and metadata about the sessions performed are injected into the data for every measurement reading the session ID. It identifies whether a session was capturing the subject reading while sitting, reading while walking, writing while sitting, writing

while walking, navigating a map while sitting, or navigating a map while walking. We identified these six task-body modes combinations from the *Activity.csv* file of the HMOG dataset, although some gaps in activity descriptions were found. We handled activity description gaps by simply assuming that every session during data collection had a single task-body mode combination. Hence, if we have that activity labeled for any time interval in the session, we can assume the label identifies the entire session's duration and all of its measurements.

With regards to sampling, Centeno *et al.* [20] used both 100Hz and 25Hz for their Siamese CNN approach, but reported better results for 25Hz. It was unclear how Centeno *et al.* achieved that, so Buech [15], during re-implementation, calculated the mean of every four samples, using a sliding window, as seen in Figure A.3. In general, Centeno *et al.* [20] provided very limited information about their data normalization procedure. For instance, the authors claim to perform channel-wise MINMAX normalization into the zero to one range, but they do not describe how that was applied. It could be applied to the whole dataset at once, for every subject individually, for each session, or even for every window of samples, [13]. Other works, even earlier works by Centeno *et al.* [16], did not clarify what normalization techniques were used, if any, making this component's implementations ambiguous. Hence, we are left to experiment and yet to find a proper method.

Through Buech's [13] correspondences with Centeno *et al.* [16, 13], they concluded that no differentiation was made between normalization of training and testing sets in Centeno's work [16, 20], so as most of the works that use HMOG [24, 14]. Centeno's *et al.* [16, 20] normalization was performed subject-wise for OCSVM, and Siamese CNN. Normalizing should not happen before splitting the dataset into training and testing subsets. Fitting a scaler on the training set then normalizing both training and testing sets using the same scaling parameters would avoid leakage of information between the sets. The same scaling and normalization need to be used for benign and malicious instances and all subjects alike since in real-world deployments we can not tell the difference.

Hence following Buech [13], we let normalization become a degree of freedom in our experiments and aim to find a realistic experiment setting, unlike what we can see in the research corpus utilizing HMOG as well as other datasets for continuous authentication.

5.2 Experiments Design

Simply, we mimic scenarios were the model on the owner's phone is trained to detect the owner, and then suddenly another user starts using the phone. We try to mimic that scenario accurately by removing any data that can fingerprint the surrounding, and only focus on data that signifies the user behaviour, not the transition between users. We also test every owner against all possible imposters, and with different durations and sessions time. The data was initially collected on the same phone and was labelled with the user and the activity, and hence allows us to surgically patch them together smoothly to recreate experiments that mimic the various threat models and use cases introduce in Chapter 2.

We randomly split, five times using different randomness seeds, the 90 subjects into a group of sixty subjects representing our validation data set for hyper-parameter tuning purposes and another group of thirty subjects for testing and performance evaluation purposes. Our splitting technique is similar to that of Buech [13] to remain comparable.

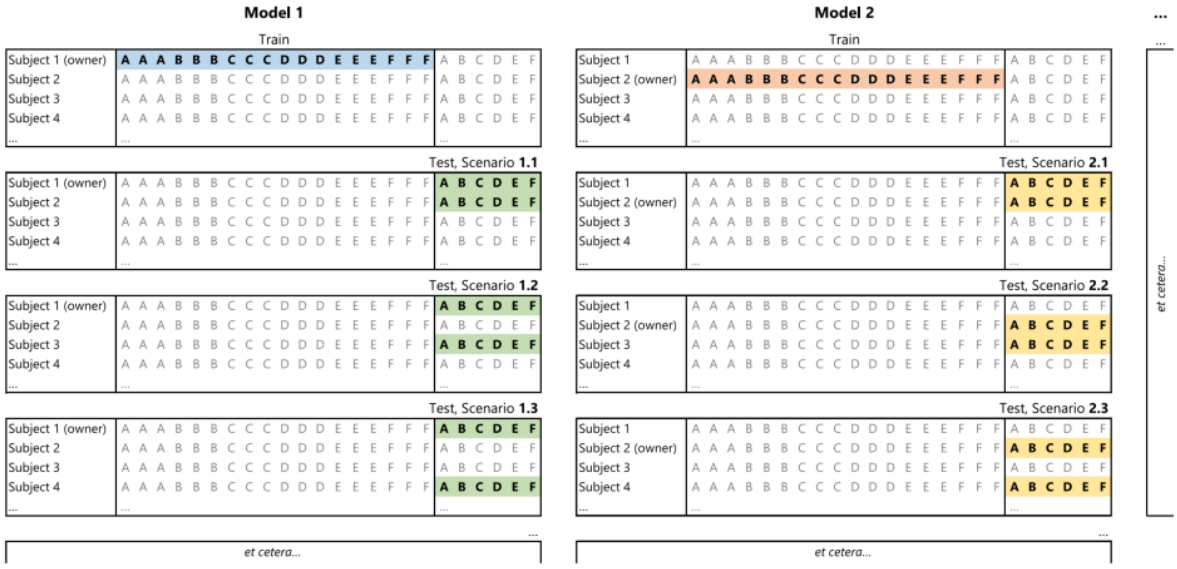

Figure 5.3: Schema of data splitting for training and testing. A,B,C,D,E,F refer to the six different tasks-body modes combinations. Every subject is once selected as owner and tested against all remaining subjects. Source: [15]

Both of these sets are further split into training as well as validation or training subsets.

Similar to what is seen in Figure 5.3, in every experiment a subject is selected, as the device owner and models are trained on its training sessions. Pairs of owner and impostor samples from the validation as well as testing sessions are generated and selected for the experiment. Note that since there are sixty subjects in the validation dataset, we train sixty different models during hyper-parameter optimization. Once a hyper-parameter value set has been concluded, then the validation dataset is put aside, and the testing with the testing dataset begins.

Centeno *et al.* [16] thoroughly explain how their cross-validation was performed for training and testing. The cross-validation employed in our evaluations is identical to that used and re-implemented by Buech [13]. We also added a validation split to allow for a dedicated hyper-parameter tuning and optimization component in the experiments. It is important to note that hyper-parameter tuning happens for all subjects *i.e.*, the model is generalized for as many users as possible. For future work we recommend the usage of individual specialized models, hyperparameter tuned and optimized for each subject or owner (*cf.*, Chapter 7). Hyper-parameter tuning for each subject would undoubtedly enhance the results and is the way to go for a real-world scenario as described in Chapter 4 but makes our work hard to compare to others. Our work prioritized comparability to the existing body of work, to evaluate a variety of models and specifically what generative models, like the VAE, can offer to a continuous and implicit authentication system.

There are many questions and variables regarding the experiment design. For instance what feature set to use of the data available? Are we going to utilize deep feature extractions or use raw features? Is the collected data presenting a fair evaluation grounds for the designed solution? We decided to classify all these degrees of freedom regarding the design, in a table, and logically choose what to experiment with. We tweak them to make sure we do not have any biases or questionable results in our evaluations. The results are shown in Table 5.1. The NAIVE setting refers to a configuration where the data is initially normalized feature-wise per subject using the MINMAX algorithm, before splitting into training and testing sets. This setting obviously introduces bias into our testing subset and is not realistic, since the owner and an impostor cannot be distinguished upfront. As discussed before, normalization needs to take place using the same parameters, and hence the VALID setting is where the dataset is split into training and testing subsets upfront.

Degrees of freedom and Initial Settings				
Experiment Name	**Data**	**Normalization**	**Scaling**	**Features Extraction**
VALID-ROBUST-FCN	accelerometer 5s windows (walking)	Valid	Robust	Deep via FCN
NAIVE-MINMAX-2D	accelerometer, gyroscope, magnetometer 1s windows	Naive	Minmax	Deep via 2D CNN
VALID-ROBUST	accelerometer 5s windows (walking)	Valid	Robust	Raw
NAIVE-MINMAX	accelerometer, gyroscope, magnetometer 1s windows	Naive	Minmax	Raw

Table 5.1: Different initial experiments settings

Then, the normalization scaler is fit using the owner's training data. Finally, the same fitted scaler is applied to the testing samples for both owner and impostor.

We exhausted all combinations looking for what effects are observed. Buech [13] indicated that NAIVE-MINMAX-2D, using the accelerometer, gyroscope, and magnetometer data types, is what is used by other researchers and is how we can benchmark other works, most importantly that of Centeno *et al.* [20]. The author also proposed that VALID-ROBUST-FCN, using only the accelerometer data type, is a more realistic experiment to see how good are solutions performing in close to realistic scenarios. The arguments were that NAIVE is not realistic as it had normalized on malicious actor data which really would not be available during training.

Buech [13] also argued that magnetometer data in HMOG was very different between different sessions, to the extent that it can fingerprint certain locations the dataset was collected in and ends up being a shortcut for the model to achieve better performance,

utilizing this bias. He found that the gyroscope does not allow the model to achieve better performance and decided to remove it, too. As all related works Chapter 3 had shown that models perform better in walking body modes since the user profile is easier to detect, Buech [13] decided to omit the seated body mode from a few experiments, and claimed that improved the performance.

We were able to validate these claims and tried all the combinations, especially those missing from his evaluations which were using accelerometer and gyroscope, and accelerometer and magnetometer with all the other combinations. Our results indicate that there is no gain from including any other data types to the VALID-ROBUST-FCN experiment.

In our work, we found that the two that stood out are one that was composed of NAIVE-MINMAX and another that was composed of VALID-STD. We evaluated with deep feature extraction and without, and VAE, as well as KNN, have consistently been the best two models. We found that KNN is very interpretable and VAE was very fast in prediction time. Interpretability is not important in the solution design but fast prediction times were of pivotal importance, as well as being consistently resulting in low EERs and high accuracy scores across the test in all kinds of experiment settings, with and without deep feature extractions.

5.2.1 Deep Feature Extraction

Computing features and engineering feature sets involved, traditionally, a lot of manual labor. As seen in Table A.2,Table A.3, and Table A.4 there are plenty of features a data scientist can choose for any particular project. Deep feature extraction allows for deep learning techniques and architectures to learn representations that can replace the feature engineering process.

In the state-of-the-art ensemble solution presented by Centeno *et al.* [16] a Siamese CNN is trained to generate deep features that are passed on to train an OCSVM as an authentication model. The SCNN network consists of two separate subnetworks, that share the weights, but each has its input vector and a single output. The output value is derived from the distance between the output of the subnetworks and relates to the similarity of the two input vectors.

In the SCNN pairs of input vectors $x(i)$ and $x(j)$ are paired to a label $y(i)$ that is either positive, if $x(i)$ and $x(j)$ are related to the same device user, or negative, if $x(i)$ and $x(j)$ are related to different device users or profiles.

Both CNN's reduce the dimensionality of $x(i)$ and $x(j)$. These vectors of lower dimensionality that result from the siamese network have a distance d between them. Now contrastive loss function L is given d and $y(i)$ and outputs a high a loss value, only if d is low and $y(i)$ is less than zero, or d is high, and $y(i)$ is greater than zero. On the other hand, if either d is low and $y(i)$ is greater than zero, or d is high and $y(i)$ is less than zero outputted loss becomes low. As indicative of the behavior modeled, the network is forced to learn aspects of the data that are useful to distinguish between samples from the same user and samples from different users, effectively doing feature reduction and selection in an automated fashion.

Once the SCNN is fully trained with both the same user and different user pairs from a labeled training subset, any one of the identical CNNs is used to output the lower-dimensional features only. The OCSVM makes use of these for its training and testing. This completely removes the feature construction tedious process and was shown to improve the OCSVM performance when compared to training on raw features.

Buech [15] implemented the deep feature extractor in a manner that is slightly different from what Centeno *et al.* [16] described to not introduce bias into the model. The key difference is 50 instead of 60 subjects remained for training the Siamese CNN, as 10 subjects are used as a validation set for the OCSVM, and still 30 for subjects remain completely unknown to the whole ensemble until the testing phase. This made sure the experiment setup is more realistic and the results will be more representative of what can be found in a real-world deployment.

The SCNN deep feature extractor architecture, shown in Figure A.10, was re-implemented [15] to match the parameters described by the original authors and Buech in Table A.5. All layers had Rectified Linear Units (ReLUs) as their activation function, first to implement ReLUs as activation functions were Krizhevsky *et al.* [167] in AlexNet, their architecture was one of the largest convolutional neural networks on the subsets of ImageNet. We encourage future work to optimize this architecture and explore deeper models as making a

network larger usually improves the performance [168]. When ReLUs caused dying units, the author has reported exchanging them for exponential linear units.

The way the input to the SCNN was made was through halving all the subjects' data, shuffling them, and aligned as pairs for the network. It is important that the SCNN was implemented in three variations by Buech [15], namely CNN with 2D filters, detailed in Figure A.10, CNN with 1D filters, detailed in Figure A.8, and FCN with 1D filters, detailed in Figure A.9. The reasoning is the architecture makes more sense with 1D filters, so the 2D mentioned by Centeno is interpreted as a possible typo, and the last layer is dense is common instead of flattening the final pooling layer. These variations gave the experiments more degrees of freedom, as seen in Table A.6.

Further details of the SCNN implementation are made available in the original authors' publication [20] and the recent re-implementation [15].

The outlier detection models we choose to test in the experiments are all of the ones discussed in Chapter 4. A brief description of those is in Appendix A.5. Their implementation details are in Chapter 6.

5.3 Results

As we conducted more than ten experiment settings (VALID FCN ROBUST, VALID FCN STD, VALID FCN MINMAX, NAIVE ROBUST 2D, NAIVE STD 2D, NAIVE MINMAX 2D, VALID ROBUST, VALID STD, VALID MINMAX, NAIVE ROBUST, NAIVE STD, NAIVE MINMAX) across our more than ten models with numerous hyper-parameters and feature set combinations (accelerometer, gyroscope, magnetometer), we generated lots of results. Commented and clean code base and data from these available on our GitHub repository [169]. In this section, we discuss and present the main highlights and relevant insights.

To evaluate the authentication performance, cross-validation was performed using the *testing* subset, which involved thirty subjects who have not previously included invalidation and have never seen by the models or the data pipeline. The experiment results

are interpreted using calculated EERs, and performance accuracies for all the intricate owner–impostor scenarios that play out in these cross-validations. In each cross-validation, one of the 30 subjects is treated as the device owner and only 18 of their sessions are used to train the model. Six different owner sessions, each with a unique task body mode combination of the owner, are then tested against six sessions of some other user, playing the impostor, with an equal number of testing data instances and samples. This process is repeated for all 29 imposters, against any single owner. A total of 870 such crafted testing scenarios were simulated. To improve the robustness of our results, this evaluation was repeated five different times using different random seeds. The presented results are the average of the five independent trials, or 4530 testing scenarios, each consisting of six tests from an owner and six tests from an imposter. This means that every owner got to have 29 imposters attack their phone for six sessions five times each.

As detailed in Section 5.2, the NAIVE approach to normalization was initially performed, both on the entire data and individually for every subject. However, in the VALID approach, the normalization scaler was fitted on the training data of the owner only and then applied to transform the testing data for both the owner and the imposter.

5.3.1 Initial Results

Our initial results are based on the NAIVE MINMAX 2D experiment, where normalization occurs naively and simplistically. In this set-up, testing dataset values are MINMAX scaled, and the SCNN uses the 2D CNN implementation discussed in Section 5.2.1. The codebase for most of the experiment instruments was open-sourced by Buech [13] in their implementation and validation of the state-of-the-art ensemble model proposed by Centeno *et al.* [16]. Their self-reported results, under the NAIVE MINMAX 2D setting, are 14.70% EER and 90.00% accuracy, which are produced by running an ensemble of the author OCSVM model after deep feature extraction using a Siamese Convolutional Neural Network. We aim to enhance these scores, thanks to the properties generative models inherently possess, as discussed thoroughly in Chapter 4.

Our results were based on running the following models through the NAIVE MINMAX 2D experiment: VAE, β-VAE, KNN, OCSVM, author OCSVM (proposed by Centeno *et*

al. [20] and reimplemented by Buech [15])[1], ABOD, CBLOF, FeatureBag, HBOS, IForest, LOF, PCA. Note that we experimented with different VAE variations and configurations in this step, to find the best architecture and hyper-parameters to proceed with. We also conducted hyper-parameter tuning for all models, for both the VALID-FCN-ROBUST and the NAIVE-MINMAX-2D experiments. This was done via the Random Search algorithm [170] which is faster and more efficient than the Grid Search algorithm [171]. As discussed in Chapter 2, EER is the preferred metric for this study, but model accuracy is also included to improve comparability. Our experiment results are summarized in Figure 5.5 and Figure 5.4, in both metrics. The timing results for the experiments are shown in Figure 5.6. The results shown under each model, represent the best possible performance reachable by the model when it is optimized for the experiment settings.

[1]OCSVM refers to an OCSVM model that we hyper-parameter tuned for optimal performance in our experiments, while author OCSVM refers to the model hyper-parameter tuned by Centeno *et al.* [20] and re-implemented by Buech [15] for the the various experiment settings

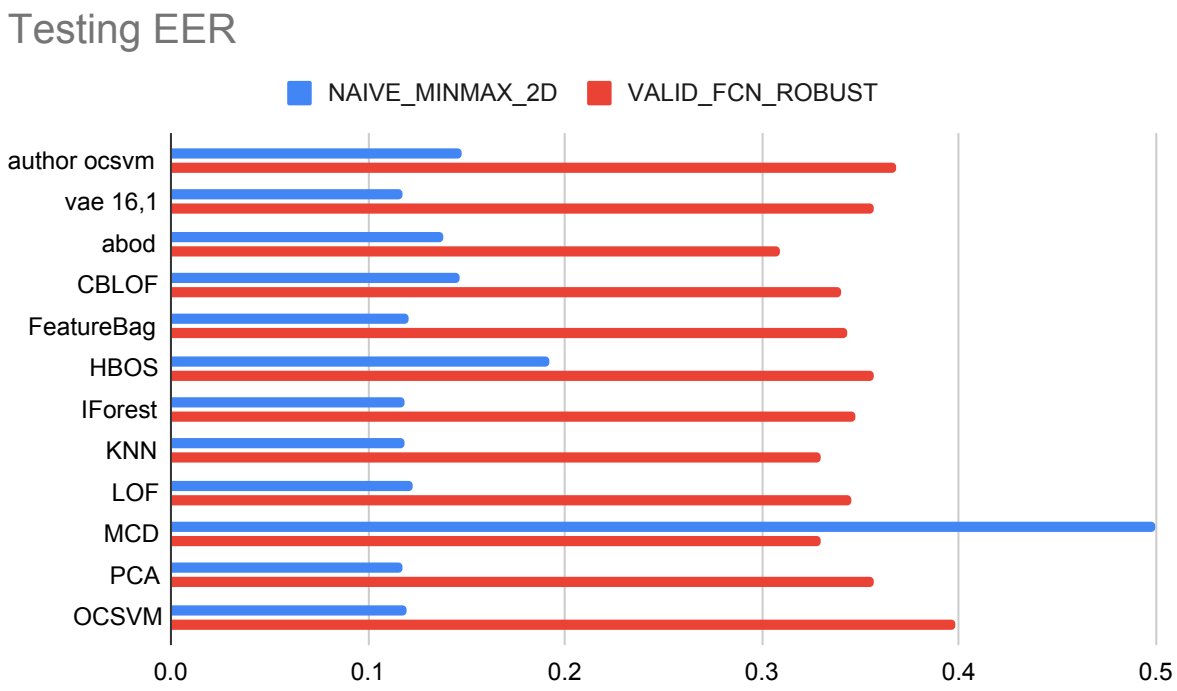

Figure 5.4: Testing EER of 12 models in 2 different experiment settings.

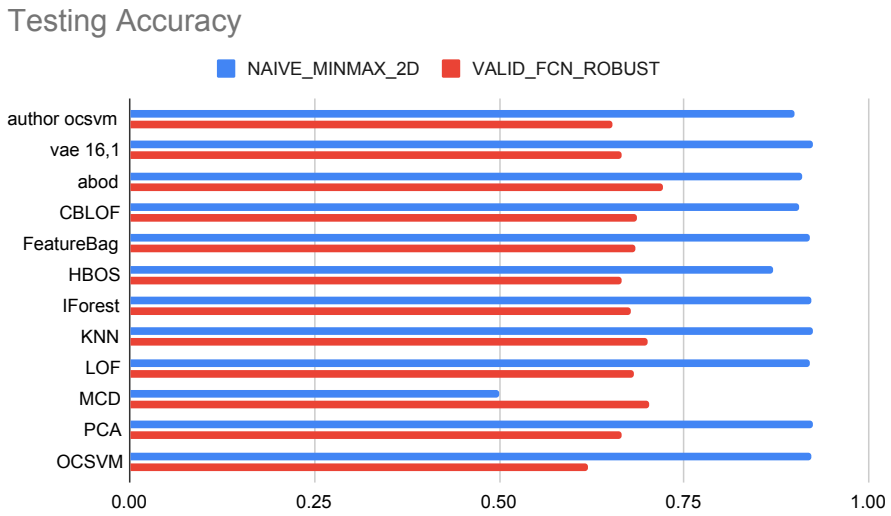

Figure 5.5: Testing accuracy of 12 models in 2 different experiment settings.

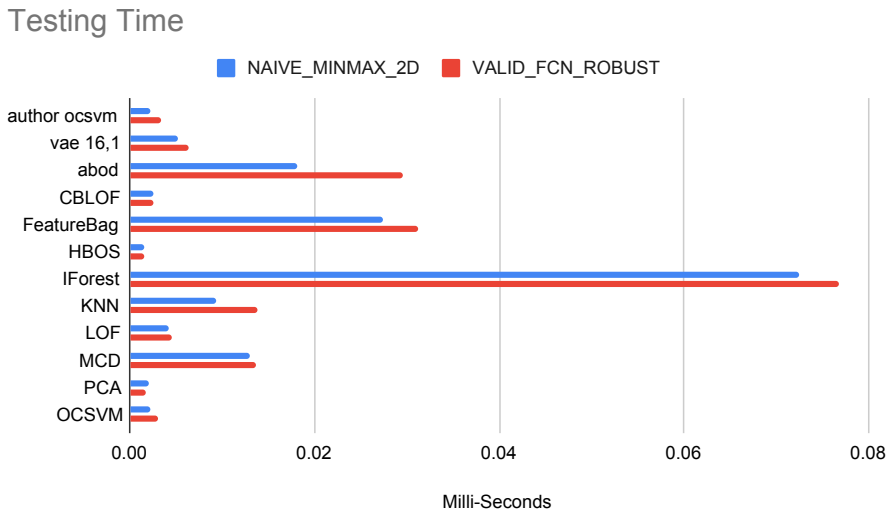

Figure 5.6: Testing results of 12 models in 2 different experiment settings. Scoring time reported in seconds.

5.3.2 Promising Models

Based on the initial results, we disqualify most of the models for being slow, measured by inference time, or having sub-optimal performance, measured by EER and accuracy. This leaves us with VAE, ABOD, KNN, LOF, and PCA only, as the most promising five models, as in Figure 5.7. The initial results show us that although all models are performing relatively well, the models mentioned above achieve the best mean accuracy and EER with the lowest mean scoring time.

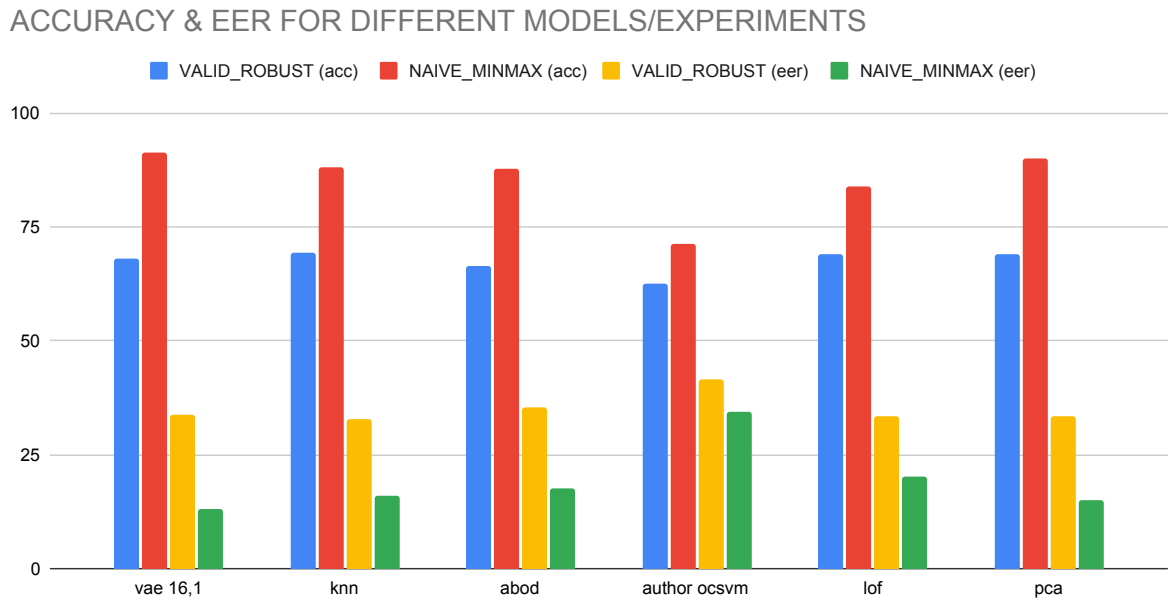

Figure 5.7: Previously reported testing accuracy and EER of the state-of-the-art and 5 best models in 2 different experiment settings.

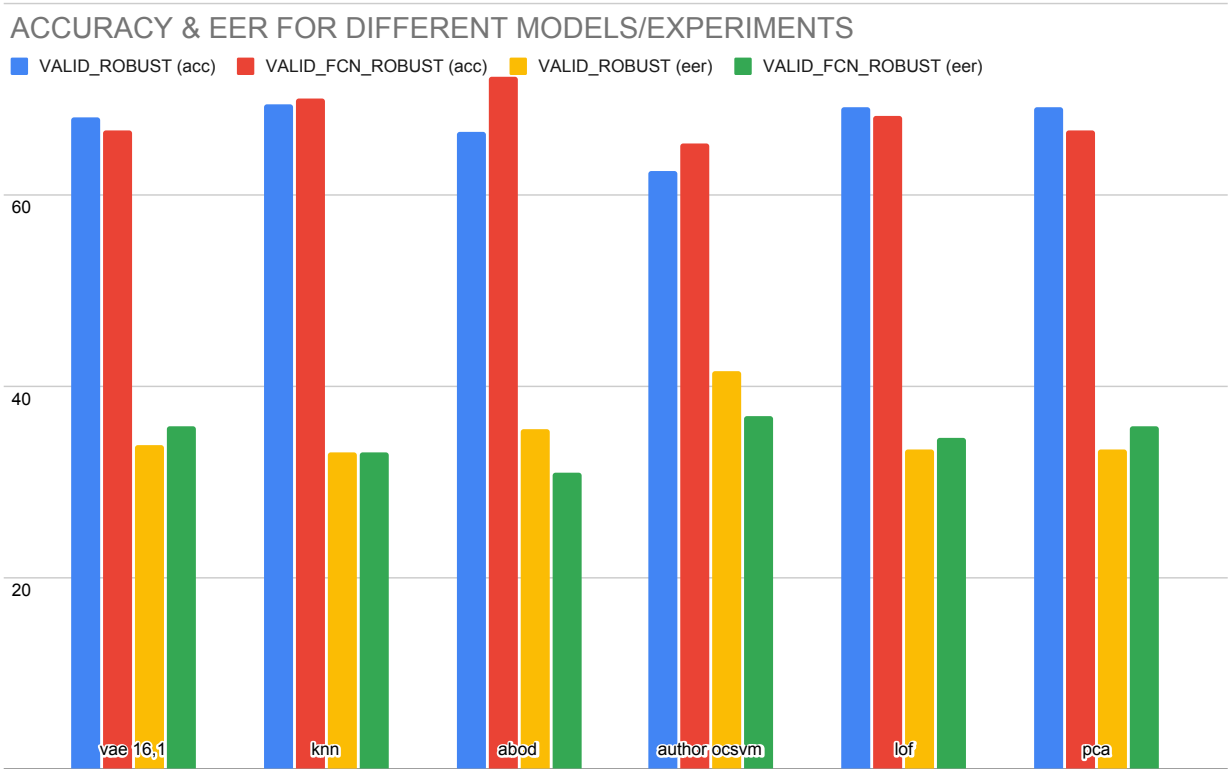

Figure 5.8: Previously reported testing accuracy and EER of the state-of-the-art and 5 best models in 2 different experiment settings.

In Figure 5.9 and in Figure 5.8, the impact of the deep feature extraction step using the Siamese Convolutional Neural Network on the performance of these promising models are analyzed. It is apparent that the usage of such a resource-intensive step causes significant performance gains from some models, like the OCSVM, but negligible performance gains for others, like the VAE.

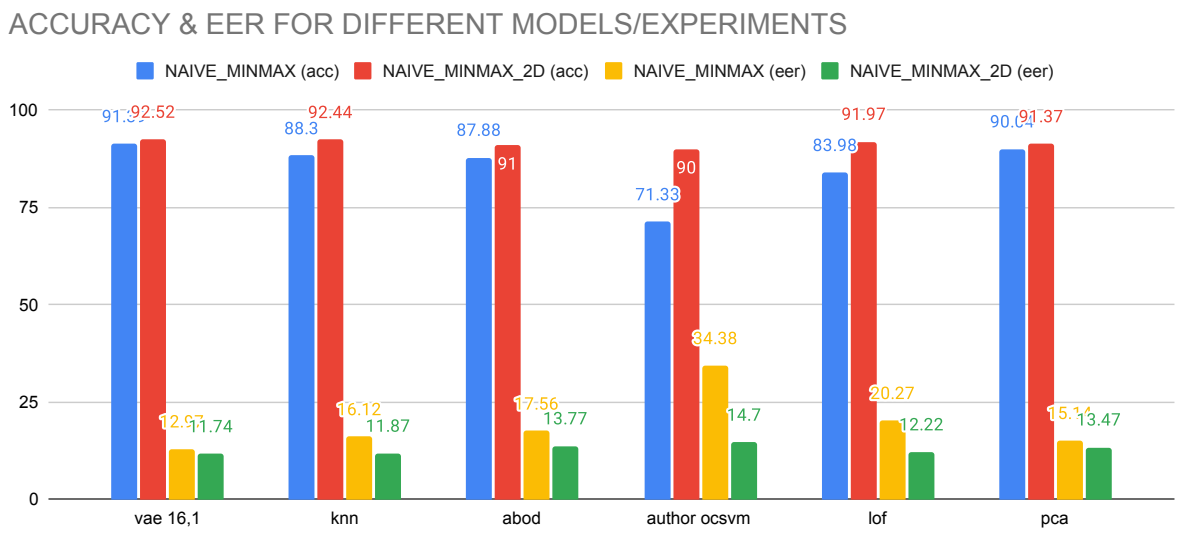

Figure 5.9: Testing accuracy and EER of the state-of-the-art and 5 best models in 2 different experiment settings.

It is worth noting that the loss due to the removal of the deep feature extractor is only 1% in accuracy for the 2 layered VAE, with 16 neurons in the first layer and 1 neuron in the second, experiment while it ranges between 4% to 20% in other models, and that can be attributed to the VAE's inherent dimensionality reduction capabilities, thanks to its autoencoder and generative models roots.

Under the same experiment settings (NAIVE MINMAX 2D), while SCNN and OCSVM ensemble reached 14.7% EER and 90% accuracy, our proposed model achieved 11.71% EER and 92.52% accuracy when using a VAE with only 2 layers of size 1 and 16, respectively, amassing a 3% gain in EER, our most prominent performance metric. In these experiments, we also found a KNN configuration that produced 11.87% EER and 92.44% accuracy. ABOD also achieved 13.77% EER and 91% accuracy.

However, the VAE had a mean scoring time of 0.005 seconds and negligible variance. Furthermore, it is worth noting, as seen in Figure 5.10, that the VAE is 8-10 times faster in scoring than the KNN or the ABOD, who also have good results. That is comparable to the OCSVM scoring time without the Deep feature extractor SCNN. We recognize that inference speed is very important in this solution. The user experience and utility of the

solution are highly affected by the scoring time, and resource limitations are common on smartphones.

The experiments demonstrate how removing the deep feature extractor, being the Fully Connected Network (FCN) or the 2D network in these experiments, helps us better understand the strengths and flaws of the anomaly detection models. The best and most consistent model across all experiments so far, from accuracy, EER, time mean, and variance standpoints combined is the VAE.

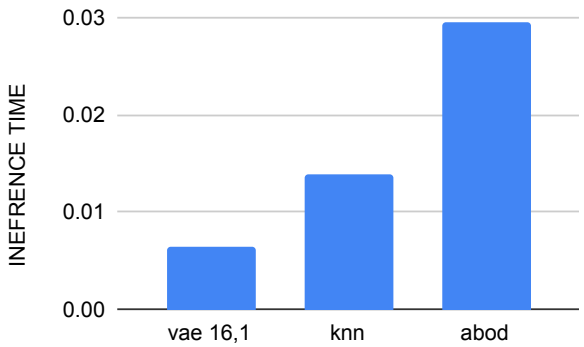

Figure 5.10: Inference time results of VAE, KNN, and ABOD in seconds.

5.3.3 Observations and Improvements

An indication that the models trained on MINMAX scaled data are learning general differences of the distributions as opposed to the intended learned difference of patterns of the actual user movements, is seen in the results of walking only body modes scenarios being the lowest instead of the highest accuracies, which is observed in experiments that use another scaler (*e.g.*, ROBUST or STANDARD), an effect reported by Buech [13]. We turn towards the experiment setting yet again and start questioning whether the VALID-FCN-ROBUST performance can be enhanced, by perhaps changing the scaler. In better-controlled datasets, magnetometer, gyroscope, and other data sources like WiFi or background noise can be used in continuous and implicit authentication systems, however in HMOG fingerprinting physical locations subjects were in, and hence, their recorded activ-

ity sessions is a concern. We remove any data that can be used by the model to fingerprint sessions and ROBUST scale any data we feed the model.

VALID normalization is more realistic than NAIVE since we do not have the attacker data to normalize on in a real-life deployment. An accelerometer is also better than a gyroscope and magnetometer, and any combination of them is worse than an accelerometer alone as we have found within our extensive set of experiments, even when ROBUST scaling is used. However, since ROBUST scaling was used to remove the outliers fingerprinting a small number of sessions (based on unique magnetometer values) that were causing the bias in the models, we aim to validate the need for ROBUST scaling when only accelerometer data, which is assumed to not fingerprint physical locations of data collection, is being utilized.

We now move on to validate why this scaling method was used in particular. To begin with, we look at scaling methods commonly used, summarized in Table 5.2.

Function	*Range*	*Mean*	*Distribution*	*When to Use*	*Definition*	*Notes*
MinMaxScaler	0 to 1 default, can override	varies	Bounded	Use first unless have theoretical reason to need stronger medicine.	Add or substract a constant. Then multiply or divide by another constant. MinMaxScaler subtracts the mimimum value in the column and then divides by the difference between the original maximum and original minimum.	Preserves the shape of the original distribution. Does not reduce the importance of outliers. Least disruptive to the information in the original data. Default range for MinMaxScaler is 0 to 1.
RobustScaler	varies	varies	Unbounded	Use if have outliers and do not want them to have much influence.	RobustScaler standardizes a feature by removing the median and dividing each feature by the interquartile range.	Outliers have less influence than with MinMaxScaler. Range is larger than MinMaxScaler or StandardScaler.
StandardScaler	varies	0	Unbounded, Unit variance	When need to transform a feature so it is close to normally distributed.	StandardScaler standardizes a feature by removing the mean and dividing each value by the standard deviation.	Results in a distribution with a standard deviation equal to 1 (and variance equal to 1). If you have outliers in your feature (column), normalizing your data will scale most of the data to a small interval.
Normalizer	varies	0	Unit norm	Rarely.	An observation (row) is normalized by applying l2 (Euclidian) normalization. If each element were squared and summed, the total would equal 1. Could also specify l1 (Manhatten) normalization.	Normalizes each sample observation (row), not the feature (column)!

Table 5.2: Table of commonly used scaling functions

When using the MINMAX scaler, the maximum and minimum of the distribution have a considerable effect on the normalization, which linearly transforms all values into

a provided range (*e.g.*, zero to one). This makes the normalization sensitive to the many outliers in the dataset. These outliers and their distributions are easily picked up on by the models as a user data session fingerprint but do not signify the user behaviors or movement. We set to explore how the STANDARD scaler can impact our experiment results. In the upcoming section, we explore how the top five models and the baseline OCSVM perform when the data is normalized using the STD scaler rather than the MINMAX or robust.

We also encourage future researchers to explore Power Transformer [172], Quantile Transformer [173] with Gaussian or uniform outputs, and Normalizer [174] scalers. The Power Transformer scaler makes the data distribution fit better to the Gaussian distribution to stabilize variance and minimize skewness as much as it can. The Quantile Transformer, on the other hand, applies a non-linear transformation such that the probability density function (PDF) of the data will be mapped to a uniform or Gaussian distribution. The Normalizer scaler is very different than both in its being a method for rescaling the vector so that every sample would have a unit norm, independently of the distribution of the samples [175].

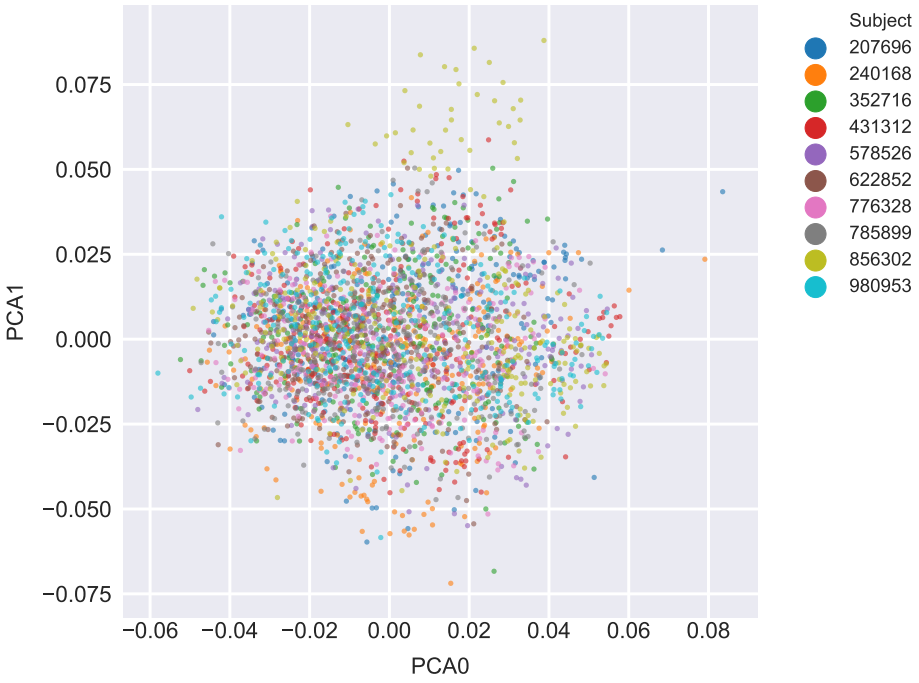

Figure 5.11: PCA visualization of the deep features generated by the original SCNN with 2D filters and using the ROBUST scaler. Source: [13]

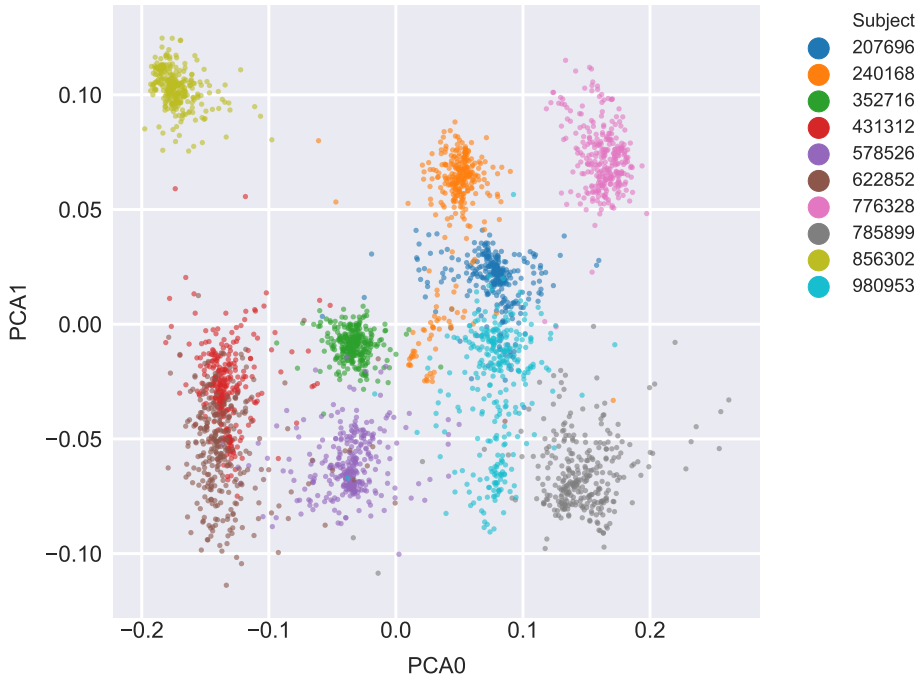

Figure 5.12: PCA visualization of the deep features generated by the original SCNN with 2D filters and using the MINMAX scaler. Source: [13]

The two-dimensional PCA visualization of the deep features, extracted by the SCNN 2D network when using the MINMAX scaler, is shown in Figure 5.12. When compared to the features extracted when using the ROBUST scaler, displayed in Figure 5.11, it is clear that MINMAX scaling allows the models to short-cut the behavioural indicators learning process and have unrealistic performances on HMOG. These inflated performances are simply not indicators of true learning of user-specific behavioural indicators within the data.

Another thing to notice here is that the original OSCVM improved from 65.3% accuracy and 36.8% EER when using data that is ROBUST scaled to 67.4% and 33.36% ERR using data MINMAX scaled when using the VALID experiment setting and the FCN deep feature extractor. This improvement is due to the fact that MINMAX highlights outliers and gives an unfair advantage to the models, as they can fingerprint sessions, rather than capturing behavioural identifiers in the data.

5.3.4 Final Results

We start by introducing the new experiment setting that uses VALID normalization logic, along with STANDARD (std) scaling in place of robust, both with and without the deep feature extraction step. The results presented in Figure 5.13 suggest that this method is beneficial to the performance of most promising models. STANDARD scaling, as we saw in Table 5.2, removes the mean and scales the data to unit variance. accelerometer data does not suffer from too many random outliers like other sensors data, and hence using ROBUST scaling might be doing more harm than good.

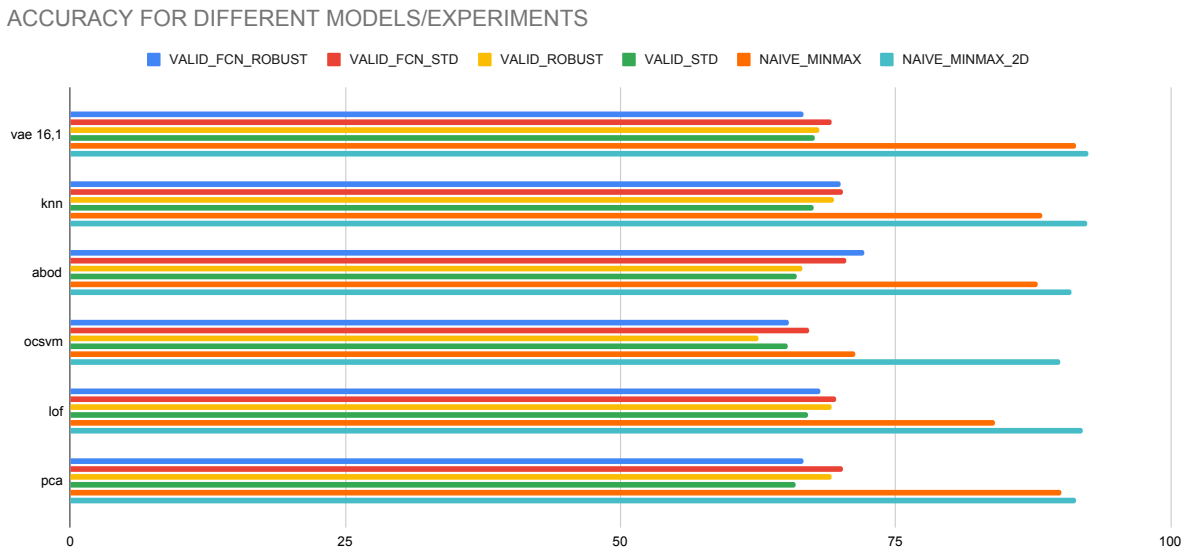

Figure 5.13: Testing accuracy of the state-of-the-art and the top 5 models in 6 different experiment settings.

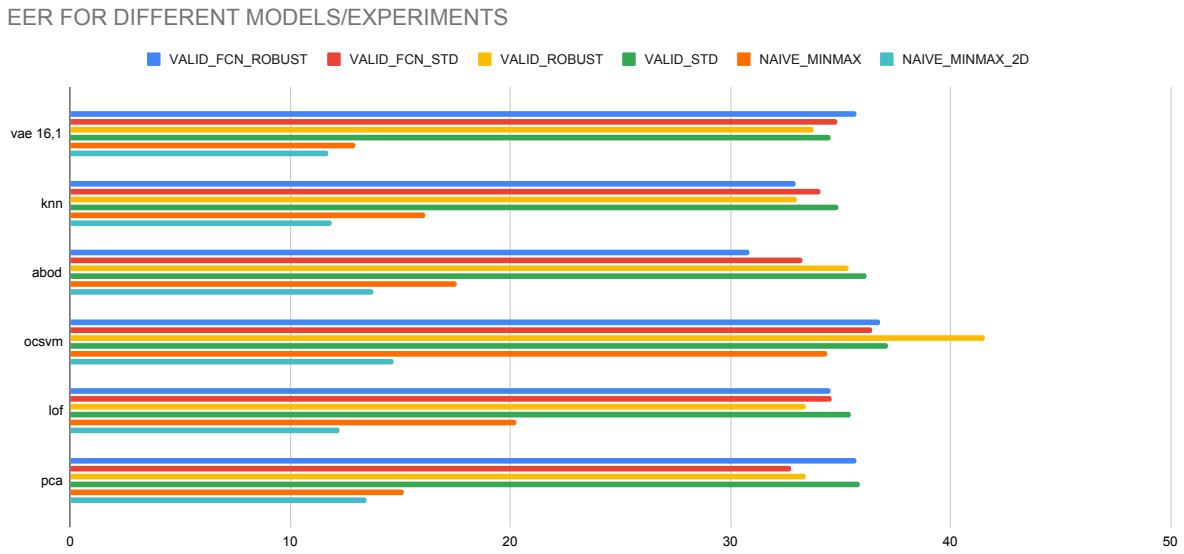

Figure 5.14: Testing EER of the state-of-the-art and the top 5 models in 6 different experiment settings.

We observe the state-of-the-art and the 5 highest-performing models according to the experiments in Section 5.3.1:

- VAE (Reconstruction-Probabilistic Model)
- KNN (Proximity/Distance Based Model)
- ABOD (Probabilistic Model)
- PCA (Linear Model)
- LOF (Proximity/Distance Based Model)
- State of the art: Author OCSVM (Linear Model)

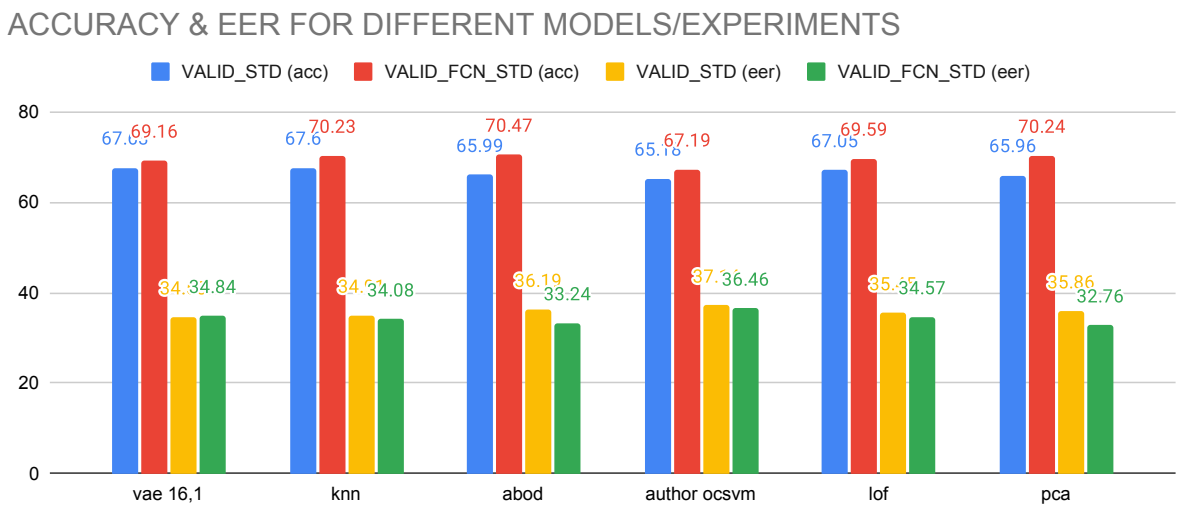

Figure 5.15: Testing accuracy and EER of the state-of-the-art and 5 best models in 2 different experiment settings.

The results of four VALID STD experiments can be seen in Figure 5.15. We find that the ensemble of PCA and SCNN with the FCN architecture has a great performance. However, due to the deep feature extraction's high computational cost, we decide to look into the experiment variant where the heavy, complicated, and slow deep feature extractor is no more present. As displayed in Figure 5.16 and Figure 5.17, we study the PCA performance without deep features, relying only on the STANDARD scaler.

The PCA results over more than 4000 tests for EER, F1 score, accuracy, and inference time are: EER mean is 35.86% (standard deviation 17.05%), and with 95% confidence, it falls between 35.35% and 36.36%. F1 mean is 50.21% (standard deviation 33.67%), and with 95% confidence it falls between 49.21% and 51.22%. Accuracy mean is 65.96% (standard deviation is 19.06%) and with 95% confidence, it falls between 65.40% and 66.53%. Inference time mean is 18.4 milliseconds (standard deviation 1.9 milliseconds) and with 95% confidence, it falls between 18.3 and 18.4 milliseconds.

The VAE results over more than 4000 tests for EER, F1 score, accuracy, and inference time are: EER mean is 34.53% (standard deviation 15.92%) and with 95% confidence, it falls between 34.06% and 35.01%. F1 mean is 62.94% (standard deviation 27.45%) and

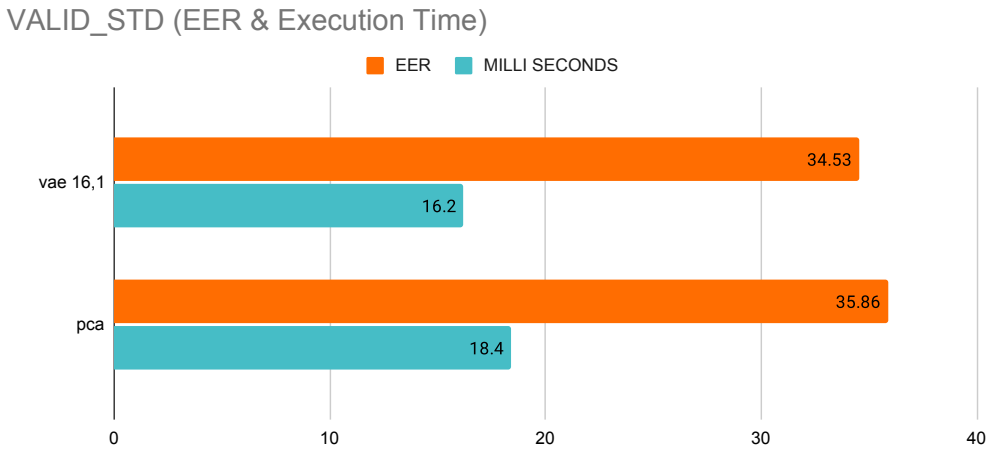

Figure 5.16: Testing EER and inference time (milliseconds) of PCA and VAE models under the newly proposed experiment setting.

with 95% confidence it falls between 62.13% and 63.76%. The accuracy mean is 67.63% (standard deviation 19.26%) and with 95% confidence, it falls between 67.06% and 68.20%. Inference time mean is 16.2 milliseconds (standard deviation 2.9 milliseconds) and with 95% confidence, it falls between 16.1 and 16.3 milliseconds.

We compute F1, accuracy, EER, and time delay to compare the PCA and the VAE models under the VALID normalization approach and STANDARD scaling method. It is apparent that VAE is outperforming PCA, and the F1 score of PCA is an order of magnitude lower. This confirms that VAE is the superior method for this use case.

In Figure 5.18, we present the best performing models according to our extensive evaluations under the two important experiment settings for continuous and implicit authentication. NAIVE MINMAX represents a setting comparable to what is being used by other academic works, and VALID STD represents a setting comparable to real-life deployments of a CIA system.

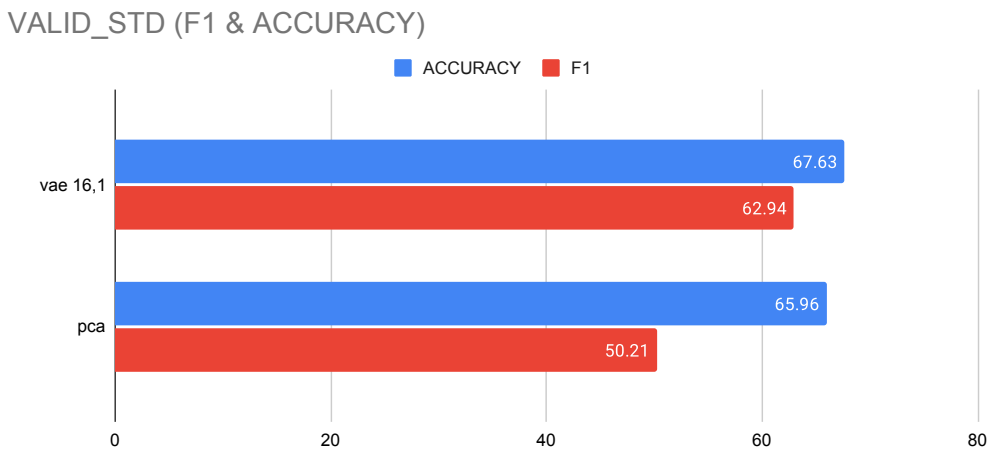

Figure 5.17: Testing accuracy and F1 of PCA and VAE models under the newly proposed experiment setting.

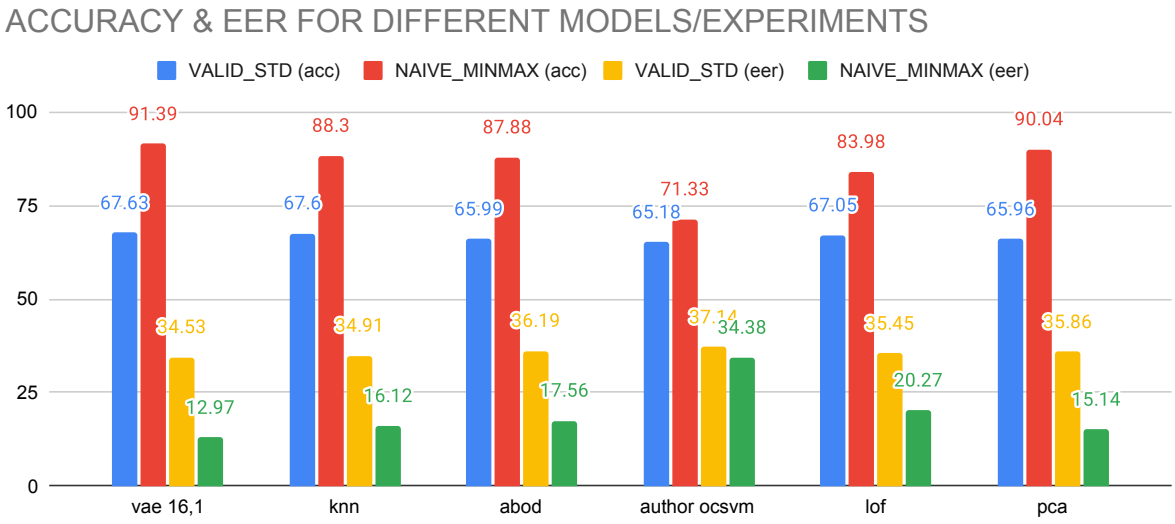

Figure 5.18: Testing accuracy and EER of the state-of-the-art and 5 best models in VALID-STD and NAIVE-MINMAX experiment settings.

Parameters

We found the best setting for VAE in NAIVE MINMAX 2D was that of the following configurations:

- Number of layers in encoder: 2
- Number of neurons in first layer: 1
- Number of neurons in second layer: 16
- Activation function to use for hidden layers: hyperbolic tangent function
- Activation function to use for output layer: Soft-max
- Optimizer = stochastic gradient descent
- Loss function: mean squared error
- L2 regulaizer = 0.5
- Dropout rate: .25
- Epochs = 100
- Batch size: 32
- Coefficient of beta VAE regime: 1.5
- Maximum capacity of a loss bottle neck: 1

The architecture is detailed in the codebase and in Appendix A.1.

The best setting for ABOD is

- Number of neighbors to use by default for k neighbors queries: 10
- Method: Only consider a number of neighbors of training points

The best setting for KNN is

- Number of neighbors used for k neighbors queries: 5
- use the distance to the kth neighbor as the outlier score
- Range of parameter space to use by default for radius neighbors queries: 1
- Leaf size passed to BallTree: 30
- Metric used for the distance computation: Minkowski with euclidean distance l2

The best setting for PCA is

- The eigenvalues are used in score computation
- Perform standardization first to convert data to zero mean and unit variance

The best setting for LOF is

- Number of neighbors used for k neighbors queries: 20
- Leaf size passed to BallTree or KDTree:30
- Metric used for the distance computation: Minkowski with euclidean distance l2

The best setting for the state of the art Author OCSVM, as reported in [13] is

- kernel type: rbf
- An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors (nu): 0.110
- Kernel coefficient (gamma): 59.636

5.4 Discussion

We expected our results to be more conservative than some of those reported by other works due to our smaller training dataset caused by our additional validation split to support our extensive hyper-parameter tuning, our more extensive cross-validation, and the preprocessing and normalization steps taken to avoid data contamination, in our efforts to make our evaluations comparable to real-life deployments of such a system. However, our results outperform many of the reported numbers in the domain, as covered Chapter 3. The work presented by Centeno *et al.* [20] reported 97.8% accuracy on their SCNN and OCSVM ensemble and 86.9% accuracy on their OCSVM. However, despite extensive efforts, including correspondences with Centeno, by Buech [13] for reproducing the closed-source model, they could not reach above 90% accuracy in their re-implementation of the ensemble and 86.3% in the OCSVM case. It is unclear why the SCNN implemented by Buech is unable to perform similarly to Centeno's, although it has an identical architecture and parameter set. In the interest of transparency, we public-ally release all relevant software, including our VAE architecture to prevent such issues in reproducing our evaluations.

As previously discussed in Chapter 2, an EER of 15% means that in 85% of the times, the authorized user would be authenticated in a non-intrusive manner, without the need for active co-operation. This also entails that in 85% of the events where CIA authenticates the user, it is the correct person who gets authenticated. This, of course, is assuming the result is transferable to real-world scenarios and the experiment settings are representative of the real-world circumstances.

Our hypothesis that the VALID setting, representing a real-world deployment scenario, in which the normalization scaler was trained only on training data and is then used as-is to scale imposter and owner data instances during testing, causes a significant drop in performance compared to results from experiments using the NAIVE setting, where the scaler is given access to all data including the testing subset. The NAIVE setting additionally introduces bias by applying user-wise MINMAX normalization on all data and outliers.

Magnetometer data introduces bias because it captures data from the surrounding

environment and in theory, does not represent the device movement patterns as much as the other sensors.

VAE and KNN are outperforming all other models consistently, with VAE having a clear edge in performance, consistency, as well as being up to ten times faster in inference time. The common experiment used in the domain is NAIVE MINMAX with the deep feature extraction 2D variant as well as no deep feature extractor variant. The VAE can reach 11.71% EER and 92.52% accuracy as well as 12.97% EER and 91.39% accuracy without any deep feature extraction. Hence, we conclude that the VAE is a good fit for the problem statement and offers a performance gain as well as is a fast, reliable, and consistent model that can do its feature extraction. Most importantly, it shines in the VALID experiments under both ROBUST and STANDARD scalings which clean the data from any fingerprints that the models can use to circumvent the learning process.

5.4.1 Empowering the Edge

The edge defined in this use case as the smartphone is growing more powerful by the day. Smartphones today have incredible resources, similar to those of computers a few years ago, and are perfectly able to run their analysis. This is very important for privacy concerns over data being sent to the cloud, resource efficiency with lower bandwidth utilization, cheaper business models, and more efficient analysis. Its also important that in the hand over to authority figures threat model (B), we saw how border services are instructed to disable internet access for a variety of reasons, and if the solution was running on the cloud it would not have been able to detect and respond to the threat.

We chose fast models that are simple and easy-to-access data types that are available on all smartphones to allow for the solution to be as generalizable as possible and work efficiently. We were able to run our model on TensorFlow Lite, an open-source deep learning framework for edge AI, and packaged the data collection, data streaming, model training, and exporting mechanism in our open source repository. We also included all the experiments, models, utility code, implementations on both IOS and Android (*cf.*, Chapter 6), and notes in our repository to lower the barrier of entry to this research domain. We hope our contributions can improve reproducibility and encourage openness,

collaboration, and the thriving of the research in the intersection of cybersecurity and data science.

Chapter 6

Solution Implementations

Our experiments are all available as open-source on GitHub, the HMOG dataset processing modules we used have also been made available. We provide a Python environment that was used in Jupyter Ipython to run our experiments as documented in the notebooks provided. The easily followed notebooks allow the reader to understand how to reproduce the results very easily and how they fit into the logical flow of the work. These notebooks are also used to produce visualizations for the various results found in the experiments. Data acquisition and machine learning applications have been developed for IOS and Android and open-sourced. Software required to operate the experiments on a supercomputer has also been made available to shorten the development cycle and lower the barriers to entry for other researchers to adopt our code base and contribute further to the continuous and implicit authentication domain that will make our smart devices infused hyper-connected future more secure, convenient and efficient.

6.1 Solution Code Base

6.1.1 Experiments code-base

The opensource project is found under a publicly accessible GitHub organization named $UW - CIA$, in reference to the University of Waterloo and Continuous and Implicitly Authentication [169]. Under the experiments repository, there are three sections, similar to and adopted from Buech's [15]. The data section under the *data* folder hosts the HDF5 format of HMOG dataset, as well as other raw, interim, processed, or external data formats.

Appropriate data transformations, processing, and loading modules are found under the source folder *src*. Necessary data visualizations, explorations, and modeling as well as all kinds of experiments and evaluations are accessible under the Jupyter notebooks folder *notebooks*. Jupyter notebooks were the best vehicle to showcase our work since they offer clarity and explainability, unlike any other format, by combining source code, documentation, and output. Necessary third-party code and libraries needed are documented in the *environment.yml* file. It has metadata denoting the versions of Python and other modules and can easily be used to reproduce the whole programming environment, and accelerate the reproducibility of our work, Anaconda Python distribution or Python pip processes this file. The enclosed documentation $README.md$ contains a detailed description of all necessary steps needed to utilize the repository, and I am going to be supportive of using the $Issues$ feature on GitHub that allows anyone to ask me questions, publicly, relating to the usage of the codebase or even improve upon the codebase.

To improve reproducibility and re-usability, all the models utilized existing common software libraries like TensorFlow, and ScikitLearn [176, 177]. The codebase was cleaned and comments, as well as read-me files, were added where relevant.

6.1.2 Data Collection, Data Cloud Streaming, and AI at Edge

As part of our effort and commitment to make our domain and project closer to real-world deployments, we built and open-sourced a data collection and streaming application

that works on IOS and Android devices; it utilizes a software framework put together by Patton *et al.* [178], in MIT. The data acquisition is very simple. The application reads the data from the sensors in real-time as they become available and sends the relevant measurements in real-time to the cloud in an encrypted channel and a secured fashion. In our implementation of the receiving cloud, we kept it simple and had the results received by a spreadsheet hosted on a file sharing service known as Google Drive, but a simple database or even a data lake or warehouse can be utilized as well. Once the data is in the cloud the code base for the experiments can be utilized to run the whole set of experiments, explorations, and tests on the collected data. The data collection module was tested and is functional.

If a model is chosen, we also built and open-sourced the means for deploying any of the models we presented, especially the VAE, to run on IOS and Android devices, as well as on embedded devices commonly used in the Internet of Things applications. Our implementations are built on top of a software framework built by David *et al.* [179], in Google. There were challenges faced during the implementation of the VAE model, specifically, due to it being relatively new. For instance, to get one of the VAE implementations to run on tensor flow lite, the experimental interface *OpsSet* needs to be called, and *targetspec* as well as *targetops* flags need to be set to allow for certain operation and built-ins in the model transformation during the run time of the exported model. New or unpopular models are usually relatively harder to implement compared to more common classical models, due to the scarcity of reference implementations, architectures, and learning resources. We ended up with two implementations to support two different VAE implementations at the edge, one based on the PYOD software library while the other is based on the more common Tensorflow software library. Both implementations were tested at the edge and both are functional.

6.2 Solution Computing Resources

During the experiments, we were fortunate to access the Niagara supercomputer, part of an initiative called Compute Canada to support academic intensive computing needs. As

part of this work, we had to develop a set of software to allow for better management of the data, experiments, and working jobs for the supercomputer system, managed by a software platform known as slum. All the relevant code is made available under our repo in the *home* folder.

Chapter 7

Conclusion

7.1 Recommendations

Besides our detailed suggestions across the chapters that would allow future works to contribute to the domain strategically in different categories of the work from general approach and modeling Chapter 4, to evaluations Chapter 5 and implementations Chapter 6, we also would like to give high-level directions worthy of further explorations.

For those who tackle this domain in the future, the most impactful investigations in our opinion are the following:

- Extensively hyperparameter tune all the anomaly detector models that were not covered, *e.g.*, HMM, Bayesian Nets, or fuzzy logic, to find the optimal configuration for each using the whole HMOG dataset mean score as your optimization factor for every possible experiment configuration
- Repeat the previous point but this time to find the optimal configuration for each using one specific user at a time from HMOG dataset mean score as your optimization factor, so the models are fit for each user
- Train and test a VQ-VAE (Vector Quantised-Variational AutoEncoder) [180]. Vector Quantisation-VAE learns a discrete latent variable by the encoder. VQ maps k

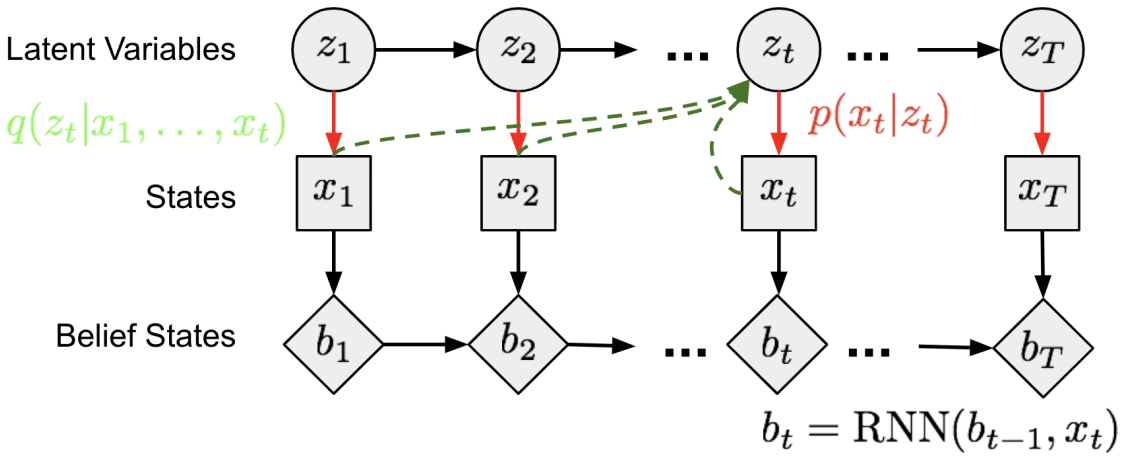

Figure 7.1: TD VAE state-space model as a Markov Chain. Source: [10]

dimensional vectors into a finite set of vectors in a process similar to that of KNN: the optimal centroid code vector that a sample should be mapped to is the one with minimum Euclidean distance. [10]

- Train and test a VQ-VAE-2 [181]. VQ-VAE-2 [181] is a two-level hierarchical VQ-VAE combined with a self-attention auto-regressive model.
- Train and test a TD-VAE [182]. Temporal difference VAE, as in Section 7.1 was built for sequential data and relies on state-space models, belief states which encode all past states, and jumpy prediction or predicting states several steps further into the future.
- Anomaly detection models based on meta learning [183, 184, 185, 186, 187], self supervised [188, 189] and reinforcement learning [190, 191, 192] are emerging and might be worth exploring. However they are very resource intensive.
- Evaluate multi-modal to improve reliability [193]. Though the base data should be inertial, areas like touch, gesture, wearables [194], etc. can be used as the privacy posture, and the user permissions appetite allow in certain cases the system to access more revealing data sources.

- Differential privacy might be used if the user allows for cloud-based training
- Explore federated and split learning for model improvement and collaborative learning
- Explore third-party integrations that can allow for contextual awareness that helps set a threshold for suspicion in different use cases, locations, and day times
- Collect a better data set that is bigger and represents real-world scenarios and attacks
- As discussed in Chapter 5 a study of Normalizer, Power and Quantile transformers scalers can add to the design of the standard experiment settings used in the body of research
- As discussed in Chapter 3 detecting the user activity first (walking, or sitting) then using a specifically optimized inference model, and collaborating with relevant third party systems can improve reliability in real-world scenarios
- Use an absolute coordinate system by removing the gravity vector from the accelerometer to calculate a spatial orientation and coordinate system for the device which can be a good feature to add

One of the biggest issues in the continuous and implicit authentication domain is the reproducibility of the work. Information on the bigger picture is usually well presented but finer details, usually found in code, and necessary for reproducing the results and approach are almost impossible to find publicly available and highly unlikely to be acquired through correspondence. Such details like standardization, normalization, data cleaning, preprocessing, resampling, and splitting as well as model hyperparameters and experiment cross-validation details are usually guessed by later works and left for future researchers speculation. It is the case because a research paper format does not allow for too many, commonly forgettable, details to be shared and prioritization of main methodologies and bigger picture approaches is commonplace. We argue that open-sourcing the codebase should become the standard in such works. It is, unfortunately, the case that none of the related works, but Buech [15] have made their source code public, and the ones we

tried reaching and even corresponded with like Shen *et al.* [19] refused to share data or code. As covered in Chapter 2 this makes the studies harder to compare and learn from as the reader is left unsure which of the many degrees of freedom, *e.g.*, dataset, metrics, experiment settings, cross-validation, or data preprocessing contributed to the performance and results reported. Our work hopes to be a step closer to a standard evaluation and deployment environment and the code and our contributions are all open-sourced [169] and thoroughly described in Chapter 4.

The domain does not only need openness and standards for sharing work, but also a better publicly available dataset than HMOG. HMOG is by far the best dataset right now for this domain for comparability, as extensively described in Chapter 3, but it does come with a few critical flaws. HMOG limits our work's transfer-ability to real-world deployments due to it being recorded in a controlled setting with no attack scenarios. This makes the sessions deviate considerably from real-world scenarios and does not aid in making the great wealth of work in this area transferable to real-world applications. It would be an amazing feat if a dataset is collected with attack scenarios based on the real threat models and use cases described in Chapter 4. Such a contribution could transform this domain and enable more commercialization of the plethora of great innovations in it, as highlighted in Chapter 3.

It is worth exploring to integrate the microenvironment sensing [195] to analyze what activity the user is doing or where the phone is (*e.g.*, right thigh pocket, left chest pocket, purse, strapped to the arm, in hand). This would allow the anomaly detector to assess what activity is being performed. In its current state, the anomaly detector will have labeled ground truth data to support every activity there is and will utilize contextual awareness through third-party applications to refine its assessments but adding this kind of situational awareness via environment and activity sensing can add to the overall solution.

Because the most straightforward way of improving the performance of deep neural networks is by increasing their size [168], we encourage other researchers to experiment with deeper architectures.

7.2 Discussion

We have introduced the domain and briefly listed our contributions in Chapter 1. We have assessed the feasibility of only using inertial sensor data, which were covered in Chapter 2, from mobile devices to classify the current user as either the owner or an unauthorized imposter, continuously and implicitly. A framework and methodology were proposed, in Chapter 4 that not only modeled threats and scenarios but also integrated current third party applications, technologies, and investments in information technology concaveness in an enterprise, campus, military, or home setting towards a more comprehensive and collaborative continuous authentication system. Our work also delivered a cross-study, in Chapter 4 and in Chapter 5, comparing various classic and new machine learning approaches under various, and reproducible controlled experiments based on an opensource framework [15] with a public dataset [24], and highlighted the use case for generative models and their inherent strengths in such systems and threat models, use cases and attack scenarios. In our extensive literature review, in Chapter 3 , we identified that this domain did not have a similar machine learning models comparative study other than one in 2017 [19] and another in 2014 [196], which was published by our lab under the direction of Prof. Urs Hengartner. The domain had one major flaw: severely complicated reproducibility, which we addressed by choosing to work with public data and releasing our experiments, models, and implementations, as well as propose a template and a standard for other works to follow in reporting their metrics, and experiments. Our pivotal work on evaluating generative models, allowed us to note that GANs and fm-GANS, might not be great for unsupervised and semi-supervised learning since too many labels are needed for the excessive hyper-parameter tuning needed, while on the other hand, other generative models, like VAEs in particular, show promising potential in semi-supervised outlier detection with the data and should be studied in a greater depth as shown in Chapter 5 and suggested by other researchers [121]. The VAE outperforms all other models in our various experiments, some of which are proposed by us to mimic realistic scenarios more closely, in this continuous and implicit authentication system and paves the way for generative models into this domain, for the first time. Our work replaces the need for heavy deep feature extraction, as well as manual feature engineering, by relying on shallow variational autoencoder models, that

have inherent probabilistic and dimensionality reduction capabilities. Our work also open sources multiple Android and IOS implementations, in Chapter 6, needed to utilize the novel contributions in the real-world, as well as stream sensor data to the cloud to allow for cloud-based training as well as on-device training, and advocates for openness in this domain to further its transfer-ability into the real-world. The reasoning for every decision from threat modeling to model selection to cross-validation and data normalization and cleaning was given with thorough explanations and references to the existing body of work.

> "By securing, I mean: building systems to remain dependable in the face of malice, error, or mischance" - Prof. Ross Anderson (Security Engineering Pioneer)

References

[1] D. Crouse, H. Han, D. Chandra, B. Barbello, and A. K. Jain, "Continuous authentication of mobile user: Fusion of face image and inertial measurement unit data," in *2015 International Conference on Biometrics (ICB)*. IEEE, 2015, pp. 135–142.

[2] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *Foundations and Trends® in Signal Processing*, vol. 11, no. 1-2, p. 1–153, 2017. [Online]. Available: http://dx.doi.org/10.1561/2000000094

[3] E. Hering, G. Schönfelder *et al.*, *Sensoren in Wissenschaft und Technik*. Springer, 2012.

[4] M. Masoud, Y. Jaradat, A. Manasrah, and I. Jannoud, "Sensors of smart devices in the internet of everything (ioe) era: Big opportunities and massive doubts," *Journal of Sensors*, vol. 2019.

[5] K. Kalantar-Zadeh, *Sensors: an introductory course*. Springer Science & Business Media, 2013.

[6] S. Liehr, "Optical Measurement of Currents in Power Converters," Apr. 2006.

[7] M. Azzurri, "Face recognition system and calculating FRR, FAR and EER for Biometric system evaluation + code," Jun. 2020. [Online]. Available: https://medium.com/@mustafaazzurri/face-recognition-system-and-calculating-frr-far-and-eer-for-biometric-system-evaluation-code-2ac2bd4fd2e5

[8] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, 2021.

[9] F.-F. Li, J. Johnson, and S. Yeung, "Lecture 11: Generative Models," p. 136, 2019.

[10] "From Autoencoder to Beta-VAE," Aug. 2018. [Online]. Available: https://lilianweng.github.io/2018/08/12/from-autoencoder-to-beta-vae.html

[11] A. Dertat, "Applied Deep Learning - Part 3: Autoencoders," Oct. 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

[12] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[13] H. Büch, "Continuous Authentication using Inertial-Sensors of Smartphones and Deep Learning," mastersthesis, Hochschule der Medien, Stuttgart, 6 2019. [Online]. Available: https://hdms.bsz-bw.de/frontdoor/index/index/docId/6506

[14] Q. Yang, G. Peng, D. T. Nguyen, X. Qi, G. Zhou, Z. Sitová, P. Gasti, and K. S. Balagani, "A multimodal data set for evaluating continuous authentication performance in smartphones," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. Memphis Tennessee: ACM, Nov. 2014, pp. 358–359. [Online]. Available: https://dl.acm.org/doi/10.1145/2668332.2668366

[15] H. Büch. (2019, 6) ContinAuth. [Online]. Available: https://github.com/dynobo/ContinAuth

[16] M. P. Centeno, A. van Moorsel, and S. Castruccio, "Smartphone continuous authentication using deep learning autoencoders," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2017, pp. 147–1478.

[17] Z. Wang, W. Yan, and T. Oates, "Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline," *arXiv:1611.06455 [cs, stat]*, Dec. 2016, arXiv: 1611.06455. [Online]. Available: http://arxiv.org/abs/1611.06455

[18] M. Ehatisham-ul Haq, M. A. Azam, J. Loo, K. Shuang, S. Islam, U. Naeem, and Y. Amin, "Authentication of smartphone users based on activity recognition and mobile sensing," *Sensors*, vol. 17, no. 9, p. 2043, 2017.

[19] C. Shen, Y. Li, Y. Chen, X. Guan, and R. A. Maxion, "Performance analysis of multi-motion sensor behavior for active smartphone authentication," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 48–62, 2017.

[20] M. P. Centeno, Y. Guan, and A. van Moorsel, "Mobile based continuous authentication using deep features," in *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning*, 2018, pp. 19–24.

[21] "Number of mobile devices worldwide 2020-2024." [Online]. Available: https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide/

[22] P. Ruggiero and J. Foote, "Cyber Threats to Mobile Phones," p. 6.

[23] S. Z. S. Idrus, E. Cherrier, C. Rosenberger, and J.-J. Schwartzmann, "A review on authentication methods," *Australian Journal of Basic and Applied Sciences*, vol. 7, no. 5, pp. 95–107, 2013.

[24] Z. Sitová, J. Šeděnka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. S. Balagani, "Hmog: New behavioral biometric features for continuous authentication of smartphone users," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 877–892, 2015.

[25] C. B. S. A. Government of Canada, "Examining digital devices at the canadian border," Jan 2021. [Online]. Available: https://www.cbsa-asfc.gc.ca/travel-voyage/edd-ean-eng.html#01

[26] 2018, vol. U.S. CUSTOMS AND BORDER PROTECTION.

[27] F. Stajano, "Pico: No more passwords!" in *Security Protocols XIX*, B. Christianson, B. Crispo, J. Malcolm, and F. Stajano, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 49–81.

[28] S. Depatla, A. Muralidharan, and Y. Mostofi, "Occupancy estimation using only wifi power measurements," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 7, pp. 1381–1393, 2015.

[29] A. Khalili, A.-H. Soliman, M. Asaduzzaman, and A. Griffiths, "Wi-fi sensing: Applications and challenges," *The Journal of Engineering*, vol. 2020, no. 3, pp. 87–97, 2020.

[30] Q. Xu, R. Zheng, and E. Tahoun, "Detecting location fraud in indoor mobile crowdsensing," in *Proceedings of the First ACM Workshop on Mobile Crowdsensing Systems and Applications*, 2017, pp. 44–49.

[31] Y. Gu, Y. Zhang, J. Li, Y. Ji, X. An, and F. Ren, "Sleepy: Wireless Channel Data Driven Sleep Monitoring via Commodity WiFi Devices," *IEEE Transactions on Big Data*, 2020.

[32] H. Yiğitler, O. Kaltiokallio, R. Hostettler, A. S. Abrar, R. Jäntti, N. Patwari, and S. Särkkä, "RSS Models for Respiration Rate Monitoring," *IEEE Transactions on Mobile Computing*, 2020.

[33] S. Vakalis, L. Gong, and J. Nanzer, "Imaging With WiFi," *IEEE Access*, 2019.

[34] X. Wu, Z. Chu, P. Yang, C. Xiang, X. Zheng, and W. Huang, "TW-See: Human Activity Recognition Through the Wall With Commodity Wi-Fi Devices," *IEEE Transactions on Vehicular Technology*, 2019.

[35] H. Lee, C. Ahn, N. Choi, T. Kim, and H. Lee, "The Effects of Housing Environments on the Performance of Activity-Recognition Systems Using Wi-Fi Channel State Information: An Exploratory Study," *Sensors*, 2019.

[36] M. A. A. Haseeb and R. Parasuraman, "Wisture: Touch-Less Hand Gesture Classification in Unmodified Smartphones Using Wi-Fi Signals," *IEEE Sensors Journal*, 2019.

[37] Z. Fu, J. Xu, Z. Zhu, A. X. Liu, and X. Sun, "Writing in the Air with WiFi Signals for Virtual Reality Devices," *IEEE Transactions on Mobile Computing*, 2019.

[38] A. Khalili, A. Soliman, and Asaduzzaman, "A Deep Learning Approach for Wi-Fi Based People Localization," 2018.

[39] I. Sobrón, J. Ser, I. Eizmendi, and M. Vélez, "A Deep Learning Approach to Device-Free People Counting from WiFi Signals," in *IDC*, 2018.

[40] F. M. Adib, Z. Kabelac, and D. Katabi, "Multi-Person Localization via RF Body Reflections," in *NSDI*, 2015.

[41] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, "Whole-home gesture recognition using wireless signals," *MobiCom*, 2013.

[42] S. Gupta, D. Morris, S. Patel, and D. S. Tan, "SoundWave: using the doppler effect to sense gestures," *CHI*, 2012.

[43] L. Liu, M. Popescu, M. Skubic, M. Rantz, T. Yardibi, and P. Cuddihy, "Automatic fall detection based on Doppler radar motion signature," *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, 2011.

[44] M. Souppaya and K. Scarfone, "Guidelines for managing the security of mobile devices in the enterprise," *NIST special publication*, vol. 800, p. 124, 2020.

[45] E. Atwater and I. Goldberg, "Shatter secrets: Using secret sharing to cross borders with encrypted devices (transcript of discussion)," in *Security Protocols XXVI*, V. Matyáš, P. Švenda, F. Stajano, B. Christianson, and J. Anderson, Eds. Cham: Springer International Publishing, 2018, pp. 295–303.

[46] E. Tahoun and R. Khedri, "Exploring strategies for digital security," 2017.

[47] "Fido2: Moving the world beyond passwords using webauthn ctap," Jun 2020. [Online]. Available: https://fidoalliance.org/fido2/

[48] "Piv usage guides." [Online]. Available: https://piv.idmanagement.gov/#:~:text=findtheStandards?-,WhatisPIV?,attheappropriatesecuritylevel.

[49] "Motion Sensors Explainer." [Online]. Available: https://www.w3.org/TR/motion-sensors/#absolute-orientation-sensor

[50] S. Kamburugamuve, L. Christiansen, and G. Fox, "A Framework for Real Time Processing of Sensor Data in the Cloud," *Journal of Sensors*, vol. 2015, p. e468047, Apr. 2015, publisher: Hindawi. [Online]. Available: https://www.hindawi.com/journals/js/2015/468047/

[51] Y. Xu, M. Lin, H. Lu, G. Cardone, N. Lane, Z. Chen, A. Campbell, and T. Choudhury, "Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns," in *Proceedings of the 2013 International Symposium on Wearable Computers*, ser. ISWC '13. New York, NY, USA: Association for Computing Machinery, Sep. 2013, pp. 69–76. [Online]. Available: https://doi.org/10.1145/2493988.2494333

[52] C. Xu, S. Li, G. Liu, Y. Zhang, E. Miluzzo, Y.-F. Chen, J. Li, and B. Firner, "Crowd++: unsupervised speaker count with smartphones," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, ser. UbiComp '13. New York, NY, USA: Association for Computing Machinery, Sep. 2013, pp. 43–52. [Online]. Available: https://doi.org/10.1145/2493432.2493435

[53] J.-K. Min, A. Doryab, J. Wiese, S. Amini, J. Zimmerman, and J. I. Hong, "Toss 'n' turn: smartphone as sleep and sleep quality detector," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: Association for Computing Machinery, Apr. 2014, pp. 477–486. [Online]. Available: https://doi.org/10.1145/2556288.2557220

[54] U. A. Abdulla, K. Taylor, M. Barlow, and K. Z. Naqshbandi, "Measuring Walking and Running Cadence Using Magnetometers," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Jul. 2013, pp. 1458–1462, iSSN: 2324-9013.

[55] M. Masoud, Y. Jaradat, A. Manasrah, and I. Jannoud, "Sensors of Smart Devices in the Internet of Everything (IoE) Era: Big Opportunities and Massive Doubts,"

Journal of Sensors, vol. 2019, p. e6514520, May 2019, publisher: Hindawi. [Online]. Available: https://www.hindawi.com/journals/js/2019/6514520/

[56] J. L. Kröger, P. Raschke, and T. R. Bhuiyan, "Privacy implications of accelerometer data: A review of possible inferences," in *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, ser. ICCSP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 81–87. [Online]. Available: https://doi.org/10.1145/3309074.3309076

[57] S. Sugrim, C. Liu, M. McLean, and J. Lindqvist, "Robust performance metrics for authentication systems," in *Network and Distributed Systems Security (NDSS) Symposium 2019*, 2019.

[58] J. Jang and H. Kim, "Performance Measures," in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain, Eds. Boston, MA: Springer US, 2009, pp. 1062–1068. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5_111

[59] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[60] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[61] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, "Complex human activity recognition using smartphone and wrist-worn motion sensors," *Sensors*, vol. 16, no. 4, p. 426, 2016.

[62] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum, "Livelab: measuring wireless networks and smartphone users in the field," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 3, pp. 15–20, 2011.

[63] M. Abuhamad, A. Abusnaina, D. Nyang, and D. Mohaisen, "Sensor-based continuous authentication of smartphones' users using behavioral biometrics: A contemporary survey," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 65–84, 2020.

[64] L. Gonzalez-Manzano, J. M. D. Fuentes, and A. Ribagorda, "Leveraging user-related internet of things for continuous authentication: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–38, 2019.

[65] A. Roy, T. Halevi, and N. Memon, "An hmm-based multi-sensor approach for continuous mobile authentication," in *MILCOM 2015-2015 IEEE Military Communications Conference*. IEEE, 2015, pp. 1311–1316.

[66] N. Neverova, "Deep learning for human motion analysis," Ph.D. dissertation, Université de Lyon, 2016.

[67] D. Deb, A. Ross, A. K. Jain, K. Prakah-Asante, and K. V. Prasad, "Actions speak louder than (pass) words: Passive authentication of smartphone* users via deep temporal features," in *2019 International Conference on Biometrics (ICB)*. IEEE, 2019, pp. 1–8.

[68] "CrowdSignals.io: A Massive New Mobile Data Collection Campaign." [Online]. Available: http://crowdsignals.io/

[69] Y. Li, H. Hu, G. Zhou, and S. Deng, "Sensor-based continuous authentication using cost-effective kernel ridge regression," *IEEE Access*, vol. 6, pp. 32 554–32 565, 2018.

[70] J. Zhu, P. Wu, X. Wang, and J. Zhang, "Sensec: Mobile security through passive sensing," in *2013 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2013, pp. 1128–1133.

[71] U. Mahbub, S. Sarkar, V. M. Patel, and R. Chellappa, "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)*, Sept 2016, pp. 1–8.

[72] A. K. Belman, L. Wang, S. Iyengar, P. Sniatala, R. Wright, R. Dora, J. Baldwin, Z. Jin, and V. V. Phoha, "Insights from bb-mas–a large dataset for typing, gait and swipes of the same person on desktop, tablet and phone," *arXiv preprint arXiv:1912.02736*, 2019.

[73] N. Zheng, K. Bai, H. Huang, and H. Wang, "You are how you touch: User verification on smartphones via tapping behaviors," in *2014 IEEE 22nd International Conference on Network Protocols*. IEEE, 2014, pp. 221–232.

[74] M. Trojahn and F. Ortmeier, "Toward mobile authentication with keystroke dynamics on mobile phones and tablets," in *2013 27th International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2013, pp. 697–702.

[75] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, classifiaction," 1992.

[76] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2, pp. 131–163, 1997.

[77] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid."

[78] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, and G. Taylor, "Learning human identity from motion patterns," *IEEE Access*, vol. 4, pp. 1810–1820, 2016.

[79] L. Yang, Y. Guo, X. Ding, J. Han, Y. Liu, C. Wang, and C. Hu, "Unlocking smart phone through handwaving biometrics," *IEEE Transactions on Mobile Computing*, vol. 14, no. 5, pp. 1044–1055, 2014.

[80] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[81] B. Shrestha, N. Saxena, and J. Harrison, "Wave-to-access: Protecting sensitive mobile device services via a hand waving gesture," in *International Conference on Cryptology and Network Security*. Springer, 2013, pp. 199–217.

[82] B. Draffin, J. Zhu, and J. Zhang, "Keysens: Passive user authentication through micro-behavior modeling of soft keyboard interaction," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2013, pp. 184–201.

[83] Y.-S. Hwang and S.-Y. Bang, "An efficient method to construct a radial basis function neural network classifier," *Neural networks*, vol. 10, no. 8, pp. 1495–1503, 1997.

[84] M. O. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive user-authentication on mobile phones using biometric gait recognition," in *2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2010, pp. 306–311.

[85] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[86] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S.-M. Makela, and H. Ailisto, "Identifying users of portable devices from gait pattern with accelerometers," in *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 2. IEEE, 2005, pp. ii–973.

[87] G. Kambourakis, D. Damopoulos, D. Papamartzivanos, and E. Pavlidakis, "Introducing touchstroke: keystroke-based authentication system for smartphones," *Security and Communication Networks*, vol. 9, no. 6, pp. 542–554, 2016.

[88] M. Pal, "Random forest classifier for remote sensing classification," *International journal of remote sensing*, vol. 26, no. 1, pp. 217–222, 2005.

[89] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2003, pp. 986–996.

[90] T. Feng, X. Zhao, B. Carbunar, and W. Shi, "Continuous mobile authentication using virtual key typing biometrics," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2013, pp. 1547–1552.

[91] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE transactions on information forensics and security*, vol. 8, no. 1, pp. 136–148, 2012.

[92] N. Sae-Bae and N. Memon, "Online signature verification on mobile devices," *IEEE transactions on information forensics and security*, vol. 9, no. 6, pp. 933–947, 2014.

[93] M. Kunz, K. Kasper, H. Reininger, M. Möbius, and J. Ohms, "Continuous speaker verification in realtime," *BIOSIG 2011–Proceedings of the Biometrics Special Interest Group*, 2011.

[94] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[95] M. Shahzad, A. X. Liu, and A. Samuel, "Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it," in *19th annual international conference on Mobile computing & networking*, 2013.

[96] N. L. Clarke and A. Mekala, "The application of signature recognition to transparent handwriting verification for mobile devices," *Information management & computer security*, 2007.

[97] A. Das, O. K. Manyam, M. Tapaswi, and V. Taranalli, "Multilingual spoken-password based user authentication in emerging economies using cellular phone networks," in *2008 IEEE Spoken Language Technology Workshop*. IEEE, 2008, pp. 5–8.

[98] B. D. Martin, V. Addona, J. Wolfson, G. Adomavicius, and Y. Fan, "Methods for Real-Time Prediction of the Mode of Travel Using Smartphone-Based GPS and Accelerometer Data," *Sensors*, vol. 17, no. 9, p. 2058, Sep. 2017, number: 9 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/17/9/2058

[99] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine," in *Ambient Assisted Living and Home Care*, ser. Lecture Notes in Computer Science, J. Bravo, R. Hervás, and M. Rodríguez, Eds. Berlin, Heidelberg: Springer, 2012, pp. 216–223.

[100] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, Oct. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417416302056

[101] "UCI Machine Learning Repository: Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set." [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions

[102] M. Ehatisham-ul Haq, M. A. Azam, U. Naeem, S. u. Rhman, and A. Khalid, "Identifying Smartphone Users based on their Activity Patterns via Mobile Sensing," *Procedia Computer Science*, vol. 113, pp. 202–209, Jan. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917317593

[103] W.-H. Lee and R. B. Lee, "Implicit Smartphone User Authentication with Sensors and Contextual Machine Learning," *arXiv:1708.09754 [cs]*, Aug. 2017, arXiv: 1708.09754. [Online]. Available: http://arxiv.org/abs/1708.09754

[104] H. C. Volaka, G. Alptekin, O. E. Basar, M. Isbilen, and O. D. Incel, "Towards Continuous Authentication on Mobile Phones using Deep Learning Models," *Procedia Computer Science*, vol. 155, pp. 177–184, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091930941X

[105] Y. Li, H. Hu, and G. Zhou, "Using data augmentation in continuous authentication on smartphones," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 628–640, 2018.

[106] S. Amini, V. Noroozi, S. Bahaadini, S. Y. Philip, and C. Kanich, "Deepfp: A deep learning framework for user fingerprinting via mobile motion sensors," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 84–91.

[107] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019, arXiv: 1906.02691. [Online]. Available: http://arxiv.org/abs/1906.02691

[108] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *International Conference on Machine Learning*. PMLR, Jun. 2014, pp. 1278–1286, iSSN: 1938-7228. [Online]. Available: http://proceedings.mlr.press/v32/rezende14.html

[109] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114 [cs, stat]*, May 2014, arXiv: 1312.6114. [Online]. Available: http://arxiv.org/abs/1312.6114

[110] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *arXiv:1406.2661 [cs, stat]*, Jun. 2014, arXiv: 1406.2661. [Online]. Available: http://arxiv.org/abs/1406.2661

[111] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial Autoencoders," Nov. 2015. [Online]. Available: https://arxiv.org/abs/1511.05644v2

[112] ——, "Adversarial Autoencoders," *arXiv:1511.05644 [cs]*, May 2016, arXiv: 1511.05644. [Online]. Available: http://arxiv.org/abs/1511.05644

[113] E. Principi, F. Vesperini, S. Squartini, and F. Piazza, "Acoustic novelty detection with adversarial autoencoders," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 3324–3330, iSSN: 2161-4407.

[114] X. Chen and E. Konukoglu, "Unsupervised Detection of Lesions in Brain MRI using constrained adversarial auto-encoders," *arXiv:1806.04972 [cs]*, Jun. 2018, arXiv: 1806.04972. [Online]. Available: http://arxiv.org/abs/1806.04972

[115] "VAE-GAN Based Zero-shot Outlier Detection | Proceedings of the 2020 4th International Symposium on Computer Science and Intelligent Control." [Online]. Available: https://dl.acm.org/doi/10.1145/3440084.3441180

[116] T. Ueda, Y. Tohsato, and I. Nishikawa, "Temporal Anomaly Detection by Deep Generative Models with Applications to Biological Data," in *Artificial Neural Networks and Machine Learning – ICANN 2020*, ser. Lecture Notes in Computer Science,

I. Farkaš, P. Masulli, and S. Wermter, Eds. Cham: Springer International Publishing, 2020, pp. 553–565.

[117] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery," Mar. 2017. [Online]. Available: https://arxiv.org/abs/1703.05921v1

[118] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient GAN-Based Anomaly Detection," *arXiv:1802.06222 [cs, stat]*, May 2019, arXiv: 1802.06222. [Online]. Available: http://arxiv.org/abs/1802.06222

[119] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training," *arXiv:1805.06725 [cs]*, Nov. 2018, arXiv: 1805.06725. [Online]. Available: http://arxiv.org/abs/1805.06725

[120] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi, "A Survey on GANs for Anomaly Detection," *arXiv:1906.11632 [cs, stat]*, Jun. 2019, arXiv: 1906.11632. [Online]. Available: http://arxiv.org/abs/1906.11632

[121] V. Škvára, T. Pevný, and V. Šmídl, "Are generative deep models for novelty detection truly better?" p. 7.

[122] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A Unifying Review of Deep and Shallow Anomaly Detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, May 2021, conference Name: Proceedings of the IEEE.

[123] "Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications | Proceedings of the 2018 World Wide Web Conference." [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3178876.3185996

[124] R.-Q. Chen, G.-H. Shi, W.-L. Zhao, and C.-H. Liang, "A Joint Model for IT Operation Series Prediction and Anomaly Detection," *Neurocomputing*, vol. 448, pp. 130–139, Aug. 2021, arXiv: 1910.03818. [Online]. Available: http://arxiv.org/abs/1910.03818

[125] P. Luo, B. Wang, T. Li, and J. Tian, "ADS-B anomaly data detection model based on VAE-SVDD," *Computers & Security*, vol. 104, p. 102213, May 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821000377

[126] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, Apr. 2018, pp. 187–196. [Online]. Available: https://doi.org/10.1145/3178876.3185996

[127] X. Chen, S. You, K. C. Tezcan, and E. Konukoglu, "Unsupervised lesion detection via image restoration with a normative prior," *Medical Image Analysis*, vol. 64, p. 101713, Aug. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1361841520300773

[128] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," 2015.

[129] D. Park, Y. Hoshi, and C. C. Kemp, "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, Jul. 2018, conference Name: IEEE Robotics and Automation Letters.

[130] O. Cerri, T. Q. Nguyen, M. Pierini, M. Spiropulu, and J.-R. Vlimant, "Variational autoencoders for new physics mining at the Large Hadron Collider," *Journal of High Energy Physics*, vol. 2019, no. 5, p. 36, May 2019. [Online]. Available: http://link.springer.com/10.1007/JHEP05(2019)036

[131] D. Deb, A. Ross, A. K. Jain, K. Prakah-Asante, and K. V. Prasad, "Actions Speak Louder Than (Pass)words: Passive Authentication of Smartphone Users via Deep Temporal Features," *arXiv:1901.05107 [cs, eess]*, Jan. 2019, arXiv: 1901.05107. [Online]. Available: http://arxiv.org/abs/1901.05107

[132] J. Chen, U. Hengartner, H. Khan, and M. Mannan, "Chaperone: Real-time locking and loss prevention for smartphones," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 325–342. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/chen-jiayi

[133] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A Novel Anomaly Detection Scheme Based on Principal Component Classifier," p. 8.

[134] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001. [Online]. Available: https://direct.mit.edu/neco/article/13/7/1443-1471/6529

[135] M. Hubert and M. Debruyne, "Minimum covariance determinant," *Wiley interdisciplinary reviews: Computational statistics*, vol. 2, no. 1, pp. 36–43, 2010.

[136] M. Goldstein and A. Dengel, *Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm.*

[137] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets," p. 12.

[138] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," p. 12.

[139] Z. He, X. Xu, and S. Deng, "Discovering Cluster Based Local Outliers," *Pattern Recognition Letters*, vol. 2003, pp. 9–10, 2003.

[140] H.-P. Kriegel, M. S hubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*. Las Vegas, Nevada, USA: ACM Press, 2008, p. 444. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1401890.1401946

[141] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08. USA: IEEE Computer Society, Dec. 2008, pp. 413–422. [Online]. Available: https://doi.org/10.1109/ICDM.2008.17

[142] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*. Chicago, Illinois, USA: ACM Press, 2005, p. 157. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1081870.1081891

[143] "Lecture 2: k-nearest neighbors / Curse of Dimensionality." [Online]. Available: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html

[144] T. Wang, M. Qiao, Z. Lin, C. Li, H. Snoussi, Z. Liu, and C. Choi, "Generative Neural Networks for Anomaly Detection in Crowded Scenes," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1390–1399, May 2019, conference Name: IEEE Transactions on Information Forensics and Security.

[145] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[146] A. van den Oord and N. Kalchbrenner, "Pixel rnn," 2016.

[147] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *arXiv preprint arXiv:1701.05517*, 2017.

[148] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, 2001.

[149] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[150] N. R. Goodman, "Statistical analysis based on a certain multivariate complex gaussian distribution (an introduction)," *The Annals of Mathematical Statistics*, vol. 34, no. 1, pp. 152–177, 1963.

[151] D. Park, Y. Hoshi, and C. C. Kemp, "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder," *arXiv:1711.00614 [cs]*, Nov. 2017, arXiv: 1711.00614. [Online]. Available: http://arxiv.org/abs/1711.00614

[152] G. J. Krishna and V. Ravi, "Keystroke based User Authentication using Modified Differential Evolution," in *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, Oct. 2019, pp. 739–744, iSSN: 2159-3450.

[153] Y. Guo, W. Liao, Q. Wang, L. Yu, T. Ji, and P. Li, "Multidimensional Time Series Anomaly Detection: A GRU-based Gaussian Mixture Variational Autoencoder Approach," p. 16.

[154] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[155] H. Xu, Y. Feng, J. Chen, Z. Wang, H. Qiao, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, and et al., "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, 2018. [Online]. Available: http://dx.doi.org/10.1145/3178876.3185996

[156] M. Soelch, J. Bayer, M. Ludersdorfer, and P. van der Smagt, "Variational inference for on-line anomaly detection in high-dimensional time series," 2016.

[157] S. Clachar, "Novelty detection and cluster analysis in time series data using variational autoencoder feature maps," 2016.

[158] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2828–2837.

[159] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-VAE," *arXiv:1804.03599 [cs, stat]*, Apr. 2018, arXiv: 1804.03599. [Online]. Available: http://arxiv.org/abs/1804.03599

[160] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner, "Early visual concept learning with unsupervised deep learning," 2016.

[161] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," 2016.

[162] G. Bernieri, M. Conti, and F. Turrin, "KingFisher: an Industrial Security Framework based on Variational Autoencoders," *New York*, p. 6, 2019.

[163] R. Yao, C. Liu, L. Zhang, and P. Peng, "Unsupervised Anomaly Detection Using Variational Auto-Encoder based Feature Extraction," in *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, Jun. 2019, pp. 1–7.

[164] A. A. Pol, V. Berger, C. Germain, G. Cerminara, and M. Pierini, "Anomaly Detection with Conditional Variational Autoencoders," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. Boca Raton, FL, USA: IEEE, Dec. 2019, pp. 1651–1657. [Online]. Available: https://ieeexplore.ieee.org/document/8999265/

[165] "H-MOG Data Set." [Online]. Available: http://www.cs.wm.edu/~qyang/hmog.html

[166] "Dask — Dask documentation." [Online]. Available: https://docs.dask.org/en/latest/

[167] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[168] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv:1312.6199 [cs]*, Feb. 2014, arXiv: 1312.6199. [Online]. Available: http://arxiv.org/abs/1312.6199

[169] "Continuous and Implicit Authentication @ UWATERLOO." [Online]. Available: https://github.com/UW-CIA

[170] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[171] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," 2019.

[172] "sklearn.preprocessing.PowerTransformer — scikit-learn 0.24.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer.html

[173] "sklearn.preprocessing.QuantileTransformer — scikit-learn 0.24.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html

[174] "sklearn.preprocessing.Normalizer — scikit-learn 0.24.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html

[175] "Compare the effect of different scalers on data with outliers — scikit-learn 0.24.2 documentation." [Online]. Available: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

[176] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: A Python Toolbox for Scalable Outlier Detection," p. 7.

[177] "scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation." [Online]. Available: https://scikit-learn.org/stable/

[178] E. W. Patton, M. Tissenbaum, and F. Harunani, "MIT App Inventor: Objectives, Design, and Development," in *Computational Thinking Education*, S.-C. Kong and H. Abelson, Eds. Singapore: Springer, 2019, pp. 31–49. [Online]. Available: https://doi.org/10.1007/978-981-13-6528-7_3

[179] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," *arXiv:2010.08678 [cs]*, Mar. 2021, arXiv: 2010.08678. [Online]. Available: http://arxiv.org/abs/2010.08678

[180] "Neural Discrete Representation Learning," p. 10.

[181] A. Razavi, A. v. d. Oord, and O. Vinyals, "Generating Diverse High-Fidelity Images with VQ-VAE-2," *arXiv:1906.00446 [cs, stat]*, Jun. 2019, arXiv: 1906.00446. [Online]. Available: http://arxiv.org/abs/1906.00446

[182] K. Gregor, G. Papamakarios, F. Besse, L. Buesing, and T. Weber, "Temporal difference variational auto-encoder," 2019.

[183] K. Ding, Q. Zhou, H. Tong, and H. Liu, "Few-shot Network Anomaly Detection via Cross-network Meta-learning," *arXiv:2102.11165 [cs]*, Feb. 2021, arXiv: 2102.11165. [Online]. Available: http://arxiv.org/abs/2102.11165

[184] Y. Zhao, "yzhao062/MetaOD," May 2021, original-date: 2020-09-22T15:59:50Z. [Online]. Available: https://github.com/yzhao062/MetaOD

[185] A. Nagabandi, C. Finn, and S. Levine, "Deep Online Learning Via Meta-Learning: Continual Adaptation for Model-Based RL," Sep. 2018. [Online]. Available: https://openreview.net/forum?id=HyxAfnA5tm

[186] S. Zhang, F. Ye, B. Wang, and T. G. Habetler, "Few-Shot Bearing Anomaly Detection Based on Model-Agnostic Meta-Learning," *arXiv e-prints*, vol. 2007, p. arXiv:2007.12851, Jul. 2020. [Online]. Available: http://adsabs.harvard.edu/abs/2020arXiv200712851Z

[187] Y. Zhao, R. A. Rossi, and L. Akoglu, "Automating Outlier Detection via Meta-Learning," *arXiv:2009.10606 [cs, stat]*, Mar. 2021, arXiv: 2009.10606. [Online]. Available: http://arxiv.org/abs/2009.10606

[188] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "CutPaste: Self-Supervised Learning for Anomaly Detection and Localization," *arXiv:2104.04015 [cs]*, Apr. 2021, arXiv: 2104.04015. [Online]. Available: http://arxiv.org/abs/2104.04015

[189] V. Sehwag, M. Chiang, and P. Mittal, "SSD: A Unified Framework for Self-Supervised Outlier Detection," Sep. 2020. [Online]. Available: https://openreview.net/forum?id=v5gjXpmR8J

[190] J. Liu, L. Xiao, G. Liu, and Y. Zhao, "Active authentication with reinforcement learning based on ambient radio signals," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 3979–3998, Feb. 2017. [Online]. Available: https://doi.org/10.1007/s11042-015-2958-x

[191] T. Wu and J. Ortiz, "RLAD: Time Series Anomaly Detection through Reinforcement Learning and Active Learning," *arXiv:2104.00543 [cs]*, Mar. 2021, arXiv: 2104.00543. [Online]. Available: http://arxiv.org/abs/2104.00543

[192] G. Pang, A. v. d. Hengel, C. Shen, and L. Cao, "Deep Reinforcement Learning for Unknown Anomaly Detection," *arXiv:2009.06847 [cs, stat]*, Sep. 2020, arXiv: 2009.06847. [Online]. Available: http://arxiv.org/abs/2009.06847

[193] V. M. Patel, R. Chellappa, D. Chandra, and B. Barbello, "Continuous User Authentication on Mobile Devices: Recent progress and remaining challenges," *IEEE Signal Processing Magazine*, vol. 33, no. 4, pp. 49–61, Jul. 2016, conference Name: IEEE Signal Processing Magazine.

[194] N. Al-Naffakh, N. Clarke, and F. Li, "Continuous User Authentication Using Smartwatch Motion Sensor Data," in *12th IFIP International Conference on Trust Management (TM)*, ser. Trust Management XII, N. Gal-Oz and P. R. Lewis, Eds., vol. AICT-528. Toronto, ON, Canada: Springer International Publishing, Jul. 2018, pp. 15–28. [Online]. Available: https://hal.inria.fr/hal-01855982

[195] J. Zuo, H. Xia, S. Liu, and Y. Qiao, "Mapping urban environmental noise using smartphones," *Sensors*, vol. 16, no. 10, 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/10/1692

[196] H. Khan, A. Atwater, and U. Hengartner, "A Comparative Evaluation of Implicit Authentication Schemes," in *Research in Attacks, Intrusions and Defenses*, ser. Lecture Notes in Computer Science, A. Stavrou, H. Bos, and G. Portokalidis, Eds. Cham: Springer International Publishing, 2014, pp. 255–275.

[197] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2011, pp. 1609–1615, iSSN: 2153-0017.

[198] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, "Mobile phone based drunk driving detection," Jun. 2010. [Online]. Available: https://eudl.eu/doi/10.4108/icst.pervasivehealth2010.8901

[199] K. Kunze, G. Bahle, P. Lukowicz, and K. Partridge, "Can magnetic field sensors replace gyroscopes in wearable sensing applications?" in *International Symposium on Wearable Computers (ISWC) 2010*, Oct. 2010, pp. 1–4, iSSN: 2376-8541.

[200] G. M. Weiss, J. W. Lockhart, T. T. Pulickal, P. T. McHugh, I. H. Ronan, and J. L. Timko, "Actitracker: A Smartphone-Based Activity Recognition System for Improving Health and Well-Being," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct. 2016, pp. 682–688.

[201] R. Gouveia, E. Karapanos, and M. Hassenzahl, "How do we engage with activity trackers? a longitudinal study of Habito," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '15. New York, NY, USA: Association for Computing Machinery, Sep. 2015, pp. 1305–1316. [Online]. Available: https://doi.org/10.1145/2750858.2804290

[202] Z. Zhao, Y. Chen, S. Wang, and Z. Chen, "FallAlarm: Smart Phone Based Fall Detecting and Positioning System," *Procedia Computer Science*, vol. 10, pp.

617–624, Jan. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091200436X

[203] P. N. A. Fahmi, V. Viet, and C. Deok-Jai, "Semi-supervised fall detection algorithm using fall indicators in smartphone," in *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '12. New York, NY, USA: Association for Computing Machinery, Feb. 2012, pp. 1–9. [Online]. Available: https://doi.org/10.1145/2184751.2184890

[204] "Kansiz: Selection of time-domain features for fall... - Google Scholar." [Online]. Available: https://scholar.google.com/scholar_lookup?title=Selection%20of%20time-domain%20features%20for%20fall%20detection%20based%20on%20supervised%20learning&author=A.%20O.%20Kansiz&author=M.%20A.%20Guvensan&author=&author=H.%20I.%20Turkmen#d=gs_cit&u=%2Fscholar%3Fq%3Dinfo%3Ah2BYEAFe2-8J%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Den

[205] "The MobiFall Dataset: Fall Detection and Classification with a Smartphone: Security & Forensics Journal Article | IGI Global." [Online]. Available: https://www.igi-global.com/article/the-mobifall-dataset/116732

[206] Y. Lee, S. S. Iyengar, C. Min, Y. Ju, S. Kang, T. Park, J. Lee, Y. Rhee, and J. Song, "MobiCon: a mobile context-monitoring platform," *Communications of the ACM*, vol. 55, no. 3, pp. 54–65, Mar. 2012. [Online]. Available: https://doi.org/10.1145/2093548.2093567

[207] K. Lorincz, B.-r. Chen, G. W. Challen, A. R. Chowdhury, S. Patel, P. Bonato, and M. Welsh, "Mercury: a wearable sensor network platform for high-fidelity motion analysis," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09. New York, NY, USA: Association for Computing Machinery, Nov. 2009, pp. 183–196. [Online]. Available: https://doi.org/10.1145/1644038.1644057

[208] G. M. Weiss, J. L. Timko, C. M. Gallagher, K. Yoneda, and A. J. Schreiber, "Smartwatch-based activity recognition: A machine learning approach," in *2016*

IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), Feb. 2016, pp. 426–429, iSSN: 2168-2208.

[209] F. B. A. Ramos, A. Lorayne, A. A. M. Costa, R. R. d. Sousa, H. Almeida, and A. Perkusich, "Combining Smartphone and Smartwatch Sensor Data in Activity Recognition Approaches: an Experimental Evaluation," Jul. 2016, pp. 267–272. [Online]. Available: http://ksiresearchorg.ipage.com/seke/seke16paper/seke16paper_40.pdf

Appendix A

Appendix

A.1 Model Details

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, 2)]	0	
model_6 (Functional)	[(None, 2), (None, 2 79	input_5[0][0]	
model_7 (Functional)	(None, 2)	75	model_6[0][2]
dense_18 (Dense)	(None, 2)	6	input_5[0][0]
dense_19 (Dense)	(None, 16)	48	dense_18[0][0]
dropout_8 (Dropout)	(None, 16)	0	dense_19[0][0]
dense_20 (Dense)	(None, 1)	17	dropout_8[0][0]
dropout_9 (Dropout)	(None, 1)	0	dense_20[0][0]
dense_22 (Dense)	(None, 2)	4	dropout_9[0][0]
dense_21 (Dense)	(None, 2)	4	dropout_9[0][0]
tf.__operators__.add_4 (TFOpLam (None, 2)	0	dense_22[0][0]	
tf.math.square_2 (TFOpLambda)	(None, 2)	0	dense_21[0][0]

Table A.1 continued from previous page

Layer (type)	Output Shape	Param #	Connected to
tf.math.subtract_6 (TFOpLambda) (None, 2)	0	tf.__operators__.add_4[0][0]	
tf.math.square_2[0][0]			
tf.math.exp_2 (TFOpLambda)	(None, 2)	0	dense_22[0][0]
tf.math.subtract_7 (TFOpLambda) (None, 2)	0	tf.math.subtract_6[0][0]	
tf.math.exp_2[0][0]			
tf.math.reduce_sum_2 (TFOpLambd (None,)	0	tf.math.subtract_7[0][0]	
tf.convert_to_tensor_2 (TFOpLam (None, 2)	0	model_7[0][0]	
tf.cast_2 (TFOpLambda)	(None, 2)	0	input_5[0][0]
tf.math.multiply_7 (TFOpLambda) (None,)	0	tf.math.reduce_sum_2[0][0]	
tf.math.squared_difference_2 (T (None, 2)	0	tf.convert_to_tensor_2[0][0]	
tf.cast_2[0][0]			
tf.math.subtract_8 (TFOpLambda) (None,)	0	tf.math.multiply_7[0][0]	
tf.math.reduce_mean_4 (TFOpLamb (None,)	0	tf.math.squared_difference_2[0][0	
tf.math.abs_2 (TFOpLambda)	(None,)	0	tf.math.subtract_8[0][0]

Table A.1 continued from previous page

Layer (type)	Output Shape	Param #	Connected to
tf.math.multiply_6 (TFOpLambda) (None,)	0	tf.math.reduce_mean_4[0][0]	
tf.math.multiply_8 (TFOpLambda) (None,)	0	tf.math.abs_2[0][0]	
tf.__operators__.add_5 (TFOpLam (None,)	0	tf.math.multiply_6[0][0]	
tf.math.multiply_8[0][0]			
tf.math.reduce_mean_5 (TFOpLamb ()	0	tf.__operators__.add_5[0][0]	
add_loss_2 (AddLoss)	0	0	tf.math.reduce_mean_5[0][0]

A.2 Human Activity Recognition

Based on accelerometer and gyroscope data to detect complex activities, Johnson *et al.* [197] classify bad driving, and Dai *et al.* [198] have succeeded to classify drunk driving. Kunze *et al.* [199] and Abdulla *et al.* [54] have leveraged magnetometer data along with accelerometer data to detect the location of the device on the human body. Weiss *et al.* [200] proposed a health monitoring app called "Actitracker" that utilizes motion sensors and feeds them into a Random Forest classifier. Their solution uses a threshold set by the user for their detection and measures daily user activities. Gouveia *et al.* [201] assessed another app called "Habito", and concluded that historical logs of the user activity sensor data are not useful to the user and should be deleted on a daily basis for security and privacy reasons. Falling detection is a big use case for the health monitoring and human activity recognition domain. "FallAlarm" is an app presented by Zhao *et al.* [202] that was developed to detect human users falling and alarm their contacts. It is built on top of many contributions in the domain, like those of Fahmi *et al.* [203] and Kansiz *et al.* [204], which use time, frequency, and wavelet features for fall detection in their semi-supervised and supervised works, respectively. Similarly, he authors of "MobiFall" [205] have evaluated multiple models to detect the forward falls using hands, forward falls using knees, side-ward falls, and backward falls with up to 99% accuracy. Lee *et al.* [206], Lorincz *et al.* [207], and others had also contributed diseases classification models (Parkinson's, strokes, and epilepsy) and frameworks using machine learning which leverage motion data for detecting patients' movement patterns. Motion sensors from smartwatches and other gadgets have also been utilized in this domain. Weiss *et al.* [208] and Ramos *et al.* [209] used motion data from smartwatch to detect different activities and whether an individual is intoxicated, or drunk, with high accuracies. The utilization of these gadgets are outside the scope of our work, as we are focused on making the solution applicable to as many users as possible. Nevertheless, if gadgets do exist, they can be integrated into our framework for additional contextual awareness as covered in Chapter 2. Academic works have used motion sensors to locate users indoor with great accuracies, classify diseases, detect imperfections unique to every device, sense human characteristics and emotions, detect activities and the status of human users surrounding the device, and authenticate users via a variety of parameters

and techniques. More on those in health monitoring, use status indoor localization, device fingerprinting, personal traits, and keystroke authentication using motion data is available in a recent survey published by Masoud *et al.* [55].

A.3 HMOG Data-set Stats

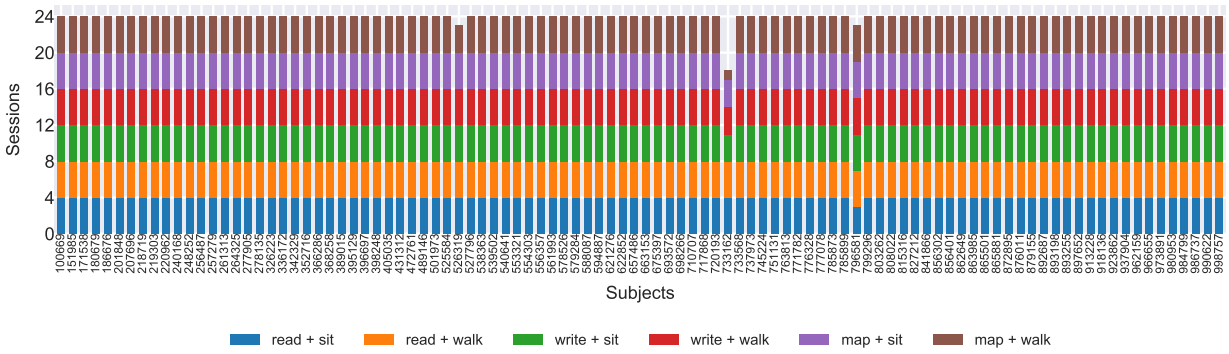

Figure A.1: HMOG sessions categories and total count per subject. Adapted from [15]

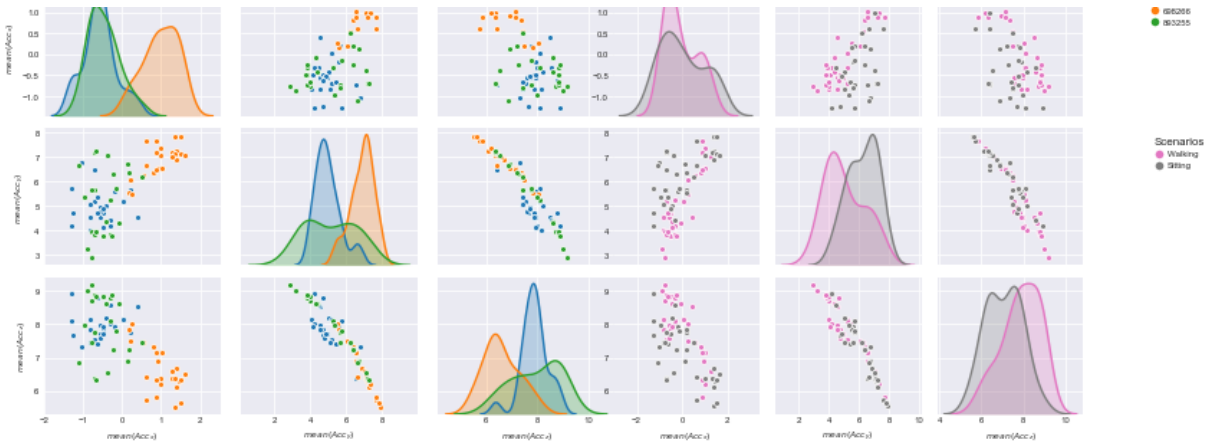

Figure A.2: HMOG accelerometer data between subjects and walking and sitting settings. Adapted from [15]

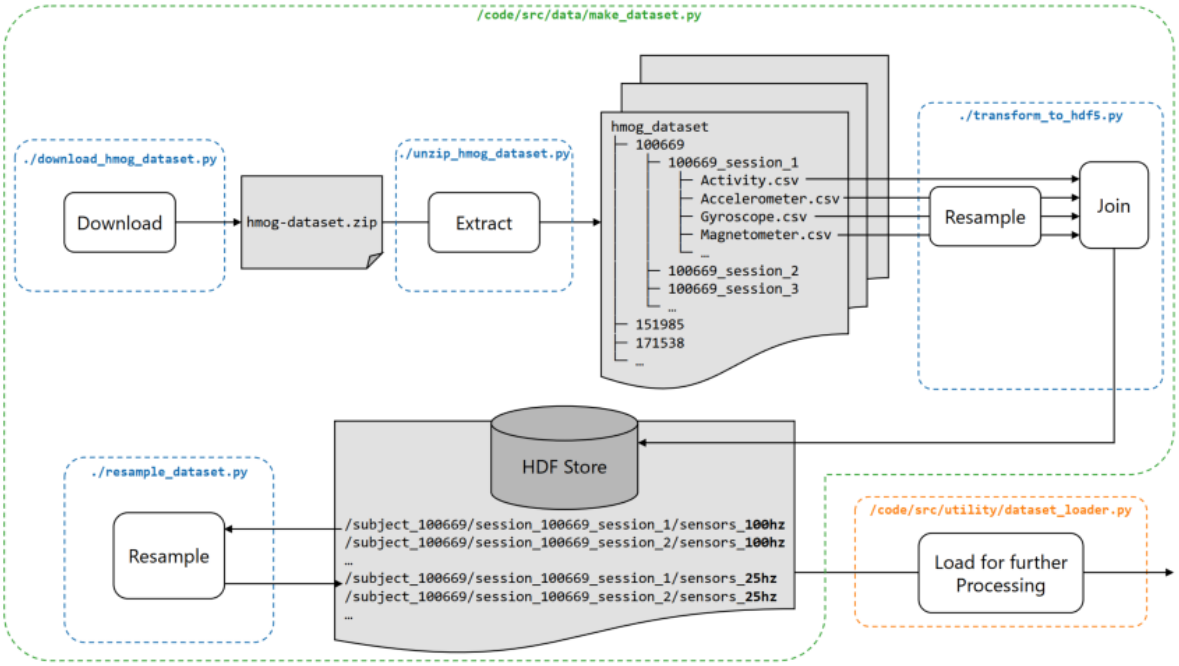

Figure A.3: Initial data preparation procedure along with corresponding python software modules. Adapted from [15]

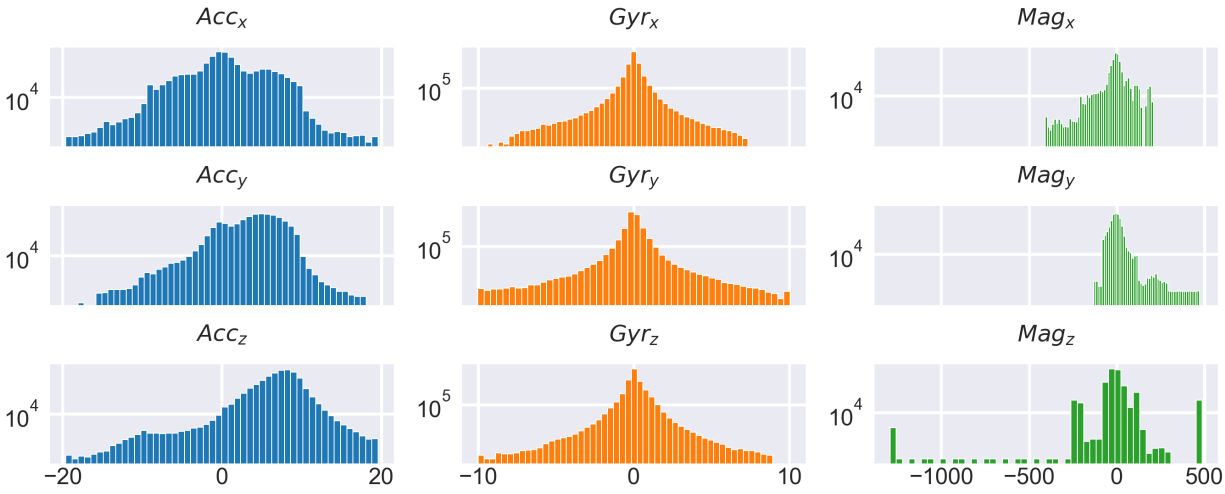

Figure A.4: HMOG inertial sensors data distribution on log scale. Adapted from [15]

Figure A.5: HMOG inertial data pairwise relationships for three subjects. Adapted from [15]

Feature Set	Number of resulting features	Description
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)
Cross-mean Rate	3	Fluctuation of the signal

Table A.2: The time domain's commonly computed features and number of resulting features per a three-axis sensor. Adapted from [13, 19]

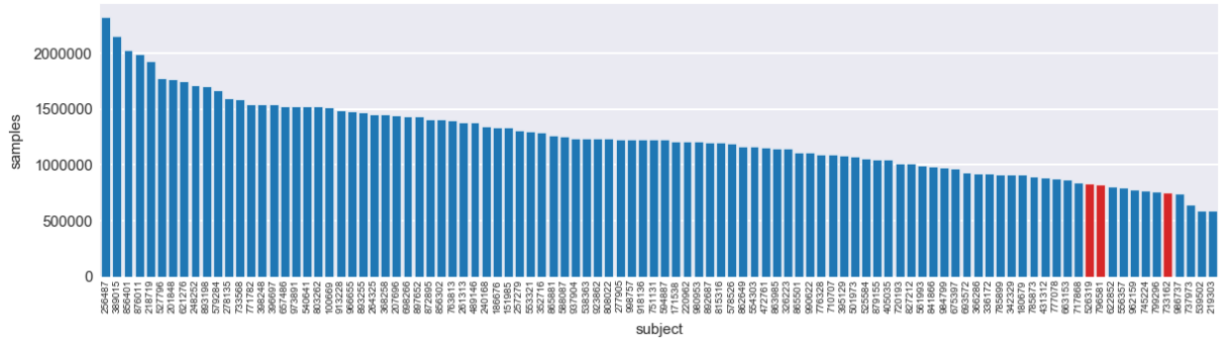

Figure A.6: HMOG samples count per subject. Adapted from [15]

Figure A.7: All users in HMOG and all their sessions durations in minutes.

Feature Set	Description
Entropy	Dispersion of signal
Peek occurrences	Number of peeks occurred
Time between peeks	Average time between peeks
Slope between peeks	the steepness and direction of the peeks
Peek to peek signal value	difference between the maximum amplitude in the negative direction and in the positive direction
Max Latency	longest interval between two consecutive iterations
Min Latency	shortest interval between two consecutive iterations
ALAR	Absolute Latency to Amplitude Ratio

Table A.3: The frequency domain's commonly computed features per a three-axis sensor. Adapted from [13, 19]

Feature Set	Number of resulting features	Description
Correlation coefficient	3	Relationship between two axes
Cosine similarity	3	Pairwise cosine similarity measurements between axes
Co-variance	3	Pairwise co-variances between axes
DTW	3	Dynamic Time Warping
Band Power	3	Dynamic Time Warping
SNR	3	Signal to Noise Ratio

Table A.4: Miscellaneous commonly computed features and number of resulting features per a three-axis sensor. Adapted from [13, 19]

A.4 Deep Feature Extractor Pare-meters

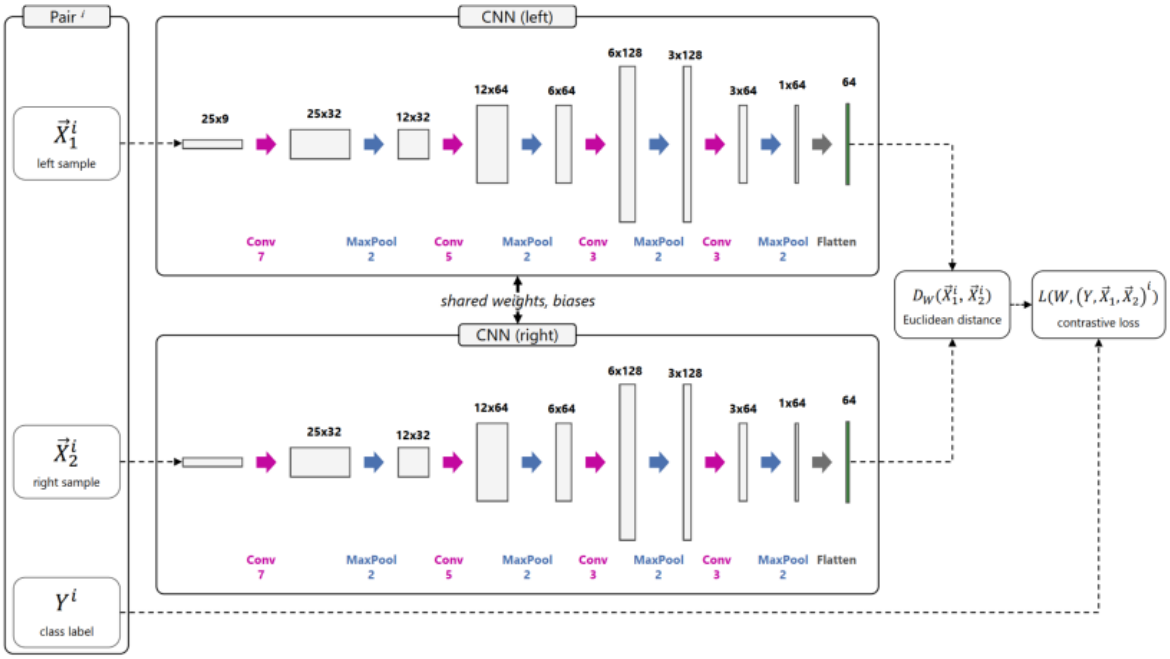

Figure A.8: Siamese Convolutional Neural Network architecture with 1D filters proposed by [16]. All filters use padding and the vector of the last CNN layer (marked in green) is considered the deep feature representation. Adapted from [13]

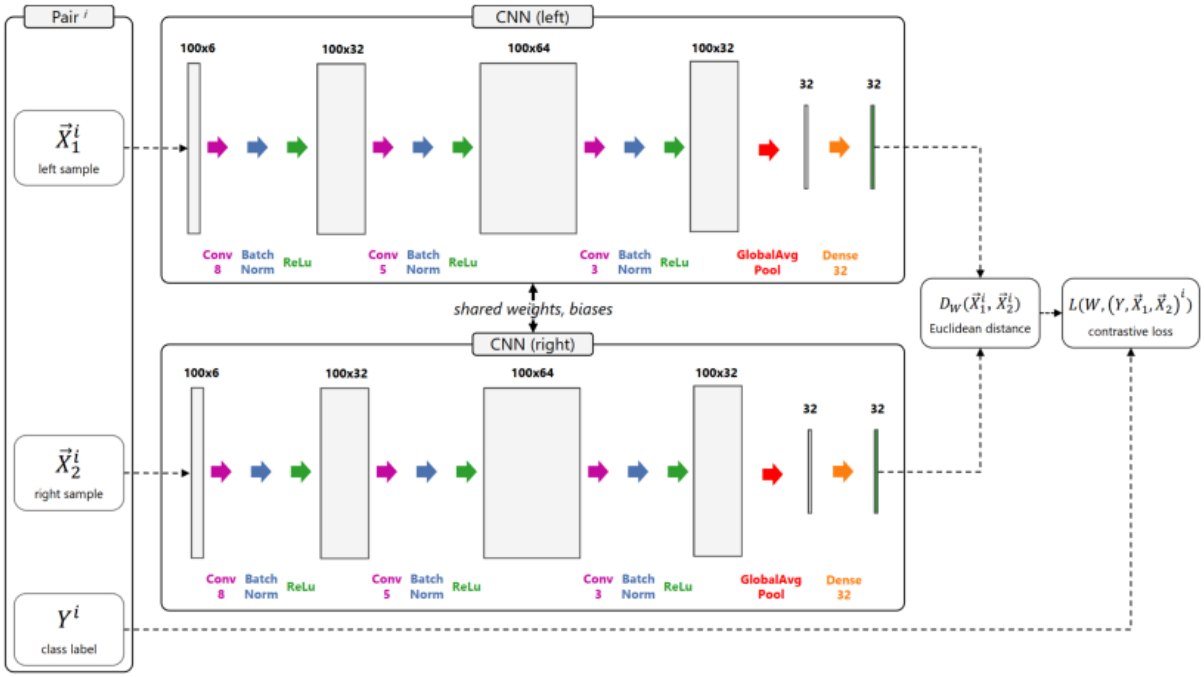

Figure A.9: Siamese Convolutional Neural Network architecture with FCN sub networks proposed by [15] as modeled after [17]. All filters use padding and the vector of the last layer (marked in green) is considered the deep feature representation. Adapted from [13]

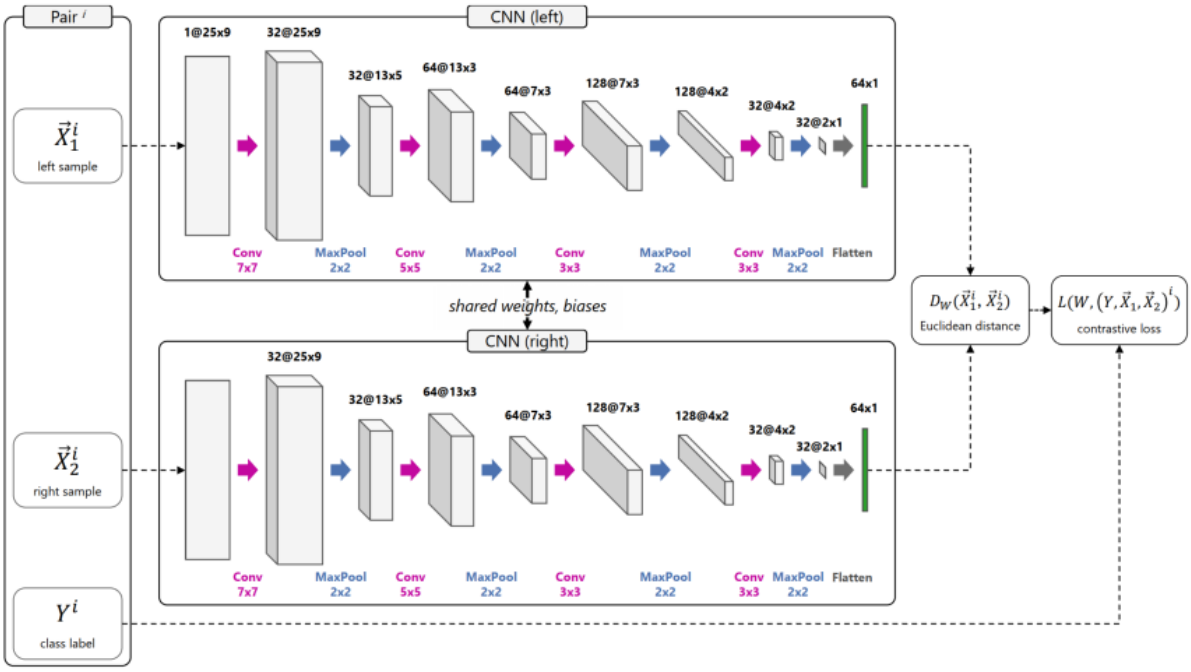

Figure A.10: Siamese Convolutional Neural Network architecture proposed by [16]. All filters use padding and the vector of the last CNN layer (marked in green) is considered the deep feature representation. Adapted from [13]

Parameter	Value	Comment
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 $\sim$64 dimensional output vector.
Distance Function	Eucl. Dist.	Euclidean distance.
Loss	Contr.Loss	Contrastive loss function

Table A.5: Siamese CNN parameters. Adapted from [20, 13]

Parameter	Variations
CNN architecture	CNN (2D filters), CNN (1D filters), FCN (1D filters)
Window size	0.5, 1, 2, 5 sec.
Sampling rate	100 Hz, 25 Hz
Body Modes	{sit, walk}, {walk}, {sit}

Table A.6: Variations of parameters tested for Siamese CNN approach. Adapted from [13]

A.5 Models

VAE [109] is the graphical Bayesian inference probabilistic variant of the Auto-Encoder. As opposed to the Auto-Encoder where the encoder and decoder implement two complementary deterministic transformations, in VAE it is *a distribution* that is being learned and the output is *a draw* from that underlying distribution. β-VAE [159] is a variation of the VAE with the goal to discover disentangled latent factors. PCA (Principal Component Analysis) [133] is a linear transformation often used for dimensional reduction to allow for easy data exploration and analysis. OCSVM (One-class Support Vector Machine) [134] is a linear algorithm that aims at learning a decision boundary to group the data points. MCD (Minimum Covariance Determinant) [135] is an estimator of multivariate location and scatter. HBOS (Histogram-based Outlier Score) [136] is a proximity based technique that models uni-variate feature densities using histograms with a fixed or a dynamic bin width. KNN (K-Nearest neighbors) [137] is a proximity based model where for each data point, the whole data set is examined to extract the k data points with the most similar feature values. LOF (Local Outlier Factor) [138] is a proximity based method that captures exactly the relative degree of isolation of an object from its surrounding neighborhood. CBLOF (Cluster-Based Local Outlier Factor) [139] is a proximity based method where the outlier score is computed by the distance of each instance of the data to its respective cluster center multiplied by the instances belonging to its cluster. ABOD (Angle-based Outlier Detection) [140] is a probabilistic Proximity based technique built for high-dimensional data anomaly detection tasks. IForest (Isolation Forest) [141] is an ensembling technique that is based on the usage of numerous isolation trees, a tree structure constructed effectively to isolate every single instance. FeatureBag [142] is an ensemble technique applying Feature Bagging for Outlier Detection.