Aalto University
School of Electrical
Engineering

Master's Programme in Automation and Electrical Engineering

Motion-Sensor-Based Anti-Spoofing in a Mobile Identity Verification Application

Erkka Rantahalvari

Master's Thesis
2025

© 2025

This work is licensed under a Creative Commons "Attribution-NonCommercial-ShareAlike 4.0 International" license.

Aalto University
School of Electrical Engineering

Author Erkka Rantahalvari

Title Motion-Sensor-Based Anti-Spoofing in a Mobile Identity Verification Application

Degree programme Automation and Electrical Engineering

Major Electronic and Digital Systems

Supervisor Prof. Simo Särkkä

Advisors Prof. Olli Silvén, Dr. Zinelabidine Boulkenafet

Collaborative partner Candour Oy

Date 30 September 2025 **Number of pages** 64+2 **Language** English

Abstract

Remote identity verification solutions face new challenges: with advances in generative AI, a single facial image of a victim can suffice for adversaries to synthesize lifelike video models and conduct deepfake video-injection attacks. A practical mitigation to such attacks would be to include additional data modalities in the verification process, reducing reliance on the video feed alone, complicating potential attacks. On consumer mobile devices, one widely available modality would be motion sensors, for example, accelerometers and gyroscopes. This thesis examines the potential of combining motion-based sensor data collected during Candour ID's selfie-capture phase with state-of-the-art time-series classification and anomaly-detection methods. For this purpose, a data set was collected from 30 subjects (10–15 samples each), totaling 375 time-series sequences of multi-sensor data sampled during selfie captures. Three distinct, comprehensive benchmarks were conducted: (i) multi-class classification to identify individual users, (ii) an anomaly-detection evaluation to flag possible spoofing attacks or unusual behavior, and (iii) a 10-shot, semi-supervised one-class learning -based verification (anomaly detection) to verify individual users without relying on a multi-class classifier. The results indicate that a selfie-capture-centered motion data analysis check shows promise as a complementary security layer supporting Candour's face-matching engine.

Keywords time series classification, motion sensors, remote identity verification

Aalto-yliopisto
Sähkötekniikan
korkeakoulu

Tekijä Erkka Rantahalvari

Työn nimi Liikeantureihin perustuva huijauksenesto tunnistautumissovelluksessa

Koulutusohjelma Automaatio- ja sähkötekniikka

Pääaine Elektroniset ja digitaaliset järjestelmät

Työn valvoja Prof. Simo Särkkä

Työn ohjaajat Prof. Olli Silvén, Dr. Zinelabidine Boulkenafet

Yhteistyötaho Candour Oy

Päivämäärä 30.9.2025 **Sivumäärä** 64+2 **Kieli** englanti

Tiivistelmä

Henkilöllisyyden etävarmennusratkaisut kohtaavat uusia haasteita: generatiivisen tekoälyn kehityksen myötä jo yksi uhrin kasvokuva voi riittää hyökkääjille elävänoloisten videomallien tuottamiseen, ja deepfake-videoinjektiohyökkäysten toteuttamiseen. Eräs tapa puolustautua tällaisilta hyökkäyksiltä olisi sisällyttää todennusprosessiin lisää datamodaliteetteja, jolloin riippuvuus pelkästä kamerasyötteestä vähenisi ja mahdolliset hyökkäykset vaikeutuisivat. Mobiililaitteilla yksi tällainen laajasti saatavilla oleva datamodaliteetti ovat liikeanturit, kuten kiihtyvyys- ja gyroskooppianturit. Tässä työssä tarkastellaan mahdollisuutta yhdistää selfie-kuvauksen aikana kerätty liikepohjainen anturidata tämänhetkisiin huipputason aikasarjaluokittelu- ja poikkeavuuksien havaitsemismenetelmiin. Tätä varten kerättiin aineisto 30 henkilöstä (10–15 näytettä kustakin), sisältäen yhteensä 375 anturidataan perustuvaa aikasarjasekvenssiä, jotka tallennettiin selfie-kuvausten aikana. Työssä suoritettiin kolme erillistä kokeilua: (i) yksittäisten käyttäjien tunnistaminen käyttäen moniluokkaisia aikasarjojen luokittimia, (ii) aikasarja-anomalioiden havaitseminen esityshyökkäysten tai muun epätavallisen toiminnan paljastamiseksi, ja (iii) 10 näytteen osittain ohjattuun yhden luokan oppimiseen perustuva todennus, jossa yksittäiset käyttäjät todennetaan ilman moniluokkaista luokitinta. Tulokset viittaavat siihen, että selfie-kuvauksen aikana kerätyn liikedatan analyysitarkistus tarjoaa lupaavan, Candourin kasvojentunnistusta täydentävän turvakerroksen.

Avainsanat aikasarjojen luokittelu, liikeanturit, henkilöllisyyden etätunnistus

Preface

This thesis was written with financial assistance admitted by the Technology Industries of Finland Centennial Foundation (Teknologiateollisuuden 100-vuotissäätiö). I would like to thank them for their generous grant.

I would like to thank my advisors Olli Silvén and Zinou Boulkenafet, and my supervisor Simo Särkkä, for their encouragement, guidance, and comments on the thesis. Additionally, I would like to thank everyone who participated in the data collection campaign. You made this thesis possible.

Finally, I would like to thank Harri Martinmäki for extending this opportunity to me, and Olli Silvén again for the idea of the thesis. It has been an absolute privilege to conduct my thesis on such an interesting topic.

Tampere, 30 September 2025

Erkka Juhani Rantahalvari

Contents

Symbols and abbreviations

Symbols

$\mathbf{B}$	Magnetometer measurement vector
d	Number of channels
ε	Small numerical constant
∞	Positive infinity
$\mathbb{N}$	Natural numbers $(1, 2, \ldots)$
$\mathbb{R}$	Real numbers
t	Time index
τ	Decision threshold

Operators

Δx_t	First difference of x at time t ($\Delta x_t = x_t - x_{t-1}$)
$\bar{x}$	Mean of x
$\mathbf{x}^\top$	Transpose of $\mathbf{x}$
$\hat{x}$	Estimated/approximated value of x
$\in$	Set membership
$\|\mathbf{x}\|_2$	Euclidean norm of $\mathbf{x}$
$\min\{x_1, \ldots, x_n\}$	Minimum of the listed values
$\sum_{t=1}^{T} x_t$	Sum of x_t from $t = 1$ to T

Abbreviations

AD	Anomaly Detection
AI	Artificial Intelligence
AUROC	Area Under Receiving Operating Characteristic
CPU	Central Processing Unit
COTE	Collective of Transformation Ensembles
DP	Dynamic Programming
DTW	Dynamic Time Warping
EER	Equal Error Rate
FAR	False Acceptance Rate
FRR	False Rejection Rate
GPU	Graphics Processing Unit
IF	IsolationForest
IMU	Inertial Measurement Unit
k-NN	k-Nearest Neighbors
LSTM	Long Short-Term Memory
ML	Machine Learning
MRZ	Machine Readable Zone
NFC	Near Field Communication
OCSVM	One-Class Support Vector Machine
OOF	Out-of-Fold
RBF	Radial Basis Function
RFID	Radio Frequency Identification
RIdV	Remote Identity Verification
ROCKET	Random Convolutional Kernel Transform
ROCKAD	ROCKET Anomaly Detector
ROC	Receiving Operating Characteristic
SOTA	State-of-the-art
TPR	True Positive Rate
TSAD	Time Series Anomaly Detection
TSC	Time Series Classification
UCR	University of California, Riverside
WTSAD	Whole Time Series Anomaly Detection

1 Introduction

Over the past decade, remote identity verification (RIdV) has become essential, as an increasing number of sensitive services have moved online. Tasks that once always demanded a face-to-face visit, such as opening a bank account, applying for government benefits, taking out a loan, or even clearing border control, are now routinely completed remotely (Fatima et al., 2024). In physical settings, proving one's identity normally involves little more than a brief visual inspection of an ID document, typically a passport, an ID card or a driver's license by a clerk. The simplicity and practicality of this interaction is often taken for granted, and reproducing the same level of trust and ease in a digital setting demands considerably more thought and complexity.

Existing simpler approaches involve proving the user's ownership of a physical identity document by simply taking a photo of it and sending it. However, this does not guarantee the sender of the photo to be the owner of the document. A common countermeasure is then to request the user to also send a selfie portrait, which is then compared to the picture in the ID document with a visual inspection by a human or an AI-based biometric comparison engine. However, this solution alone is still insufficient because an impostor could, in theory, obtain both the ID document and a selfie of its owner. Remote identity verification applications such as Candour ID counter this by requesting the user to take both photos in a single verification session within the application, and by applying various liveness detection methods to ensure that the person and the ID document are physically present in front of the camera, rather than displayed on another screen, theoretically forcing the ID document and its owner to be within physical proximity of each other. In addition to this, Candour ID, along with some other RIdV applications, also utilize the built-in near-field communication (NFC) hardware embedded in most smartphones today, to read the contents of the radio frequency identification (RFID) chip in the passport or ID card, making spoofing attempts with forged documents significantly more difficult.

As impressive as all this sounds, this technology is still not foolproof. Rapid progress in generative AI has recently made it possible to create highly realistic video "deepfakes". It has been demonstrated that with nothing more than a single photo, an attacker can produce a lifelike, animated model of someone's face good enough to threaten even the current state-of-the-art liveness detection methods (Pennsylvania State University, 2023; Vincent, 2022; Y. Zhang et al., 2023b). This, combined with the possibility of using rooted devices where video feed of these reconstructions could be streamed to the application from a source other than the camera of the device, poses serious risks that can compromise the integrity of such remote RIdV platforms (Carta et al., 2022).

Although such attacks may appear unlikely due to the significant expertise, time, and effort required, the deployment of RIdV applications in the banking sector significantly increases the incentive for malicious actors. This increases the need to find new ways to enhance the security of these solutions. One such way proposed by Candour Oy and many other RIdV solution vendors would be to add new data modalities to analyze, in order to reduce reliance on camera feed alone.

As these applications are mobile device -based, the user typically performs the verification by physically holding the device in their hand(s). The presence of built-in accelerometers or inertial measurement units (IMUs) in modern mobile devices introduces a potential additional data modality that could be utilized more broadly in such verification settings. With the significant improvements in mobile computational capabilities in the recent decade, it has become feasible to collect continuous, reliable, and timestamped streams of sensor data directly from the application layer without a significant impact in power consumption or performance, even in budget-level devices. This enables the representation of such measurements as a high-quality time series. When combined with recent, fast pace of advancement in artificial intelligence and time series analysis (Middlehurst et al., 2024b), new opportunities have emerged in detecting and classifying potential spoofing attempts based on motion-related behavioral patterns.

The research aim of this thesis comprises of two parts: the first is to determine the accuracy with which a real user interacting with the Candour ID application can be distinguished from a spoofing attempt purely from the way they are handling the device by comparing various semi-supervised anomaly detection techniques. The second is to benchmark the current state-of-the-art Time Series Classification (TSC) algorithms against each other on the aforementioned handling patterns to determine which category of algorithms performs the best in user verification. To this end, the thesis introduces and publicly releases CanSelfie, a uniform-length time series data set containing various sensor channels, collected from 30 voluntary participants while they were performing the selfie capture phase in the Candour ID application.

The selfie capture phase was chosen because it is the point in the verification flow where detectable spoofing measures are assumed to be most likely taken, for example, where someone tries to present an AI-generated face or a secondary image or video to the application. In the anomaly detection part, most samples in the CanSelfie dataset are used to train the algorithms in a semi-supervised manner without distinguishing between individual users, while the remainder is used to test them against physically mimicked time series samples resembling anomalous behavior.

Additionally, a 10-shot, one-class verification experiment was conducted using the same anomaly-detection techniques. The goal was to determine how well individual users can be recognized and zero-effort impostors rejected when training is limited to the enrolled users' own samples. In the time-series classification part, supervised time-series classification algorithms are benchmarked on the CanSelfie dataset to assess which algorithm can be best converted to accept enrolled users while rejecting others. In both parts, the set of sensor channels is varied within each benchmark to determine which algorithm–data configuration yields the best performance.

The remainder of this thesis is divided into four chapters. Chapter 2 provides a brief overview of concepts most relevant to this thesis, such as details about the application's verification flow and possible spoofing methods that could exploit it. It then goes on to provide background information on time series representations of motion data and reviews current state-of-the-art methods in both time series classification and anomaly detection. Chapter 3 describes the research methods, including the type of data collected, the environment in which the benchmark will be conducted, and the metrics used to evaluate performance. Chapter 4 presents visualizations of the collected dataset and overall results of the benchmark. Finally, Chapter 5 contains discussion on the interpretation and reliability of the results.

2 Background

The aim of this chapter is to introduce the key concepts and background knowledge relevant to this thesis. This includes the Candour ID application itself, motion data and its representation as time series, and brief overviews of the fields of time series classification and anomaly detection. The chapter concludes with user verification, emphasizing its relationship to anomaly detection in terms of commonalities and divergences.

2.1 The Candour ID Application and Potential Spoofing Approaches

A common use of Candour's solution is via verification invitations. The inviting party sends a personal verification invitation to each invitee, who then open them on their mobile devices via a QR code or a deep link. Candour's mobile user interface, the Candour ID application, is available for both Android and iOS devices. The user interaction flow during verification in app version 2.3.7 can be divided into 3 main phases:

1. Capturing a photo of the machine readable zone (MRZ) side of the user's ID document.

2. Reading the information and picture stored in the NFC tag of the ID document.

3. Taking a selfie portrait.

Typically, the completion of all 3 phases is required by the inviting party, although in some usage contexts, the inviting party has obtained the ID document and/or a previous selfie image of the invitee, and the invitee only needs to perform the selfie capture phase to verify their identity. Provided the threat actor has obtained the invitation link to such "selfie-only verification", the only step that needs to be spoofed is the selfie phase. The approaches the threat actors could take include video injection attack with a deepfake or real-time face morphing, or in a lower-effort attack, simply pointing the selfie camera to another screen displaying a picture or a video of the victim's face.

Recently, the more discussed and studied approach of these is the video injection attack. This attack type requires the adversary to have an access to the source code of the application, which can be done by decompiling (and possibly deobfuscating it) (Carta et al., 2022). After this is achieved, the application can be manipulated with different rooting tools such as "Frida" for Android, which can be used to analyze and modify the application code on run-time. This allows the adversary to inject snippets of code during the verification process that could allow to change the source of video from device camera to a pre-made deepfake video constructed from the victim's face.

A video injection attack could be combined with replayed or synthetic IMU streams, since Mohamed et al. (2017) show that fake motion/position data can be fed to Android apps without rooting the device (noting that some pathways have been restricted since

2017). But including such a maneuver on top of rooting the device, decompiling and deobfuscating the application, preparing deepfake video reconstructions of the victim's face and injecting it into the application on runtime would require a massive amount of preparing and competence. In addition to this, each added data stream should also be synchronized together and perfectly timed with the selfie capture moment. However, there would not likely even be a need for synthetic or pre-recorded motion sensor data streams if an adversary opted to morph their face in real time with a deepfake filter instead of a pre-recorded video. This would allow for real-time motion sensor data matching with the camera's movements and selfie capture time organically (with a small delay). Open-source tools such as DeepFaceLive (DeepFaceLive contributors, 2024) enable real-time face morphing with deepfake filters without requiring extensive GPU resources: in practice, a consumer-grade GPU (for example, a RTX 2070 as stated in the repository) is sufficient to generate a live, manipulated video stream on a local machine. This, however, doesn't take into account *user-specific* motion patterns, the recognizability of which is also studied in this thesis.

Considering the points made above, adding a "motion analysis engine" to support Candour's facial recognition engine could improve the security of the solution against such video injection and live morphing attacks significantly. Such motion data analysis engine would act strictly as a supporting feature to the facial comparison engine, and theoretically should never obstruct its function by flagging genuine verification attempts as non-genuine. A rejection from the facial comparison engine would always override an acceptance from the motion analysis engine, but a rejection from the motion analysis engine would have some weight in the decision. Therefore, solutions with low false rejection rates (FRR) are favored over those with low false acceptance rates (FAR). Figure 1 provides a visual overview of the idea.

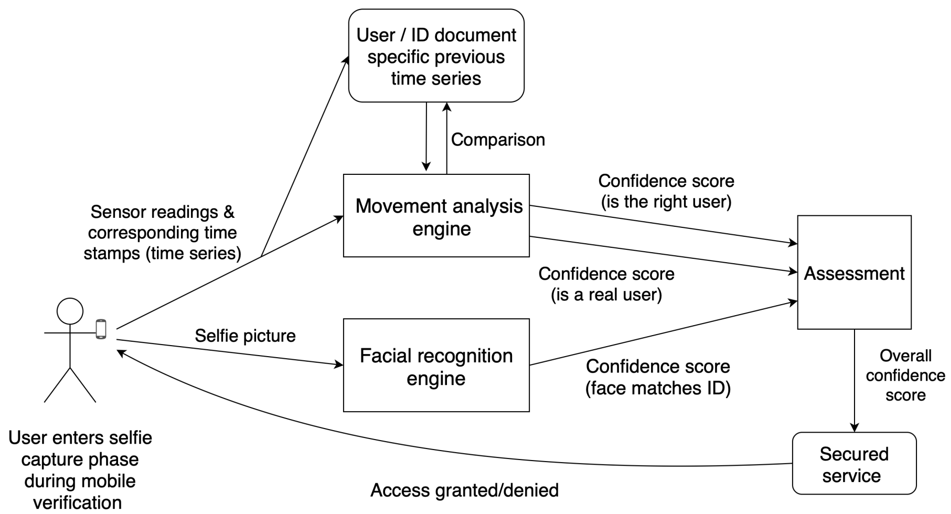


Figure 1: Motion/movement analysis engine integrated as a part of Candour's verification solution.

As one of Candour ID's strengths in the industry is its speed, an additional motion data analysis engine should also never bottleneck the facial recognition engine in compute time. In essence, there should never be a situation where the facial comparison is ready but motion analysis is still pending. Thus, algorithms/solutions with low delays at validation time are also favored.

2.2 Motion Data and Mobile Devices

Modern mobile devices contain motion-sensing hardware that enables capture of device dynamics during handling. The raw outputs are naturally expressed in the form of a time series, from which informative features can be derived. This subsection introduces the typical sensing capabilities embedded in modern consumer smart devices, the representation of motion data as time series, and the implications for feature extraction.

2.2.1 Motion Sensors in Mobile Devices

Motion sensors in modern mobile devices typically include a triaxial accelerometer, a triaxial gyroscope, and a triaxial magnetometer (Leimhofer et al., 2025). Although the magnetometer is often classified as a "position" or "environment" sensor (Android Developers, 2025), it is routinely used in motion/orientation estimation and, together with the accelerometer and gyroscope, forms the sensor device commonly referred to as the inertial measurement unit (IMU). It is therefore included here alongside the motion sensors.

In mobile applications, the accelerometer is likely the most widely utilized motion sensor. It provides the necessary information to recognize when the device is physically interacted with, such as lifted from a surface, grasped by its user, or tilted into full-display (landscape) mode. Moreover, the accelerometer simultaneously measures gravitational acceleration (9.81 m/s^2), the distribution of which across the three sensor axes displayed in Figure 2 reveals information on the orientation of the device. However, this gravitational component can interfere with the detection of dynamic motion and is therefore sometimes removed through low-pass filtering (Android Developers, 2023). Once gravity is estimated, it can be subtracted from the raw signal to yield the linear acceleration. These filtered derivations of the original accelerometer readings are provided by the respective sensor frameworks of Android and iOS (Android Developers, 2023; Apple Inc., 2025). In some devices, these signals (linear acceleration and gravity) are also extracted directly with hardware (Android Developers, 2023).

In mobile devices, the gyroscope is generally reserved for specialized use cases such as camera stabilization and applications that rely on rotational movement (for example, games). It measures angular velocity about the three axes of the device (rad/s). Integrated over time it provides relative orientation (yaw/pitch/roll) but is susceptible to bias and drift, so it is often fused with the accelerometer and magnetometer.

The third motion sensor type, the tri-axial magnetometer, measures the local geomagnetic flux-density vector. On its own it provides an absolute heading (yaw)

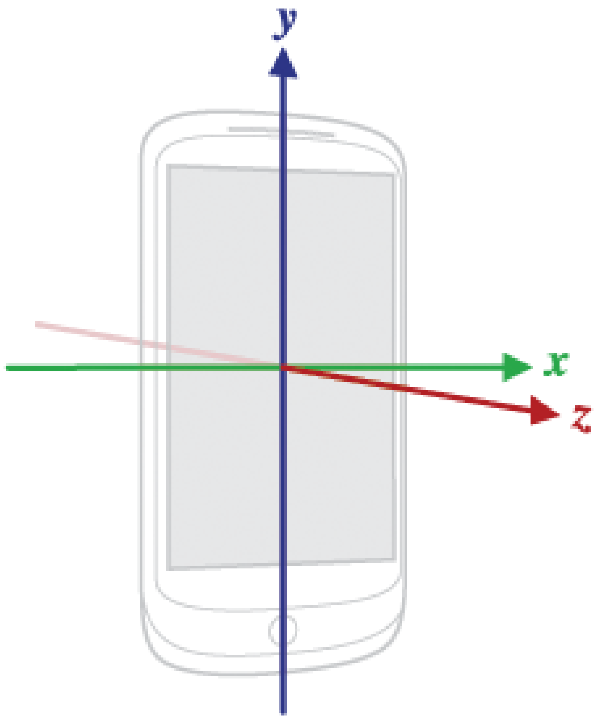


Figure 2: Android device coordinate system used to reference tri-axial accelerometer readings. Axes follow the Android sensor convention. Source: Android Developers (2024).

with respect to the magnetic north, which can be used in navigation-related applications where compass-like functionality is needed. It is also used to correct bias (drift) in 3-axis gyroscopes, making them more reliable (Android Developers, 2023).

In consumer mobile devices, the coverage of these three sensor types is varied. An umbrella study conducted this year reports that nearly all Android and iOS devices released between 2014 and 2024 integrate a tri-axial accelerometer, that gyroscopes are present in all iOS models since 2014 and in over 80% of Android models from 2021–2023 with an upward trend, and that magnetometers reach a coverage of around 80–90% on Android since 2014 and 100% on iOS (Leimhofer et al., 2025). Thus, the accelerometer can be regarded as the most readily available IMU source on Android, while on devices running iOS, all three of these sensor types should in principle be always available.

2.2.2 Time Series Representations of Motion Data

In many applications, motion sensor measurements can be represented as discrete time series, given that they are sampled at a fixed or near-fixed rate, thus forming chronologically ordered sequences that reflect physical movement and overall dynamics (Lima et al., 2019; Zhou et al., 2020). In univariate settings where only one sensor (and axis) is sampled, each measurement is a scalar value,

$$x_t \in \mathbb{R}, \qquad t \in \mathbb{N},\ 1 \le t \le T.$$

Here, x_t denotes the scalar measurement at time step t, and $T \in \mathbb{N}$ is the length of the time series (the total number of measurements). The entire univariate sequence is written as

$$\mathbf{x}_{1:T} = (x_1, \ldots, x_T),$$

which denotes the ordered collection of all T measurements. In multi-channel configurations, where multiple sensors or sensor axes are sampled simultaneously, each time step is represented by a d-dimensional column vector,

$$\mathbf{x}_t = \left[\, x_{t,1}, \ldots, x_{t,d} \,\right]^{\top} \in \mathbb{R}^d,$$

where $x_{t,i}$ denotes component $i \in \{1, \ldots, d\}$ at time t, and $d \in \mathbb{N}$ is the number of channels. Stacking the T time-step vectors with time in columns yields

$$\mathbf{X}_{1:T} = \left[\, \mathbf{x}_1 \ \cdots \ \mathbf{x}_T \,\right] \in \mathbb{R}^{d \times T},$$

with d rows (channels) and T columns (time). This data matrix is also known as a **multivariate time series**. As a concrete example, a triaxial accelerometer yields

$$\mathbf{A}_{1:T} = \begin{bmatrix} a_{x,1} & \cdots & a_{x,T} \\ a_{y,1} & \cdots & a_{y,T} \\ a_{z,1} & \cdots & a_{z,T} \end{bmatrix} \in \mathbb{R}^{3 \times T},$$

whose rows correspond to the x, y, and z axes and whose columns are synchronized time samples. This form makes windowing and per-axis preprocessing straightforward.

The length T of sensor-based time series obviously varies depending on the context and phenomenon being measured. In most publicly available datasets, such as the UCR time series classification archive, the lengths of time series containing motion sensor measurements of human movement gestures typically fall somewhere around a few hundred points (100-400), depending on how much real time (seconds) the gesture takes to perform (Dau et al., 2019). Additionally, another determining factor for the length of such a time series is the sampling frequency, the number of measurements taken per time unit. When collecting such sensor-based time series with a smartphone, the operating system often limits the frequency at which sensor data can be polled. For example, the Android operating system offers the ability to poll motion sensors at a frequency of 200 Hz or more (Android Developers, 2025), but sampling at 50 Hz can be deemed to be more than enough to capture most (if not all) relevant features associated with human hand movements. For example, Yamane et al. (2025) showed that when classifying human behavior with data from motion sensors mounted on the wrist and chest, reducing the sampling frequency from 100 to 50 Hz and even 10 Hz did not significantly affect the recognition accuracy. Similarly, Sitová et al. (2015) studied the potential of using smartphone IMU data collected while subjects were typing for user classification/verification, and showed no discernible drops in performance when comparing between 50 Hz and 100 Hz, and only a very minor deterioration at 16 Hz. Major performance drops could only be noticed when the sampling rate was lowered to 5 Hz. In some contexts however, such as electrohysterogram analysis, sequences are commonly sampled at frequencies as high as 200 Hz or more for increased accuracy (Alexandersson et al., 2015; Y. Zhang et al., 2022).

2.2.3 Manual Feature Extraction from Mobile Device Motion Data

The raw time series data collected from motion gestures could be used for various purposes (such as classification or anomaly detection) as it is, but a common practice is to additionally extract handcrafted features from it.

For example Sitová et al. (2015) extracted multiple features relating to the user's grasp resistance (how much the phone physically moves during taps) and grasp stability (how fast the perturbations disappear after taps) were computed by observing accelerometer data sampled while subjects were typing on the virtual keyboard of the device. Similarly, simpler features such as typing speed (how frequently taps happen), duration of individual taps, contact size (how many pixels are being pressed), difference in z-accelerometer's reading before and after tap event and so on. These features can also be further refined in countless ways, such as by taking the standard deviation, averages, minimum or maximum of them.

Similarly, more recent studies on behavioral biometrics concerning user verification on mobile devices have also utilized data modalities outside of motion sensors. For example, Qiao et al. (2025) collected the x and y coordinates, applied pressure and contact area of the user's touches and evaluated a Siamese neural network using these four channels (and their timestamps) as its core input features for user authentication experiments. The advantage of these touch-dynamics-based features over those based on motion sensors is that they are not as prone to performance degradation due to changes in usage context, as in previous studies, to obtain reliable authentication performance, the subjects were required to operate the smartphone in a fixed usage context (Shen et al., 2022). However, in the context of this thesis, there's very little touch dynamics -related data available to collect, as the user is simply taking a selfie, with auto-capture performing even the sole potential touch gesture required to complete the task. The selfie verification process is also generally a short and "high-stakes" process, that the users prefer to perform carefully, making it likely less prone to noise than more casual tasks.

2.3 Time Series Classification

The field of Time Series Classification has seen massive improvements in the last decade. In 2012, the dynamic time warping algorithm (DTW), which had already been known for 40 years at that time, was still among the SOTA algorithms. However, in the 2024 bake-off study tens of new algorithms had surpassed k-NN DTW and other distance-based approaches in classification accuracy by over 10% margin in most datasets (Middlehurst et al., 2024b).

2.3.1 Classical Methods

In its earliest form, the field of time series classification (TSC) relied heavily on distance-based pattern matching. The nearest neighbour (NN) rule offered a simple yet strong baseline: a k-NN classifier paired with a straightforward distance measure such as Euclidean. In this paradigm, a series is compared with all labeled reference series by calculating a distance score, and the label of the closest example (under the chosen distance) is returned. Early studies routinely used 1-NN Euclidean as the reference against which alternative measures were compared (Cover & Hart, 1967; Keogh & Kasetty, 2003). Building on this distance-based foundation, Dynamic Time Warping (DTW) emerged as an elastic alternative that aligns sequences by allowing local time distortions. Although originally introduced in the 1970s for pattern matching rather than classification (Sakoe & Chiba, 1978), DTW was quickly combined with a 1-NN classifier, yielding the widely used "1-NN DTW" approach. Notably, 1-NN DTW remained the de facto baseline (and often state of the art) across many benchmarks well into the 2010s (Bagnall et al., 2017).

The most popular variation of DTW is based on dynamic programming (DP) techniques, which compute the optimal alignment between two sequences by minimizing the cumulative distance under certain constraints, such as continuity and monotonicity. The classic DP formulation constructs a cost matrix where each element represents the distance between points in the two time series, and the optimal warping path is found by recursively accumulating the minimum cost from adjacent cells. Below is the *forward* dynamic programming formulation introduced by Sakoe and Chiba (1978):

$$D_{i,j} = d(x_i, y_j) + \min \begin{cases} D_{i-1,j} \\ D_{i,j-1} \\ D_{i-1,j-1} \end{cases}$$

with the initial conditions:

$$\begin{aligned} &D_{0,0} = 0, \\ &D_{i,0} = D_{0,j} = \infty \qquad (i, j > 0). \end{aligned}$$

Here, $d(x_i, y_j)$ is the distance between samples x_i and y_j; $D_{i,j}$ is the cumulative DTW cost of the optimal path up to indices i and j. Indices i and j index the two time series vectors $\mathbf{x}$ and $\mathbf{y}$, respectively.

Despite its effectiveness, DTW has several limitations that have motivated the development of alternative approaches. The classical form is computationally expensive, with a time complexity of $O(n \cdot m)$ given two time series of length n and

m (Bringmann et al., 2023). When both time series are of equal length, the time complexity can be simplified to $O(n^2)$. Given a data set with N reference samples, the computational complexity for the classification of a single input series increases to $O(N \cdot n^2)$, since the DTW distance must be calculated for each reference series in the data set. Despite optimizations (such as the Sakoe-Chiba band) that reduce the time complexity, 1-NN DTW still remains compute-intensive on long sequences and large reference sets. Moreover, 1-NN DTW is a distance-based technique, not a learned model. More examples can help coverage, but also slow down queries. These limits pushed a move toward more trainable, ML-adjacent approaches, yet DTW still remains a strong baseline, especially with little data or data with notable and consistent temporal misalignments.

2.3.2 Recent Developments

In the recent decade, along with machine learning (ML) in general, the field of TSC has seen a sharp rise in popularity (Middlehurst et al., 2024b). A number of creative and diverse approaches have been proposed, a large portion of them following a transform–classify pipeline consisting of two steps: (i) a representation step that maps each series into a feature space, and (ii) training step where a standard learner is trained with labeled data. Figure 3 presents a block diagram of such approaches. Novelty/creativity in new algorithms is typically concentrated on finding new feature representations, while the classifier is often an off-the-shelf implementation (for example, a ridge classifier or a random/rotation forest). Consequently, algorithms are usually identified by their data representation family rather than by the specific classifier used.

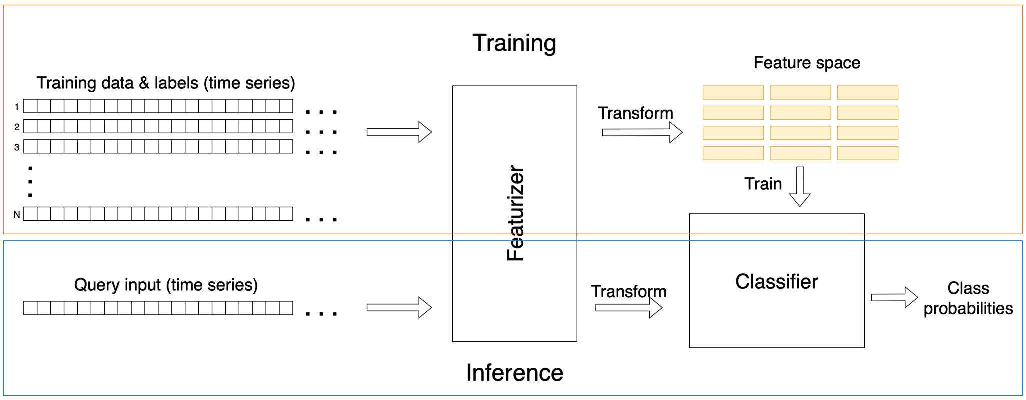


Figure 3: Block diagram of a typical TSC transform-classify pipeline.

Bagnall et al. (2017) compared tens of such time series classification algorithms in a wide benchmark covering 85 datasets, with 100 train-test split resamples on each. The approaches were divided into 6 categories [1]

[1] Unless otherwise stated, the definitions in items 1–6 follow Bagnall et al. (2017).

1. **Whole series similarity** is synonymous with distance-based, and refers to techniques in which two time series are compared directly, and a distance score is calculated. Examples of this include the simple Euclidean distance, as well as various elastic distance measures such as DTW and Weighted DTW (WDTW).

2. **Interval-based methods** extract features from randomly or systematically selected intervals within a time series. Typically many different random intervals and classifiers are generated on each interval, and resulting predictions are ensembled.

3. **Shapelet-based methods** focus on identifying short, distinctive subsequences called shapelets that can appear at any position within a time series. These patterns are phase-independent, meaning their location in the series doesn't matter. Classification is then based on whether one or more of these shapelets are present in the time series.

4. **Dictionary-based methods** break a time series into short subsequences, convert them into symbolic words, and count how often each word appears. These counts are then used to build a histogram that serves as input for a classifier.

5. **Combination methods** are algorithms that merge two or more of the previously mentioned approaches into a single classifier.

6. **Model-based methods** fit a generative model to each time series and then compare the series by measuring the similarity between their fitted models. However, these methods were not included in the benchmark.

The performance of the then new SOTA algorithm "Collective of Transformation Ensembles" (COTE) outperformed 1-NN DTW in classification accuracy on average by about 8% (Bagnall et al., 2017). It was also the only hybrid algorithm in the 2017 bake-off, as its structure consists of a weighted ensemble of 35 separate classifiers covering a total of four categories of data representations.

The situation was revisited again in 2024, and at that point tens of new and diverse approaches had surpassed the older ones such as COTE in performance and compute time (Middlehurst et al., 2024b), the new best performing new algorithms being HIVE-COTE v2.0 (HC2) and MR-HYDRA. This benchmark covered 142 datasets, and the algorithm families were categorized in a revised manner to *distance-based*, *feature-based*, *interval-based*, *shapelet-based*, *dictionary-based*, *convolution-based*, *deep learning -based* and *hybrid* approaches.

The new categories closely resemble those used in the previous 2017 bake-off, with the addition of *feature-based*, *convolution-based*, and *deep learning-based* approaches, and the elimination of *model-based*.

Feature-based approaches are a recent and popular approach that extract descriptive statistics from time series to use as input for standard classifiers. Since these features usually summarize the entire series, they are considered series-to-vector transformations. Libraries such as hctsa provide thousands of such descriptors (Fulcher & Jones, 2017), while the much lighter catch22 set shows that as few as

22 well-chosen features can achieve competitive accuracy (within ≈ 2–3% of 1-NN DTW) in 90 UCR/UEA problems (Lubba et al., 2019).

Convolution-based feature extractors such as ROCKET (Random Convolutional Kernel Transform), MiniROCKET, and MultiROCKET transform each series with thousands of randomly parameterized sliding windows (1D kernels), convert the kernels' outputs into simple statistics (for example, proportion of positive values, max), and feed the resulting high-dimensional vector to a linear classifier (Dempster et al., 2020; Middlehurst et al., 2024b).

Deep learning-based time series classification methods use Deep Neural Networks (DNNs) implemented in frameworks such as TensorFlow or PyTorch (Fawaz et al., 2019). These are models trained end-to-end using backpropagation to optimize their weights and biases directly from raw time series data, eliminating the typical feature extraction or generation phase. Deep learning is currently the most popular field of study in TSC, as it has dominated research output since the 2017 bake-off (Middlehurst et al., 2024b).

The new close second SOTA TSC algorithm MR-HYDRA (Dempster et al., 2023) is a combination of the convolutional approach of MultiROCKET and HYDRA (Hybrid Dictionary-ROCKET Architecture), which combines both convolution and dictionary -based techniques. The output feature vectors of both algorithms are concatenated into a single feature vector, which is then used to train a ridge classifier.

Several open-source time series classification libraries are available for the public. In particular, aeon Middlehurst et al. (2024a) provides reference implementations for many methods evaluated in the UCR "bake-off" studies (Bagnall et al., 2017; Middlehurst et al., 2024b), and includes reproducible baselines. Its feature transforms are also available as standalone functions, enabling them to be combined with other classifiers, or reused for other purposes such as anomaly detection.

2.4 Time Series Anomaly Detection

An anomaly in a time series can broadly be defined as a single point or a sequence of points with unexpected or abnormal values. The goal of time series anomaly detection (TSAD) is to identify such anomalies in a given time series (Sørbø & Ruocco, 2023). Time series anomaly detection techniques are frequently applied in situations where there is no inherent need to classify samples into predetermined classes but to detect whether a sample differs from other samples in a data set up to a predetermined threshold.

2.4.1 Taxonomy of Time Series Anomalies

Although time series can exhibit a diverse range of different anomalies depending on data sources and context, a widely used taxonomy groups them into three broad categories Chandola et al. (2009):

1. **Point anomalies:** Single data points that significantly deviate from the expected value.

2. **Contextual anomalies:** Data points that are considered anomalous only within a specific context, such as time of day, season, or location in a trend. For example, a temperature of 20°C may be normal during summer in Finland but anomalous in winter.

3. **Collective anomalies:** A sequence or group of data points that, taken together, form an anomalous pattern, even if individual points are not anomalous on their own. These often reflect structural changes or pattern violations in the time series.

These categories could be further refined into nine specific anomaly types, as introduced in a large TSAD benchmark study by Schmidl et al. (2022):

1. **Extremum anomaly (Point):** A single data point that exhibits a sharp peak or dip, deviating significantly from its surroundings. Often interchangeable with point anomaly.

2. **Variance anomaly (Contextual):** An abnormal change in the variability of the time series, such as becoming erratic or unusually smooth in a region. This change is typically only anomalous in certain contexts.

3. **Mean anomaly (Contextual):** A shift in the mean value over a segment of the time series. The anomaly depends on the context in which the shift occurs (for example, seasonal or trend-aware baselines).

4. **Frequency anomaly (Contextual):** A change in the periodicity or oscillation rate of a pattern.

5. **Pattern-shift anomaly (Collective):** A known, expected pattern occurs in a distorted form: for example, delayed, stretched, compressed, or shifted.

6. **Amplitude anomaly (Contextual or Point):** A sudden change in signal magnitude. It may appear as a point anomaly if abrupt, or contextual if the amplitude is sustained.

7. **Pattern anomaly (Collective):** The appearance of a new, unexpected sequence of values in the time series. These are typically longer subsequences that don't match historical motifs.

8. **Platform anomaly (Collective):** A segment of the series that flattens at an abnormal constant value, representing a sustained abnormal state.

9. **Trend anomaly (Collective):** A consistent upward or downward trajectory that deviates from the expected stationary or cyclic behavior of the series.

Some of these anomalies, can appear in motion sensor time series due to errors or noise. For example, it is known that MEMS sensors typically embedded in consumer electronics are sometimes prone to internal noise (Almeida et al., 2022; Suvorkin et al., 2024). A standard solution is to apply a preprocessing step such as a low-pass filter (for example, a Butterworth filter or a moving average filter) to smooth the signal and reduce false positives in anomaly detection (Chen et al., 2016).

However, contextual and collective anomalies can be more difficult to mitigate with such preprocessing means. In the setting of this thesis, one type of collective anomaly could appear in the accelerometer time series data if the user performs the selfie capture sequence in a moving, accelerating vehicle. Similarly, a contextual anomaly could appear in situations where the user is interrupted while taking the selfie, causing them to move abruptly. Both anomaly categories could appear in the same time series in situations such as when the user decides to change from a sitting down position to a lying-down position while performing the selfie capture.

2.4.2 Time Series Anomaly Detection Techniques

The field of time series anomaly detection (TSAD) is broad and encompasses a wide variety of techniques, ranging from statistical models to modern machine learning and deep learning -based approaches (Schmidl et al., 2022). The choice of technique often depends on the characteristics of the data, such as seasonality, trend, noise level, and the type of anomaly that is being targeted. In a comprehensive study by Schmidl et al. (2022), 71 distinct TSAD techniques were benchmarked on 976 data sets. These techniques were categorized into seven distinct method families according to their field of origin: Classic Machine Learning, Signal Analysis, Stochastic Learning, Statistics, Outlier Detection, Data Mining and Deep Learning. It was found that there's no single one-size-fits-all solution, as different techniques had strengths in detecting different anomaly types. For example, autoencoders (originating from the field of Deep Learning) performed the best at detecting pattern anomalies while GrammarViz, commonly utilized in Data Mining, performed the best in finding trend anomalies. It was also concluded that "Deep learning approaches are not (yet) competitive despite their higher processing effort on training data". However, they still formed the best-performing techniques in detecting many individual anomaly types.

It should be noted that the algorithms evaluated in this study would typically output a vector of anomaly scores for each *point* in a given time series. Some algorithms would also output scores for multiple overlapping subsequences of the time series, which would be averaged and assigned to each point under a given subsequence. The anomalous time series would also be accompanied with a corresponding ground truth vector of equal length containing binary values indicating whether each point is normal (0) or anomalous (1). For each time series, evaluation metrics such as AUC-ROC were computed from the AD algorithm's point-wise score vector and its corresponding ground truth label vector. Within each dataset, these individual series metrics were averaged to obtain dataset-level performance scores. Finally, the unweighted mean of those dataset-level scores were computed, giving the overall performance of each TSAD algorithm.

In the context of this thesis, the situation is different, as ground-truth vectors are not available and the analysis is performed on the time series samples as complete sequences.

2.4.3 Whole Time Series Anomaly Detection

As noted above, most time series anomaly detection methods focus on flagging individual points or short subsequences as anomalous, rather than assigning a single label to an entire series. In contrast, whole-series anomaly detection in which each series is classified as “normal” (0) or “anomalous” (1) has only recently been formalized and remains underexplored. To date, only a few works (namely, Theissler et al. (2023)) address this problem directly. The mentioned study explored the potential of adapting the Random Convolutional Kernel Transform used in TSC to whole time series anomaly detection, coining the algorithm "ROCKAD" (Random Convolutional Kernel Transform Anomaly Detector). The algorithm can be divided into 3 steps: (i) Feature extraction using the ROCKET’s convolution-based feature extractor, (ii) Power transformation on the ROCKET feature space, to transform it to a more Gaussian distribution and (iii) applying a k-NN detector on the transformed feature space, calculating the mean distance to the k nearest neighbors. In a benchmark conducted by Theissler et al. (2023) ROCKAD proved the current state-of-the-art algorithm, easily outperforming other well-known TSAD algorithms adapted into whole TSAD (WTSAD).

However, in biometric user authentication settings, when collected biometric data is expressed in the form of a time series, the same problem of flagging whole time series to a "normal" or an "anomalous" class is often simply referred to as "user verification", where samples with deviations from the real user are flagged as anomalous. Often, instead of using an off-the-shelf do-it-all anomaly detection algorithm such as ROCKAD, the pipeline involves a custom feature extraction sequence that takes into account the actual nature and context of the data. For example, Sitová et al. (2015) collected data while users typing on a virtual keyboard, and features such as user grasp resistance and grasp stability were computed from IMU readings, from which even more features were computed, which were then input to a one-class verifier such as a one-class support vector machine (OCSVM) to obtain the resulting anomaly score.

2.5 User Verification

As introduced in the previous section, user verification asks a different question: rather than flagging unusual behavior, the goal is to decide whether a measured sample belongs to the claimed or *enrolled* user or not. Similarly, but also in contrast to time series classification, which assigns an input sample to one of N classes, verification is a binary accept/reject decision tied to a specific identity. In practice, a verifier must accept genuine samples from the right person and reject samples from everyone else, including ordinary "zero-effort" impostors who simply present their own data without trying to imitate anyone.

The verification process in many applications is typically organized into an enrollment step and an operational step. During enrollment, a small reference dataset is created for a given user from genuine samples. During operation, new, unseen samples are compared to the enrolled reference and a distance score is calculated. The verification attempt is then either accepted or rejected depending on a chosen threshold. If the score is higher than the threshold, the attempt is rejected. The rejection threshold can be set in a user-specific manner, as variation between individual sequences varies between users. Because part of the purpose of verification is to reject unknown others, this is typically treated as an open-set decision rather than closed-set classification (Jain et al., 2011).

In user verification, compared to anomaly detection, the negative class has a different meaning. Anomaly detection labels anything unusual as "anomalous", which may include noise, context effects, sensor spikes, motion in a vehicle, or other distribution shifts. In user verification, the negative class is specifically "not the claimed user." Context-affected or noisy genuine samples should therefore still be treated as genuine in ground truth: if they are rejected, they count as false rejects, which should be mitigated by all means (ISO/IEC, 2021).

As discussed in the previous section, whole time series anomaly detection (WTSAD) also assigns a single label to an entire series, but "anomalous" there simply means "not like the other samples." As a result, noise and context-related effects are valid reasons to label a sequence as anomalous, and in this setting, to reject it. That said, WTSAD algorithms such as ROCKAD, as well as other k-nearest-neighbour approaches, can in principle be used for user verification in a simple way. Genuine enrollment samples for each user are transformed into the chosen feature space and stored as that user's reference set. A new input sample is transformed into the same feature space and its distance to the nearest neighbour(s) among the enrolled user's references is computed. If this distance is below a threshold τ, the attempt is accepted, otherwise it is rejected. The decision threshold τ can be calibrated from genuine data only on a development split by choosing the quantile from inner evaluation distance scores that achieves a target FRR. Evaluation then follows biometric practice: compute the false reject rate (FRR) on held-out genuine samples, the false accept rate (FAR) on background (impostor) samples, and report the equal error rate (EER) (ISO/IEC, 2021).

3 Research Materials and Methods

This chapter documents the material and methods used in the study. First, the data source and acquisition setup are described, including the collection application, preprocessing steps, and the procedure to simulate sequences resembling spoofing behavior for the anomaly detection experiment. The resulting final data set is then summarized. Next, the benchmarking setup is specified, covering the computational environment and which anomaly detection techniques and TSC algorithms are included in each benchmark. Finally, the training and evaluation protocols are presented, specifying the cross-validation and decision threshold calibration strategies used in the three benchmarks and the final evaluation metrics.

3.1 Data Set Acquisition

To study the potential of analyzing motion patterns during selfie capture, a dedicated dataset needed to be collected, as no suitable public dataset was available. This subsection describes the data acquisition setup and goes through the pre-processing steps applied prior to analysis.

3.1.1 Data Collection in the Candour ID Application

A special in-house version of the Candour ID application was developed to collect the data. The application was set up to start data collection in the background immediately after the user opens the selfie camera, and stop it after the result (success or failure) is obtained from the backend. The sensor polling rate was set at 50 Hz for all channels. The data collected involved the following:

1. Raw acceleration (x, y, z) from the 3-axis accelerometer (m/s^2).
2. Angular velocity (x, y, z) from the device gyroscope (rad/s).
3. Geomagnetic field strength (x, y, z) from the magnetometer (μT).
4. Linear acceleration (gravity removed) for all three axes (m/s^2).
5. Gravity vector for all three axes (m/s^2).

All of the above mentioned readings also contained their corresponding timestamps. The time stamps of events such as opening of selfie camera, selfie capture and closing of selfie camera were also stored.

The data collection sessions took place in person and were conducted using a smartphone (Huawei Honor 200 8Gb/512Gb). The location varied between sessions, but the set-up always involved a chair or a couch. The goal was to collect 12 sequences per participant, but due to a couple of mistakes made in keeping count and errors with saving data in the application, only 10 full sequences were collected from two participants, and 13-15 from 11 participants. This number may seem low, but reflects possible deployment demands. The sequences were always collected in a sitting position and consecutively with no significant breaks in between. Of the 30 participants, 20 were male and 10 were female.

To make the data collection easier and smoother for the participants, the process was optimized so that they only needed to complete the selfie taking phase instead of the typical whole verification flow, where the user is also requested to take a picture of their identity document. Each participant was first instructed through the process, which involves the following steps, the user interface screens of which are displayed in Figure 4:

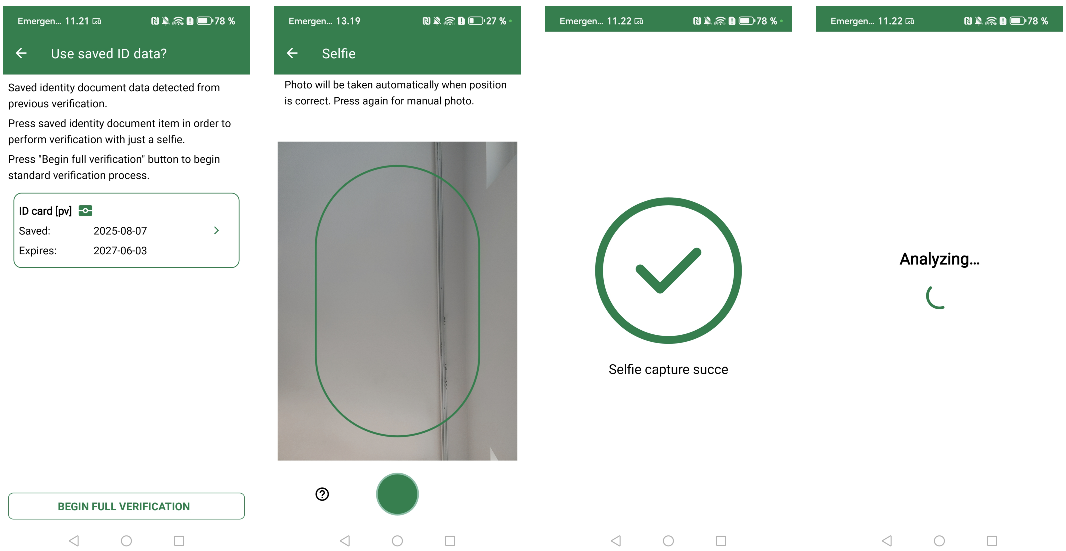
1. The participant opens the verification invitation by scanning a QR code in the application.

2. On the welcome page, the participant selects "ID card" option, which opens the selfie camera.

3. The participant aligns their face according to the on-screen instructions above the camera feed.

4. (Optional) The participant manually captures the selfie by pressing the shutter button.

5. The participant sees the “selfie capture successful” animation and waits for the server to return the verification result.



(a) Phase 1: ID selection **(b)** Phase 2: Selfie capture **(c)** Phase 3: Capture success animation **(d)** Phase 4: Waiting for result

Figure 4: User interfaces of four phases of the data collection workflow on Android.

A small number of participants opted to manually press the selfie button instead of waiting for the camera to auto-capture. Some also alternated between these two approaches, but majority still kept adjusting and waited for auto-capture. The time spent on a given selfie session varied a lot, as at times the auto-capture didn't trigger as easily, and occasionally the participant didn't immediately start attempting to capture the selfie after opening the camera.

3.1.2 Data Preprocessing and Representations

The collected sensor logs were transferred from the mobile device's memory to a PC via Android Studio's device explorer. Here, it was found that the timestamp intervals for each measurement channel were not fully uniform, as there seemed to be an offset with a median of 8-10 milliseconds, alternating between negative and positive. This translated to around half a frame at 50 Hz, and as the offset did not accumulate throughout the session, this shouldn't affect the performance of the TSC algorithms, as M. Liu et al. (2025) found that even when timestamp variation coefficient is fixed at 0.9 (almost a full frame shift on every measurement), the performance in gesture classification tasks falls only 0-3% relative to baseline, depending on the data set and the model used, when using deep learning based methods.

Upon further examination of the data set, it was also noticed that due to an odd bug in the data collection application, the time series sequences of two subjects were sampled at 400 Hz instead of the intended 50 Hz. This issue was fixed by downsampling the sensor logs to 50 Hz by only keeping every 8th sample and discarding the rest, as was done by Sitová et al. (2015). A small portion of the sequences sampled at 400 Hz also had a significantly lower real average sampling frequency (calculated from mean interval between timestamps), one of the sequences dropping as low as 337 Hz. This could have happened due to various reasons, such as increased CPU traffic or lower than normal OS resource constraints during the verification, demonstrating that using a lower sampling rate could be more reliable in delivering more steady and uniform streams of sensor data. The aforementioned 337 Hz sequence was removed from the data set for the sake of uniformity. It should also be mentioned that all sequences downsampled from 400 Hz had a slightly higher real sampling frequency as when compared to those sampled at 50 Hz (52 vs 49 Hz). Although this is unfortunate for the overall uniformity of the data set, it can simulate a hint of device heterogeneity, as real sampling frequencies tend to vary slightly between device models and operating systems (Android Developers, 2025; Stisen et al., 2015).

To keep the time series uniform length, and to reduce the amount of irrelevant features, the approach was to include n amount of data points before and and m amount after the selfie capture time stamp. As the fastest subjects successfully captured their selfies in just above 2 seconds, n was capped at around 100 data points, translating to roughly 2 seconds. However, there were plenty of data points available after the selfie capture, as sensor data was still being polled while the application was waiting for the verification result from the back end. The shortest post-selfie result acquisition time was around 6 seconds, translating to 300 samples.

Later, it was discovered that the data collection setup contained a major problem: the subjects sat facing the same direction throughout all their respective sessions. As a significant portion of the sessions were conducted in unique locations, there was a "free lunch" context leakage in the magnetometer readings, where some users could be identified solely on which direction they were facing instead of motion patterns, which are the main topic of this thesis. Also, the heading of the user shouldn't have any effect on the legitimacy of the verification attempt, and fitting into it would likely lead to false rejections in real-world deployment. In addition to this, factors such as the the smartphone model, the building materials of the room and nearby metal objects can affect the local geomagnetic field strength readings significantly (Xie et al., 2014). The only magnetometer features that could be included would be relative changes in the measurements throughout a given sequence. Applying a high-pass filter could remove the background offset that reveals the user's heading, but the amplitude would still be affected by local field strength. To emphasize relative variations throughout the time series while suppressing device and environment specific offsets, magnetometer vectors were additionally normalized to unit length and first-order temporal differences were taken (unit-vector differencing), leaving primarily changes in heading:

Given magnetometer vectors $\mathbf{B}_t \in \mathbb{R}^3$ for $t = 1, \ldots, T$ where T is the length of the time series, the sequence-level mean is first removed so only deviations from the average field remain:

$$\hat{\mathbf{B}}_t = \mathbf{B}_t - \bar{\mathbf{B}}, \qquad \bar{\mathbf{B}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{B}_t .$$

Next, amplitude information is discarded by normalizing each centered vector to unit length (with a small $\varepsilon > 0$ for numerical stability):

$$\mathbf{u}_t = \frac{\hat{\mathbf{B}}_t}{\|\hat{\mathbf{B}}_t\|_2 + \varepsilon}, \qquad \varepsilon > 0.$$

Finally, only the change in direction is kept by taking the difference of each successive data point; the first sample is set to zero:

$$\Delta\mathbf{u}_t = \begin{cases} \mathbf{0}, & t = 1, \\ \mathbf{u}_t - \mathbf{u}_{t-1}, & t = 2, \ldots, T. \end{cases} .$$

Figure 5 demonstrates the effect of these three steps applied on a the x-axis magnetometer sequences of one participant.

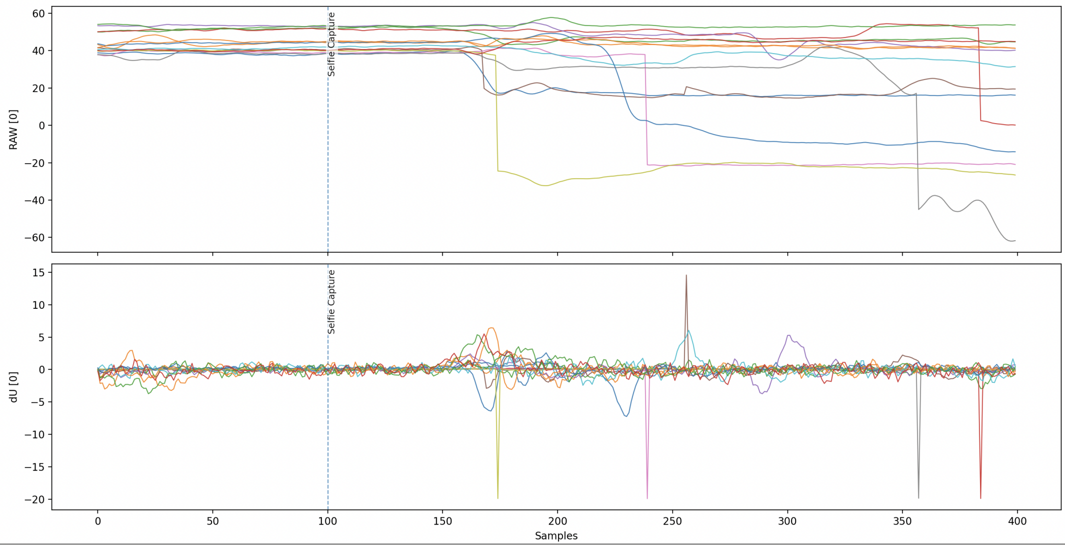


Figure 5: Effect of applying demeaning, unit vector normalizing and first-differencing on magnetometer readings (below graph).

3.1.3 Anomalous Behavior Simulation for Test Set

In order to evaluate the anomaly detection approaches, anomalous time series resembling spoofing-related behavior needed to be obtained. These sequences were physically mimicked and could be divided into 2 broad categories, displayed in Figure 6

- Phone is kept either horizontally on the table, or vertically on a mount throughout the selfie capture sequence, and a picture of a face is displayed to the camera from another phone until auto capture is activated. This means that there would be very little detectable movement (if not none). It was hypothesized that this should be the easiest anomaly out of these categories to detect, and could likely be easily done by using classical, non-ML approaches as well. This situation would likely be the most typical in video injection attacks, as the phone is likely accessed via a computer interface. A total of 6 sequences were collected.

- Phone is kept on hand, while a picture or video of a face is displayed to the camera from another phone and adjusted until auto capture is activated. This simulates a lower skill/effort level spoofing attempt, which in all likelihood should fail anyway, as the verification engine is very sensitive to reflections and uses liveness checking methods based on biosignals extracted from the selfie photo. In part of the samples, the selfie camera shutter button was also used to capture the photo. 11 sequences were collected from 2 subjects.

(a) Phone is sitting still on the table **(b)** Phone is kept on hand

Figure 6: Two types of attacks physically mimicked

In addition, 18 unseen normal sequences from 3 subjects with real selfie capture moment shifted a random value between 1.5 and 3 seconds were included, to simulate possible real-time morphing attacks using filters.

3.1.4 Final Data Set

The final data set contains data from 30 subjects, with 10-15 selfie capture sequences from each, adding up to a total of 375 sequences. The included data channels were limited to 3-axis accelerometer, gyroscope, (raw) magnetometer and linear acceleration, spanning a total of 12 channels. To keep both the camera opening moment features and the window around selfie capture, while keeping the sequences uniform length without padding, the final data set was divided into two separate parts of equal length.

The first part contains the first 200 samples after camera opening, spanning a total of 4 seconds. For most of the sequences, this also eclipses the selfie capture moment, as well as a good part of the wait for a result from the server. The second part contains the discussed window around the selfie capture moment, 50 samples before and 150 samples after. Both parts span a total of 200 samples of partly overlapping time series data. As both data sets are of equal length, they can be easily input in parallel to multivariate TSC algorithms, or alternatively cut and concatenated into a "single" multivariate time series with possible discontinuities, to avoid repetition. Both parts of the data set were packed into .ts files with identical format to the data sets in the UCR time series archive, and are publicly available. Additionally, the original folder structure containing the whole sequences and all raw sensor logs is also provided. These can be found in https://github.com/Ergzar/Selfie-Motion-AD-TSC.

3.2 Benchmarking

State-of-the-art analysis methods were applied to the collected dataset in a benchmark comparing multiple approaches. This subsection first outlines the computational environment and then introduces the anomaly detection techniques and time series classification algorithms used in each of the three benchmarks.

3.2.1 Environment Setup

The benchmarks were implemented in three distinct Python-based custom scripts. The classification algorithm implementations were imported from the aeon TSC toolkit, while the anomaly detection algorithms were gathered from various sources including aeon, listed in Table 1.

Multiple key parameters could be tweaked in all 3 benchmark scripts:

- The amount of data points to be included before and after selfie capture moment.
- Window option: after camera opening, around selfie capture or both, doubling the amount of total channels.
- Sensor channels: ability to cherry-pick individual axes of different sensors.

- Custom low-pass Butterworth filter: apply on all included channels or none.
- The amount of resamples/repeats + number of inner/outer folds.
- Ability to swap the classifier or method used.

The scripts are also available in https://github.com/Ergzar/Selfie-Motion-AD-TSC. Preliminary tuning was used to identify approximately optimal parameters, after which the benchmark evaluated and reported a range of configurations based on them. Exploring every possible parameter configuration was infeasible, so a focused subset was selected. Choices were guided by prior literature, preliminary tuning, and computational budget, with the final lineup designed to balance coverage and run-time.

3.2.2 Anomaly Detection Techniques

The anomaly detection methods in this benchmark were chosen based on (i) the availability of whole-series implementations and (ii) their prevalence in prior work. These include the current WTSAD state-of-the-art, ROCKAD (Theissler et al., 2023), Isolation Forest, a widely used tree-based outlier method (F. T. Liu et al., 2008) and a one-class SVM (OCSVM) with an RBF kernel (Schölkopf et al., 2001; Tax & Duin, 2004), which has also been applied in mobile behavioral biometrics (Sitová et al., 2015). Further distance-based baselines are also included: Euclidean k-NN on z-normalized raw series, k-NN with Dynamic Time Warping (DTW) on z-normalized series, Euclidean k-NN in QUANT's feature space, and an Isolation Forest trained on QUANT features. Finally, a lightweight long short-term memory (LSTM) autoencoder as introduced by Malhotra et al. (2016) to one-class learning and anomaly detection purposes was included to represent deep-learning-based approaches. The model is a sequence-to-sequence architecture with an input mask, a stacked LSTM encoder, a linear bottleneck of size latent, and a symmetric LSTM decoder with a time-distributed linear output layer. Training minimizes mean squared error (MSE) with Adam. At inference, the anomaly score is the mean squared reconstruction error averaged over time steps and channels. The benchmarked anomaly-detection methods, their categories, and their implementation sources are listed in Table 1.

The best window configuration found in preliminary testing was a window of 50 samples before and 100 samples after the selfie capture timestamp, with 10 first samples recorded after camera opening concatenated in front. Despite introducing a discontinuity to the time series, including this operation on the sequences improved the performance significantly. The window could have theoretically been optimized even further for better results, but this was not done because the main aim of these benchmarks was to compare the performances between different techniques, not optimizing hyperparameters. The channel configuration was varied between including 9 channels (3-axis accelerometer, 3-axis gyro, 3-axis magnetometer with unit vector differencing applied), 3 channels (one with only the x-axis of all aforementioned sensors, one with x, y and z axes of just the accelerometer) and just one channel (only the x-axis of accelerometer).

Table 1: Detectors used in anomaly detection experiments, their category, and implementation origin.

Name	Category	Origin of implementation
Isolation Forest (QUANT features)	Tree ensemble	QUANT featurizer from `aeon` + IF from `sklearn`
Isolation Forest (raw)	Tree ensemble	`scikit-learn`
k-NN DTW (z-normalized)	Distance-based	`aeon` + custom k-NN
k-NN (QUANT features)	Distance-based	QUANT featurizer from `aeon` + custom k-NN
Euclidean k-NN (z-normalized)	Distance-based	Custom implementation
LSTM autoencoder	Deep autoencoder (reconstruction error)	Custom (Keras) implementation
OCSVM (RBF)	Support vector machine	`scikit-learn`
ROCKAD	Distance-based (power-transformed ROCKET features)	ROCKAD repository (https://github.com/ml-and-vis/ROCKAD)

In the 10-shot one-class learning experiment, the time series window was set up similarly with 10 first samples concatenated in front, but a with a small tweak: instead of 50 samples before and 150 after the selfie capture timestamp, 35 were included before and 165 after. The experiment was conducted using per-user thresholds. The 2 subjects with fewer than 12 sequences in the data set were not included in this benchmark, leaving a total of 28 users.

In both experiments, as a preprocessing step, a 12.5 Hz low-pass Butterworth filter was applied on the time series in all sensor configurations to smooth out noise and to prevent the detectors from fitting into irrelevant details.

3.2.3 User Classification Algorithms

User-classification methods were selected primarily on the basis of their performance in the benchmark of Middlehurst et al. (2024b), while ensuring representation of distinct algorithm categories. All implementations were obtained from the aeon toolkit (Middlehurst et al., 2024a). The overall state-of-the-art (SOTA) method, HIVE-COTE 2.0, was excluded due to excessive training times. However, the close second MR-Hydra (Dempster et al., 2023) was included. The dictionary-based SOTA WEASEL V2 is not compatible with multichannel data, but its predecessor, WEASEL, had a separate multivariate implementation (WEASEL+MUSE, introduced by Schäfer and Leser (2018)) available in aeon's classification library, and therefore this implementation was used. Two interval-based approaches were included: QUANT (Dempster et al., 2024) and RSTSF (Cabello et al., 2024). The shapelet-based SOTA algorithm RDST (Guillaume et al., 2021) was also included. A deep-learning baseline was provided by ResNet (Wang et al., 2017), which, while not the current SOTA in its category, remained competitive in the recent bake-off. The algorithms included in the user-classification benchmark, together with their categories and default classifiers, are summarized in Table 2.

Table 2: Models, their category, and their underlying default classifier.

Model	Category	Default classifier
MR-Hydra	Hybrid (convolution + dictionary)	RidgeClassifierCV
WEASEL+MUSE	Dictionary-based	Logistic regression
QUANT	Interval-based	ExtraTreesClassifier
RSTSF	Interval-based	DecisionTreeClassifier
RDST	Shapelet-based	RidgeClassifierCV
ResNet	Deep Learning	– (end-to-end NN)
catch22	Feature-based	RandomForestClassifier

Most of the algorithms were run on their default configurations, with the exception of ResNet, which was run with a batch size of 32 and 150 epochs. The default 1500 epochs took too long to train, and after 150 epochs there seemed to be diminishing returns in training/validation loss.

The same 12.5 Hz low-pass Butterworth filter used in the anomaly detection benchmarks was applied to the sequences in this benchmark as well. The benchmark was run on 4 distinct sensor configurations listed below:

- X-axis of accelerometer (univariate).
- All 3 accelerometer axes (multivariate).
- Only the x-axis of accelerometer, gyroscope and magnetometer (multivariate).
- X, y and z axes of accelerometer, gyroscope and processed magnetometer (9 distinct channels, multivariate).

The benchmark's univariate part was run for three distinct time series windowing configurations: one centered around the selfie (50 samples before, 150 after), another one stacking the 50-150 window around the selfie capture with a window of same length (200) starting from the moment the camera opens, effectively doubling the channel count. The third configuration concatenates 50 samples after camera opening in front of the selfie-centered window. The more heavyweight, multivariate sensor channel configurations were run only on the best-performing window configuration. Additionally, the best-performing algorithm was combined with the following sensor channel configurations to compare differences between distinct sensors:

- All 3 accelerometer axes (multivariate).
- All 3 gyroscope axes (multivariate).

- All 3 linear acceleration axes (multivariate).
- All 3 processed magnetometer axes (multivariate).
- All 3 raw magnetometer axes (multivariate).

3.3 Training and evaluation

The chosen analysis methods were applied to the collected dataset. This section defines the evaluation metrics and details the training configuration for each of the three benchmarks, including cross-validation protocols and decision threshold calibration.

3.3.1 Evaluation Metrics

The primary evaluation metric used by Theissler et al. (2023) in WTSAD was the Area Under the Receiver Operating Characteristic Curve (AUROC). To compute it, all test series are first scored by the chosen TSAD algorithm. Next, every possible pair consisting of one normal and one anomalous series is formed. AUROC can be interpreted as the probability that the anomalous member of such a pair receives the higher score (Bradley, 1997).

Example: If the test set contains $N_{\text{norm}} = 30$ normal series and $N_{\text{anom}} = 5$ anomalous series, the number of pairs is $30 \times 5 = 150$. Suppose the anomalous score exceeds the normal score in 135 of those pairs; the AUROC can then be calculated as:

$$\text{AUROC} = \frac{135}{150} = 0.90.$$

Although AUROC provides a good estimate of the overall performance of an anomaly detector, the application of this thesis is in user verification, where real users are directly affected by false rejections. For this reason, it was concluded that False Rejection Rate (FRR) for genuine users and False Acceptance Rate (FAR) for spoof attempts at a fixed operating point are the most suitable primary metrics for this experiment.

The most widely used metric in biometrics and user verification applications is the Equal Error Rate (EER) (Y. Zhang et al., 2023a). This is the point on the ROC curve where the false acceptance rate (FAR) equals the false rejection rate (FRR = 1 - TPR). This was also the main metric used for performance evaluation by Sitová et al. (2015) discussed earlier. This particular value is important for balancing the risk of an impostor being successful in their spoofing attempt with the risk of a real user being falsely rejected, without having an arbitrary bias in either direction.

As mentioned in previous chapters, motion patterns would be only a supplementary extra feature being examined in deployment. To avoid inconvenience to users, the false rejection rate is to be minimized. An FAR of 50% with an FRR of <1% (preferably 0%) would mean that the motion analysis check would successfully detect half of spoofing attempts while causing minimal inconvenience to users. The ability to detect even half of potential spoofing attempts would still be a huge advantage in the industry. Thus, setting thresholds favoring low FRR are prioritized in this thesis.

3.3.2 Training Configuration in Anomaly Detection Experiments

As the benchmarked anomaly detection methods were semi-supervised, operating thresholds were calibrated from **genuine** samples only (the negative class), following common practice in keystroke dynamics (Dimaratos & Pöhn, 2023; Sae-Bae & Memon, 2022). Sequences from each user were treated as groups: group-aware splitting ensured that all sequences from a given user appeared either in the outer-test fold *or* in the outer-train fold, never both. Also within the outer-train fold, same group-aware splitting was employed between inner-train and calibration.

Out-of-fold (OOF) calibration on the training set was employed: anomaly/score values from inner-calibration folds were aggregated, and a decision threshold τ was set to the high quantile required to meet the target FRR (99^{th} percentile for FRR $\approx 1\%$). Each genuine sequence contributed once via OOF, and the resulting τ was then fixed and applied to the disjoint test set containing the emulated spoofing behavior (see Section 3.1.3). Train–test splits were reshuffled five times with different seeds to mitigate randomness (Monte Carlo, no k-fold cross-validation). Figure 7 gives a visual overview of the process.

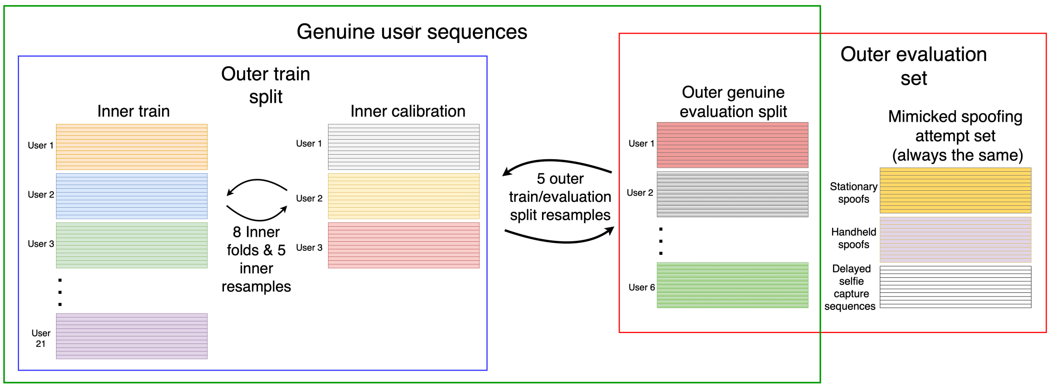


Figure 7: Data distribution in the "user vs spoof" anomaly detection experiment.

Additionally, a few-shot, one-class verification experiment was conducted in which genuine samples were used to set per-user decision thresholds. The design mirrored the anomaly-detection setup, except that each user contributed exactly 10 genuine sequences for outer-train set with 2–5 sequences held out for outer-test set. The same anomaly detection techniques were employed (with the addition of 1-NN DTW), but emulated spoofs were omitted from the data set, performance was evaluated only against *zero-effort* impostors, and thresholds were calibrated per user rather than globally, consistent with operational biometrics (Jain et al., 2004). Two equivalent inner cross-validation configurations were employed, namely a 5 fold scheme with five resamples and a 2-fold scheme with five resamples, both yielding 50 out-of-fold genuine scores per user. From these scores a per-user decision threshold was estimated and then applied in the outer folds to held-out genuine samples and *zero-effort* impostor trials to compute FRR and FAR. Final FRR and FAR figures were obtained by averaging across outer folds and across users. Figure 8 presents a block diagram of such approach that uses a k-NN classifier.

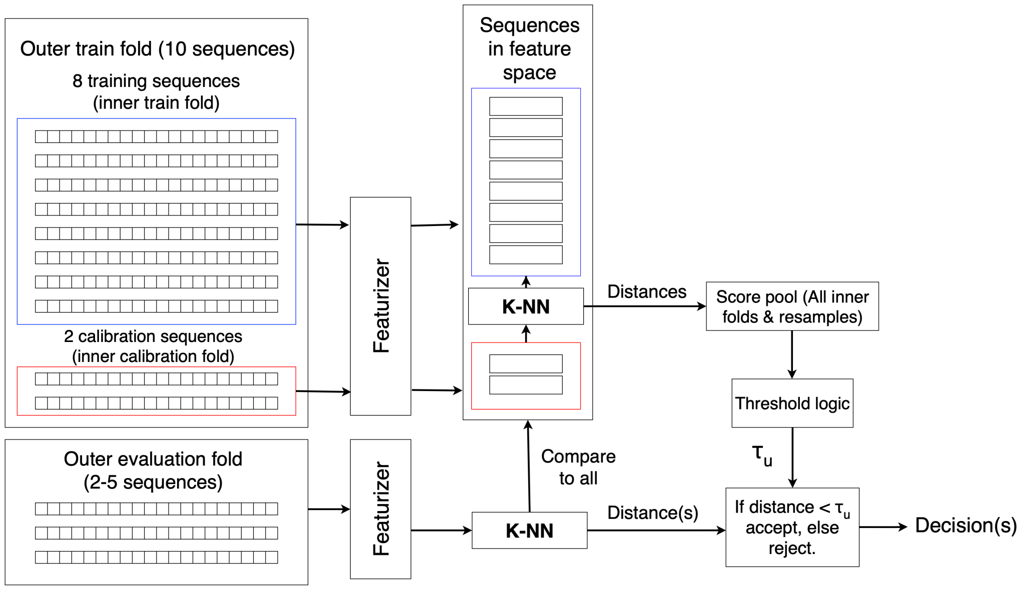


Figure 8: Flowchart of k-NN based verifiers in the 10-shot one-class learning experiment.

3.3.3 Training Configuration in User Classification Experiment

Each time-series classifier was evaluated in a multiclass setting over all users using nested cross-validation. A 5-fold stratified outer CV was run. For each outer fold, the classifier was trained on the full outer-train split and evaluated on the outer-test split. Before training on the full outer-train split, an inner stratified CV on that same outer-train split was used to calibrate a single global threshold τ targeting FRR = 1% (`FRR_TARGET`=0.01). The inner CV used up to 3 folds and 5 repeats (five repeats were sufficient given the small number of sequences per class). Within each inner fold, the model was fitted on the inner-train portion and out-of-fold genuine scores (the predicted probability of the true user/class) were collected on inner validation. These OOF genuine scores were pooled across users and repeats, and τ was set to the quantile of this pool that achieved the target FRR. Impostor scores were not used for calibration. Optionally, per-user thresholds τ_u could also be computed, but unless stated otherwise results are reported with the global τ for stability.

At test time (outer-test split), each classifier produced a probability vector over users for each sequence. For a claimed user u, the score was the probability assigned to u. Scores were partitioned into genuine (sequences of u) and impostor (all others), and FRR_{test} and FAR_{test} were computed at the train-calibrated τ. Metrics from each outer fold were macro-averaged to produce the final reported figures.

Independently of the operating point, EER was computed by sweeping a threshold over each user's test scores within each fold and taking the equal-error point (linear interpolation at the first crossing). Per-user EERs were macro-averaged within the fold and then averaged across folds.

A single multiclass classifier over all users was adopted (as preferred by Estrela et al. (2021)) rather than per-user binary models, to reduce overfitting and to maintain a consistent scoring interface for both operating-point metrics and EER.

4 Results

This chapter presents the results obtained by applying the methods to the research materials described in Chapter 3. Section 4.1 provides visualizations and an overview of key observations on the collected dataset. Section 4.2 reports the results of the anomaly detection -based benchmarks, and Section 4.3 for the time series classification benchmark.

4.1 Overview of Typical Gesture Patterns

The z-axis accelerometer is an informative data channel when visualizing the broad movement patterns during selfie capture sequences, as the movements typically mainly consist of raising and lowering, or tilting the device toward/away from the face. Figure 9 shows how a general pattern can be observed across all 12 verification sessions in the z-axis accelerometer readings of a typical subject. As can be seen, the graphs contain some jitter, indicating a slight shaking or instability of the hand. Another interesting phenomenon that can be observed is how the device notably stabilizes just before the selfie is captured, and after that there's a delay of around 1 second before notable movement or orientation changes can be detected again. The biggest variance in the orientation seems to happen a few seconds after the selfie is captured, as the subject is waiting for the verification result from the server.

Figure 10 shows the z-axis accelerometer patterns of another subject. These look very different compared to the first subject, as there seems to be no notable orientation change after the selfie capture, only a slight and delayed trend of changing the orientation to a more horizontal position, as well as slightly more hand instability. Also, a platform anomaly can be spotted in one of the samples, as the subject placed the phone on the table after the selfie was captured: the purple graph shows a clear impact at x = 300 and a flat, almost completely stable line afterwards. The almost complete absence of any kind of disturbances in the graph also demonstrates how little sensor noise there seems to be.

Figure 11 presents the z-axis accelerometer patterns of a third subject. There seems to be notable variability between the sequences, no clear one pattern dominates. Some pattern-shift anomalies can be seen, as reaction time between selfie capture and lowering of the phone back down varies between 1 and 4 whole seconds (50 and 200 data points). In one of the sequences, the orientation of the phone barely changes throughout the whole sequence (see the lime-colored graph).

Despite the interesting differences between users and individual outlier anomalies, general trends in orientation and movement patterns can be detected in multiple sensor axes, as Figure 12 plots the mean values of each accelerometer and gyro axis readings from all 375 time series sequences covering all 30 subjects in a window of 2 seconds before and 6 seconds after the selfie capture. The dark purple/green areas cover 25-75% deviations from the median, and the lighter purple/green reas the 10-90% deviations. covers one standard deviation in the sequences from the mean. The same figures in larger size can be found in Appendix B.

Despite the interesting differences between users and occasional outlier anomalies,

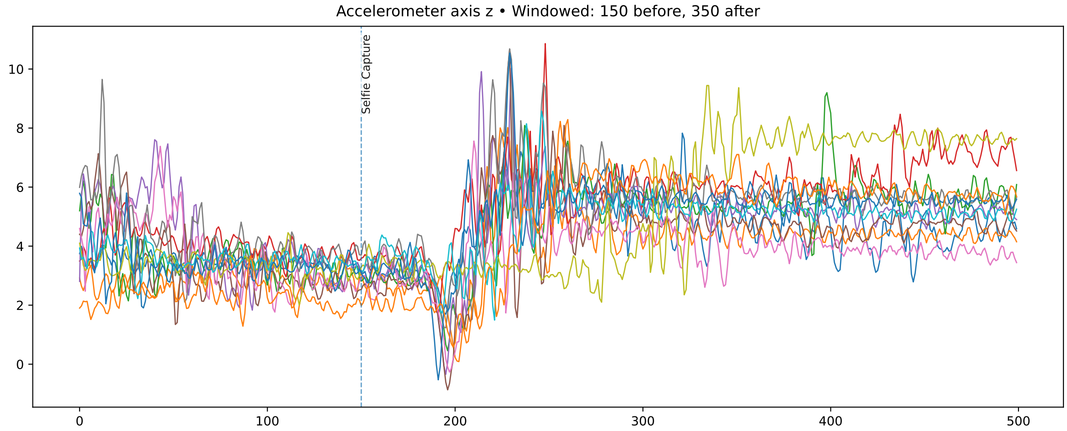


Figure 9: All 12 z-axis accelerometer time series of one subject aligned around selfie capture (at x=150 samples).

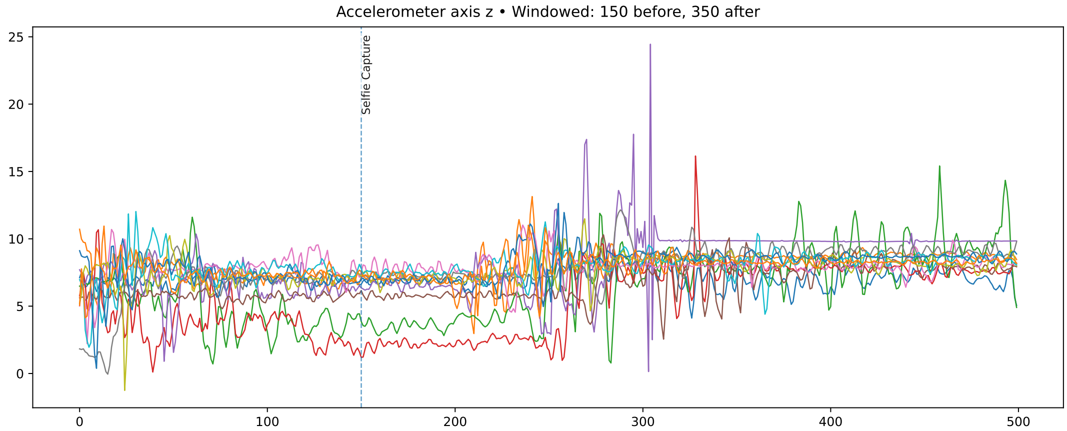


Figure 10: Plots of z-axis accelerometer time series of a second subject. Notice the purple plot.

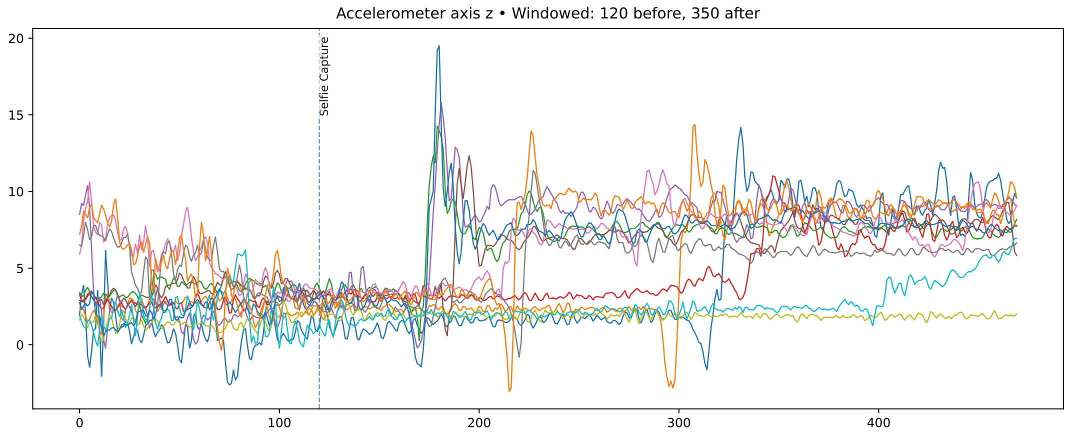


Figure 11: Plots of z-axis accelerometer time series of a third subject. Notice the variability in reaction times.

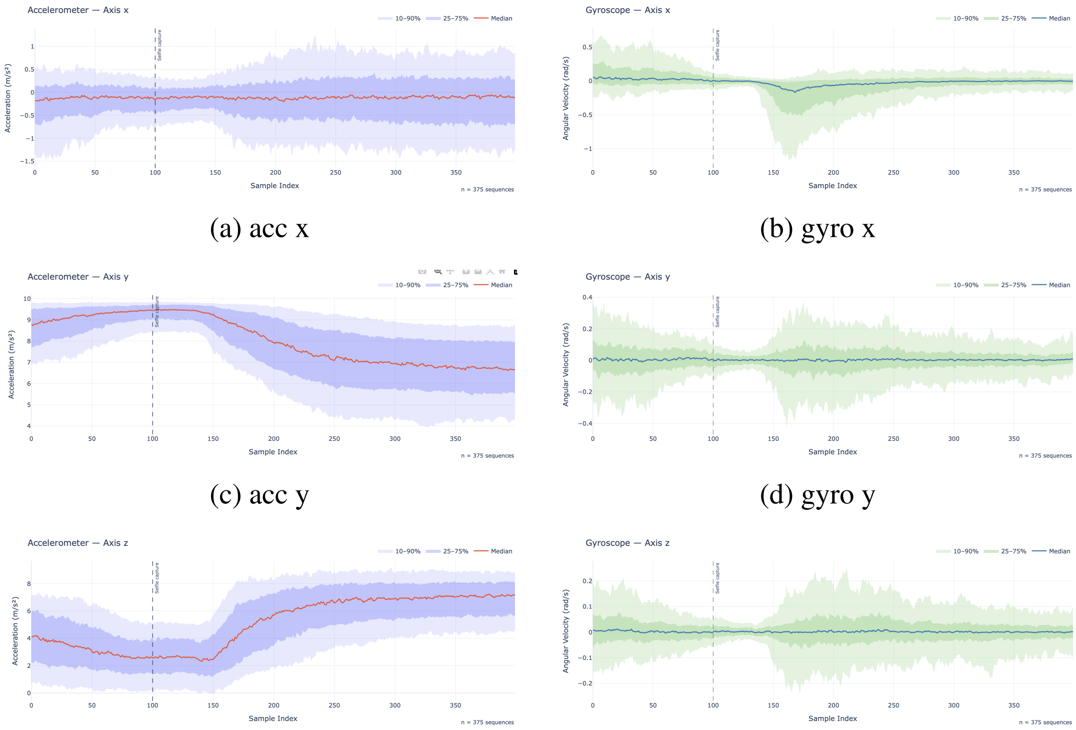
consistent orientation and movement patterns are visible in the sensor axes. Figure 12 shows the median time series for each accelerometer and gyroscope axis, each aligned to the selfie capture moment, which is marked by the vertical dashed line. The shaded regions describe variability: the darker shading covers the middle 50% of values (25th to 75th percentiles) and the lighter shading covers the middle 80% of values (10th to 90th percentiles). The horizontal axis spans two seconds before the capture and six seconds after it. Full-size versions of these plots are provided in Appendix B.

A general trend in the z-axis orientation can be noticed in Figure 12e, where it can clearly be observed how the phone is adjusted towards a more vertical orientation leading up to the moment of selfie capture (gravity offset decreases), and how the phone is kept stable for around 1 second after the selfie, and then tilted back towards a more horizontal position while waiting for the result from the server.

Similarly, in the the gyroscope readings, there's a clear-corresponding pattern to be recognized. As the phone is lowered back down, there's noticeable negative rotation around the x-axis shown in Figure 12b, matching well with gestures associated with lowering the phone back down. Another point of interest in the gyro readings is the very low variability just after the selfie is captured, indicating very little movement. This is likely due to them focusing on the "selfie capture success"-animation playing on the screen and relaxing once the "Analyzing" view pops up.

The x-axis of accelerometer and z-axis of gyroscope however, have very small variance. This direction of movement would be along the x-axis of the the device. The mean of the x-axis accelerometer readings hovers just below zero, indicating that the phone is generally kept fairly straight, with no strong tilt in either direction, except a slight preference towards the left. This could be due to most subjects being right handed, or just a coincidence. The z-axis gyro readings are almost perfectly zero-mean, following the same pattern of a period of stillness right after selfie capture, followed by a noisier period after the "selfie capture success" animation is completed.

Overall, it can be seen that after capturing the selfie, the users' focus relaxes, and their behavior becomes more chaotic and prone to random effects while waiting for the verification result. This observation supports the decision to limit the time series window so as not to include the waiting period in its entirety.



(a) acc x

(b) gyro x

(c) acc y

(d) gyro y

(e) acc z

(f) gyro z

Figure 12: Percentile bands around capture for accelerometer (left column) and gyroscope (right column) along axes x, y, and z. The dashed vertical line marks the selfie-capture moment. The same figures are displayed in a bigger size in the appendix.

4.2 Anomaly Detection

This section presents the results of the two benchmarks that compare anomaly detection techniques.

4.2.1 10-Shot One-Class User Verification

The 10-shot one-class learning user verification benchmark was run using the methods specified in Section 3.2.2. The time series window used in this experiment was 10+35+165 (10 first samples after opening the camera concatenated with 35 samples before and 165 samples after selfie capture moment). Table 3 presents results for all 9 sensor channels when using 5 inner folds:

Table 3: Per-user 10-shot motion based verification (selfie-centered; macro FAR/FRR at preset τ_u from OOF calibration; values in %; 5 inner-fold splits; 9 channels).

Method	FRR (%)	FAR (%)
ROCKAD	1.74	89.74
Isolation Forest (raw series)	4.01	64.00
OCSVM (raw series, RBF)	8.64	54.70
Euclidean 1-NN	6.40	60.94
1-NN DTW	6.40	60.94
QUANT + 1-NN classifier	4.83	59.29
QUANT + Isolation Forest	5.92	63.10

As can be seen from the results, the only method while using all 9 sensor channels and 5 inner fold splits with an FRR even approaching tolerable (1.74%) was ROCKAD. However it also yielded a FAR of 89%, meaning that roughly 9 in 10 zero-effort impostors would be accepted. It should also be noted that the ROCKAD algorithm's power transform function occasionally displayed overflow warnings during this benchmark, which may have had an effect on the results. Table 4 displays the outcome for the same 9 sensor channels when using only two inner folds instead of the previous five:

Table 4: Per-user few-shot motion-based verification (selfie-centered; macro FAR/FRR at preset τ_u from OOF calibration; values in %; 2 inner-fold splits; 9 channels).

Method	FRR (%)	FAR (%)
ROCKAD	0.00	99.70
Isolation Forest (raw series)	25.50	26.01
OCSVM (raw series, RBF)	10.32	50.34
Euclidean 1-NN	3.85	72.17
1-NN DTW	3.93	72.38
QUANT + Isolation Forest	28.45	26.07
QUANT + 1-NN	3.64	69.50

Switching from five to two inner folds lowered FRR but raised FAR for the 1-NN methods (including ROCKAD), likely due to a larger calibration share per fold providing more calibration scores and a more accurate threshold at the 1%-FRR target. Two inner folds is therefore the standard in subsequent experiments. The IsolationForest methods showed the reverse pattern and were dropped. Results for the remaining approaches on accelerometer x-axis data appear in Table 5.

Table 5: Per-user few-shot motion-based verification (selfie-centered; macro FAR/FRR at preset τ_u from OOF calibration; values in %; 2 inner-fold splits; x-accelerometer).

Method	FRR (%)	FAR (%)
ROCKAD	0.00	99.67
OCSVM (raw series, RBF)	11.82	53.27
Euclidean 1-NN	2.59	74.71
1-NN DTW	2.15	72.32
QUANT + 1-NN	4.61	83.03

In Table 6, the performance metrics are summarized across 6 single-sensor configurations for euclidean 1-NN, DTW 1-NN and QUANT + 1-NN. ROCKAD was excluded from the comparison because of persistently inferior performance (FAR approaching 100%) and a significantly higher computational cost.

Table 6: Per-user 10-shot anomaly detection (selfie-centered) for each sensor axis (rows). Macro FRR/FAR at preset τ_u from OOF calibration (2 inner-folds). Values in %.

Sensor	1-NN DTW		Euclidean 1-NN		QUANT + 1-NN	
	FRR (%)	FAR (%)	FRR (%)	FAR (%)	FRR (%)	FAR (%)
x-acc	2.15	72.32	2.59	74.71	4.61	83.03
y-acc	4.25	86.12	3.68	85.88	2.81	79.82
z-acc	5.04	85.82	5.36	85.74	2.75	70.87
x-gyro	3.36	82.97	4.15	85.37	3.98	82.18
y-gyro	2.02	84.70	1.76	82.03	4.36	83.07
z-gyro	1.47	87.85	1.45	88.88	5.06	87.69

The lowest FRR values are obtained with the gyroscope's y- and z-axis signals using 1-NN (Euclidean or DTW), although at the cost of a very high FAR (>80%). Notably, there seems to be an inverse correlation between the performances: where QUANT-1-NN underperforms, Euclidean/DTW 1-NN improves, and vice versa. Next, a multi-channel configuration using all 3 accelerometer axes is presented in Table 7.

Table 7: Per-user few-shot motion-based verification (selfie-centered; macro FAR/FRR at preset τ_u from OOF calibration; values in %; 2 inner-fold splits; all 3 axes of accelerometer).

Method	FRR (%)	FAR (%)
OCSVM (raw series, RBF)	8.72	54.23
Euclidean 1-NN	4.06	79.95
1-NN DTW	4.77	79.14
QUANT + 1-NN	3.64	69.83

Using all three accelerometer axes is outperformed by several single-axis configurations for all 1-NN variants. For instance, 1-NN DTW on the accelerometer x-axis yields an FRR of 2.15% with a FAR of 72.32%, compared with 4.77% and 79.14% respectively, when all three axes are used. Next, Table 8 reports results for an approach where the x-axes of the each sensor (accelerometer, gyroscope, and magnetometer) are combined as a multivariate input:

Table 8: Per-user few-shot motion-based verification (selfie-centered; macro FAR/FRR at preset τ_u from OOF calibration; values in %; 2 inner-fold splits, x-acc, x-gyro, x-magnet).

Method	FRR (%)	FAR (%)
OCSVM (raw series, RBF)	11.87	56.58
Euclidean 1-NN	1.72	73.03
1-NN DTW	2.23	69.63
QUANT + 1-NN	4.38	79.52

Euclidean 1-NN + x-axis readings of each of the 3 sensors yielded the overall best result out of any of the combinations so far with both a sub-2% FRR and sub-75% FAR. This configuration appeared to be the most promising, so a parameter ablation was performed. The window was adjusted to 10 samples after camera opening, 50 before selfie capture, and 150 after, and the classifier was changed from 1-NN to 3-NN (using the k-th nearest neighbor rule). With this setup, Euclidean 3-NN achieved FRR=0% at the cost of FAR=79.70%, while 3-NN DTW reduced FRR to <1% with FAR of 74.77% (Table 9).

Table 9: Per-user few-shot motion-based verification (selfie-centered; macro FAR/FRR at preset τ_u from OOF calibration; values in % (2 inner-fold splits; x-acc, x-gyro, x-magnet; 10+50+150 window).

Method	FRR (%)	FAR (%)
OCSVM (raw series, RBF)	13.04	58.64
Euclidean 3-NN	0.00	79.94

Method	FRR (%)	FAR (%)
3-NN DTW	0.87	74.77
QUANT + 3-NN	3.19	79.98

This evaluation still reflects zero-effort impostor attempts. An adversary performing a real-time morphing attack would likely move in an odd manner combined with a delay, yielding anomaly scores higher than those observed here, making them easier to reject. It was also observed that users with highly uniform motion patterns across all of their sequences tended to have an elevated FRR. This can be explained by the decision threshold being set to a low value due to low distance scores across the board in the train set, but when more anomalous unseen sequences appeared from that same user in the test set, they would be rejected. This effect was amplified in the data set collected for this thesis, because all sequences were collected within a single session, reducing natural variance. Mitigation is possible by adding a relative safety margin to the threshold, but this inevitably increases FAR even further.

4.2.2 Genuine-vs-Spoof Anomaly Detection

A similar experiment was conducted in which all 30 users in the dataset were included in the genuine class. Results for the accelerometer-only configuration (x, y, and z axes) are reported in Table 10 and Table 11. The results of the configuration combining the accelerometer (x-axis), gyroscope (x-axis), and magnetometer (x-axis) are reported Table 12 and Table 13. Finally, the full nine-channel configuration combining accelerometer, gyroscope, and magnetometer, each using the x, y, and z axes, is summarized in Table 14 and Table 15.

Table 10: Approach-wise verification results (All 3 accelerometer axes, 10+50+100 concatenated selfie window). Average FAR is an equal-weight mean over the three spoof types (stationary, handheld, delayed). Values are mean ± std across resamples.

Model	FRR (%)	FAR overall (%)	Verif. delay (ms)
ROCKAD	0.00 ± 0.00	43.80 ± 0.83	3.4846 ± 0.1072
QUANT-kNN	2.37 ± 1.46	31.99 ± 0.00	0.4647 ± 0.0065
QUANT-IF	0.00 ± 0.00	91.41 ± 0.00	0.5192 ± 0.0107
IF (raw)	4.81 ± 8.04	35.62 ± 2.54	0.1278 ± 0.0049
OCSVM	0.81 ± 1.81	43.16 ± 3.05	0.0040 ± 0.0012
Euclidean 3-NN	0.811 ± 1.81	72.22 ± 0.00	0.0256 ± 0.0007
LSTM-AE	1.33 ± 1.66	41.38 ± 0.83	0.2699 ± 0.0225

Table 11: Approach-wise FAR (%) by spoof type (All 3 accelerometer axes, 10+50+100 concatenated selfie window). Values are mean ± std across resamples.

Model	FAR (stat.)	FAR (handheld)	FAR (delayed)
ROCKAD	0.00 ± 0.00	63.64 ± 0.00	67.78 ± 2.48
QUANT 3-NN	0.00 ± 0.00	18.18 ± 0.00	77.78 ± 0.00
QUANT-IF	100.00 ± 0.00	90.91 ± 0.00	83.33 ± 0.00
IF (raw)	0.00 ± 0.00	29.09 ± 7.61	77.78 ± 0.00
OCSVM	0.00 ± 0.00	47.27 ± 4.07	82.22 ± 6.09
Euclidean 3-NN	16.67 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
LSTM-AE	0.00 ± 0.00	36.36 ± 0.00	87.78 ± 2.48

Table 12: Approach-wise verification results (x-acc, x-gyro, x-magnet, 10+50+100 concatenated selfie window). Values are mean ± std across resamples.

Model	FRR (%)	FAR overall (%)	Verif. delay (ms)
ROCKAD	1.83 ± 1.93	58.08 ± 1.31	3.4567 ± 0.0483
IF	0.51 ± 1.15	74.41 ± 0.00	0.1234 ± 0.0059
OCSVM	0.53 ± 0.72	82.39 ± 1.36	0.0114 ± 0.0017
LSTM-AE	1.05 ± 1.08	88.45 ± 1.57	0.2664 ± 0.0341
QUANT IF	0.26 ± 0.58	95.12 ± 0.00	0.5222 ± 0.0156
Euclidean 3-NN	0.27 ± 0.60	95.56 ± 2.81	0.0265 ± 0.0015
QUANT 3-NN	0.78 ± 1.15	85.05 ± 2.55	0.4882 ± 0.0266

Table 13: Approach-wise FAR (%) by spoof type (x-acc, x-gyro, x-magnet, 10+50+100 concatenated selfie window). Values are mean ± std across resamples.

Model	FAR (stat.)	FAR (handheld)	FAR (delayed 1.5–3)
ROCKAD	0.00 ± 0.00	90.91 ± 0.00	83.33 ± 3.93
IF	100.00 ± 0.00	45.45 ± 0.00	77.78 ± 0.00
OCSVM	100.00 ± 0.00	52.73 ± 4.07	94.44 ± 0.00
LSTM-AE	100.00 ± 0.00	70.91 ± 4.07	94.44 ± 5.56
QUANT IF	100.00 ± 0.00	90.91 ± 0.00	94.44 ± 0.00
Euclidean 3-NN	96.67 ± 7.45	100.00 ± 0.00	90.00 ± 2.48
QUANT 3-NN	100.00 ± 0.00	61.82 ± 4.07	93.33 ± 4.65

Table 14: Approach-wise verification results (9 channels (3 acc, 3 gyro, 3 mag), 10+50+100 concatenated selfie window). Values are mean ± std across resamples.

Model	FRR (%)	FAR overall (%)	Verif. delay (ms)
ROCKAD	1.33 ± 1.66	46.40 ± 1.66	3.7796 ± 0.2997
IF	2.15 ± 4.12	41.08 ± 0.00	0.1213 ± 0.0023
OCSVM	1.35 ± 3.02	41.25 ± 0.81	0.0105 ± 0.0017
LSTM-AE	1.61 ± 2.93	41.01 ± 1.66	0.2430 ± 0.0078
QUANT IF	0.00 ± 0.00	93.27 ± 0.00	0.9436 ± 0.0902
Euclidean 3-NN	0.80 ± 1.20	92.02 ± 5.07	0.0503 ± 0.0036
QUANT 3-NN	2.38 ± 1.78	31.99 ± 0.00	1.0290 ± 0.0234

Table 15: Approach-wise FAR (%) by spoof type (9 channels (3 acc, 3 gyro, 3 mag), 10+50+100 concatenated selfie window). Values are mean ± std across resamples.

Model	FAR (stat.)	FAR (handheld)	FAR (delayed 1.5–3)
ROCKAD	0.00 ± 0.00	63.64 ± 0.00	75.56 ± 4.97
IF	0.00 ± 0.00	45.45 ± 0.00	77.78 ± 0.00
OCSVM	0.00 ± 0.00	38.18 ± 4.07	85.56 ± 4.97
LSTM-AE	0.00 ± 0.00	36.36 ± 0.00	86.67 ± 4.97
QUANT IF	100.00 ± 0.00	90.91 ± 0.00	88.89 ± 0.00
Euclidean 3-NN	83.33 ± 11.79	92.73 ± 4.07	100.00 ± 0.00
QUANT 3-NN	0.00 ± 0.00	18.18 ± 0.00	77.78 ± 0.00

Across configurations, the best results were obtained with the accelerometer's x, y, and z axes. In this setting, the state-of-the-art WTSAD algorithm ROCKAD achieved an FRR of 0% (over five resamples) with FAR = 43.8%. QUANT + 3-NN also performed well, achieving the lowest FAR among all methods (31.99%) at the cost of a slightly higher FRR (2.37%).

Both approaches perfectly rejected static spoofs (device held completely still). For handheld spoofs (camera pointed at a face on another screen) and delayed spoofs (the capture timestamp shifted by 1.5-3.0 s to emulate live morphing/video-injection lag), ROCKAD rejected roughly one third in each category. By contrast, QUANT + 3-NN was particularly effective on handheld spoofs, reaching FAR = 18.18% (about four out of five correctly rejected).

The LSTM autoencoder delivered solid accuracy (third-best overall) but had by far the longest training time. The Euclidean 3-NN on z-scored time series, which worked best in the 10-shot semi-supervised user-verification experiment, performed poorly across these sensor configurations. IsolationForest-based variants also performed poorly, and OCSVM was consistently average without standing out on any metric.

In terms of runtime, ROCKAD had the longest verification latency (in the 3-channel setups, over 7× the next fastest method), yet the absolute time was only 3–4 ms, which is negligible in practice and unlikely to introduce any user-perceptible delay.

The sensor configuration using only the x-axis from each sensor (accelerometer, gyroscope, and magnetometer) performed by far the poorest of the three. Overall FAR exceeded 80% for 5 of the 7 models/methods. Notably, the FAR on the stationary spoof samples reached 100% (or nearly 100%) for 6 of the 7 models.

4.3 User Classification

The TSC user-classification benchmark was carried out using the algorithms described in Section 3.2.3. The first configuration was univariate, using only the x-axis of the accelerometer; results are reported in Table 16. The experiment was then repeated with parallel channels of length 200, where it was found that stacking the (partially overlapping) windows led to a clear performance gain (Table 17). Finally, 50 samples were concatenated from the start of each sequence in front of the selfie-centered window (total length 250). This window performed slightly better than the standalone selfie-centered window, but still considerably worse than the "double" window (Table 18):

Table 16: Model-wise verification results (mean ± SD over 5-folds: x-acc, single 200 data point selfie-centered window, 5 resamples of inner folds, 1 repeat.

Model	**EER (%)**	**FRR (%)**	**FAR (%)**
QUANT	12.62	1.33	68.07
MR-Hydra	30.63	0.00	100.00
RDST	28.70	0.00	100.00
ResNet	20.14	5.78	55.85
WEASEL+MUSE	16.62	3.22	46.60
RSTSF	12.78	2.00	68.71
catch22	17.15	2.78	75.01

Table 17: Model-wise verification results (mean ± SD over 5-folds: x-acc, double window, 5 resamples of inner folds, 1 repeat).

Model	**EER (%)**	**FRR (%)**	**FAR (%)**
QUANT	7.99	1.00	66.57
MR-Hydra	18.59	0.00	100.00
RDST	20.75	0.00	100.00
ResNet	14.43	4.44	50.74
WEASEL+MUSE	8.04	2.22	36.56
RSTSF	9.25	0.78	55.71
catch22	10.27	1.56	54.31

Table 18: Model-wise verification results (mean ± SD over 5-folds: x-acc, single window, concatenated 50 + 200, 5 resamples of inner folds, 1 repeat / resample of outer folds.

Model	**EER (%)**	**FRR (%)**	**FAR (%)**
QUANT	10.18	1.44	65.35

Model	EER (%)	FRR (%)	FAR (%)
MR-Hydra	24.56	0.00	100.00
RDST	23.58	0.00	100.00
ResNet	18.97	6.78	47.97
WEASEL+MUSE	12.77	3.33	43.10
RSTSF	10.61	1.56	60.54
catch22	15.55	1.89	69.69

Next, more channels were included in the benchmark, while using the seemingly best-performing "double window" configuration. In the next experiment, the x-axis of the accelerometer was accompanied with the x-axes of gyro and magnetometer, the results of which are displayed in Table 19 Including more adjacent x-axis sensors improved the performance notably for all benchmarked algorithms. The dictionary-based WEASEL+MUSE saw by far the most dramatic improvement with EER dropping as low as 2% combined with FRR of 1.67% and FAR of 14.50%, becoming easily the best-performing algorithm, no competition. The EER's for the the interval-based QUANT and RSTSF, which performed the best in the single sensor configurations, decreased even further by around 4%.

Table 19: Model-wise verification results (mean ± SD over 5-folds: x-acc, x-gyro, x-mag, double window, 5 resamples of inner folds, 1 repeat).

Model	EER (%)	FRR (%)	FAR (%)
QUANT	6.30	0.78	62.12
MR-Hydra	15.80	0.00	100.00
RDST	14.77	0.00	100.00
ResNet	9.81	2.33	44.55
WEASEL+MUSE	2.00	1.67	14.50
RSTSF	6.10	0.78	55.81
catch22	7.26	2.18	59.89

Table 20 summarizes the results for another three-channel configuration, this time with all three accelerometer axes included. In this configuration, the results improve even further across the board. The lightweight, feature-based Catch22 emerges as the second-best performer by EER, while WEASEL+MUSE's EER dropped even further to 1.29%, easily keeping its first spot. Notably, the FRR also dropped to <1%.

Table 20: Model-wise verification results (mean ± SD over 5 folds: x, y and z axes of accelerometer, double window 5 resamples of inner folds, 1 repeat).

Model	EER (%)	FRR (%)	FAR (%)
QUANT	5.17	1.00	55.26

Model	EER (%)	FRR (%)	FAR (%)
MR-Hydra	9.42	0.00	100.00
RDST	11.09	0.00	100.00
ResNet	8.46	4.00	38.47
WEASEL+MUSE	1.29	0.44	22.73
RSTSF	5.21	1.00	55.91
catch22	4.35	1.11	54.35

Table 21 reports results for the configuration that includes all three axes from the accelerometer, gyroscope, and processed magnetometer. Performance figures are averaged over five outer-fold resamples, and standard deviations across repeats are reported for each algorithm. The standard deviations are modest in absolute terms. However, for many algorithms, FRR shows notable relative variability across repeats and should be interpreted accordingly.

Including all axes from all 3 sensors yielded consistent improvements over the previously reported settings. The strongest performer was still by far WEASEL+MUSE. The improvement in EER was not as dramatic as when moving from the single channel x-accelerometer axis configuration to including all 3 axes, but still notable. The most impressive improvement was seen in the FAR value, which dropped to 7.31%, combined with an FRR of 1.40%. Relative to the other methods evaluated, these figures indicate a clear performance advantage.

Table 21: Model-wise verification results (mean ± SD over 5-folds: all 9 channels, double window, 5 resamples of inner folds. 5 repeats, only 1 for RESNET).

Model	EER (%)	FRR (%)	FAR (%)
QUANT	5.06 ± 0.50	1.07 ± 0.48	57.91 ± 2.79
MR-Hydra	9.85 ± 0.38	0.00 ± 0.00	100.00 ± 0.00
RDST	9.86 ± 0.58	0.00 ± 0.00	100.00 ± 0.00
WEASEL+MUSE	1.07 ± 0.09	1.40 ± 0.19	7.01 ± 0.34
RSTSF	5.09 ± 0.15	1.04 ± 0.28	52.84 ± 0.92
catch22	4.85 ± 1.59	1.24 ± 1.15	65.53 ± 7.31
ResNet (1 repeat)	6.76 ± 0.00	3.78 ± 0.00	33.96 ± 0.00

Overall, several state-of-the-art methods most notably MultiRocket-Hydra (MR-Hydra) and RDST—performed poorly when operating at low FRR targets: for these algorithms, FAR consistently settled at 100%. A likely cause is the steep class-probability outputs produced by ridge classifiers (used by both algorithms), which yield class probability distributions that are effectively one-hot. Such probability scores leave little room for threshold calibration, as practically all calibration scores will be either 0's or 1's.

The exceptionally strong performance of WEASEL+MUSE could be explained with its design focus on multivariate time-series classification: it explicitly builds features that leverage multi-channel information. In Tables 16, 17 and 18, it can be observed that WEASEL+MUSE lags behind interval-based methods such as QUANT and RSTSF in the single-channel setting, but easily outperforms them once genuinely multivariate sensor data are used. Even the "double-window" configuration provides two intervals of the same signal rather than correlated sensors/axes, so there is no real multi-channel structure there to exploit. While QUANT, Catch22, and RSTSF accept multichannel inputs, they generally compute features per channel and then concatenate or aggregate them, rather than modeling cross-channel dependencies explicitly. WEASEL+MUSE was designed for multivariate time series, and aims to account for interplay across dimensions (Schäfer & Leser, 2018). This more explicit handling of inter-axis and inter-sensor correlations likely explains its far stronger performance on our truly multivariate configurations.

In an additional experiment, only QUANT was included (due to its consistently strong performance across previous sensor configurations and its short training time) to quantify *relative* performance differences by sensor type. The results are presented in Table 22

Table 22: QUANT's verification results for every sensor type, all axes included: mean ± SD over 5-folds, double window (5 resamples of inner folds, 5 repeats).

Sensor	**EER (%)**	**FRR (%)**	**FAR (%)**
Accelerometer	5.30 ± 0.28	1.40 ± 0.46	57.33 ± 3.30
Linear acceleration	8.95 ± 0.70	1.16 ± 0.48	74.76 ± 1.46
Gyroscope	9.60± 0.76	1.22 ± 0.48	77.29 ± 1.50
Processed Magnetometer	16.29 ± 0.37	1.49 ± 0.63	81.45 ± 2.71
Raw Magnetometer (context leak)	1.86 ± 0.21	0.82 ± 0.29	15.97 ± 1.06

Table 22 indicates that the accelerometer carries the strongest discriminative signal, with steep declines in EER/FRR/FAR when using only gyroscope or magnetometer channels, consistent with prior findings that accelerometers often lead performance in smartphone time-series classification while gyroscopes are complementary and magnetometers are typically less informative (Huang et al., 2023; S. Zhang et al., 2022).

The raw magnetometer configuration shows clear context leakage. It reports EER = 1.86%, FRR < 1%, and FAR = 15.97%, ominously outperforming all other sensor setups by a huge margin. These figures are not considered valid, as they are mostly influenced by context rather than subject-specific patterns (see Section 3.1.2 for details).

The performance using linear acceleration also proved poor compared to raw acceleration, demonstrating the importance of gravity/orientation-related cues in the raw accelerometer readings. By only using the x, y and z axes of the accelerometer, EER drops to a level almost as low as when including all 9 sensor channels, while using only a third of the compute time. However, FRR stays 0.4% higher.

Finally, classification performance metrics for each benchmarked algorithm are reported in Table 23, with the predicted class obtained via argmax over output scores. In these results, MR-Hydra performs as intended for time-series classification, matching or finishing a very close second to MUSE across all metrics. This highlights an important point: the algorithms that excel at multi-class classification (in essence, assigning the highest probability to the correct class) is not necessarily the most compatible with user verification, where a single score is thresholded to accept or reject and the quality of the score calibration ends up mattering more than raw accuracy. In another words: user verification stresses different aspects of the algorithms than plain accuracy.

The ResNet classifier was also trained for its default 1500 epochs in addition to the earlier 150 epoch setting, yielding substantial gains in performance, and indicating that its full potential may not have been utilized in the previous benchmarks.

Table 23: Overall macro metrics (5-fold CV). x, y and z -axes of accelerometer, double window. Accuracy, precision, recall, and F1 are reported as percentages.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
WEASEL+MUSE	87.20	88.70	86.78	85.55
MR-HYDRA	87.20	87.79	86.67	85.34
QUANT	79.20	79.75	78.44	76.30
RSTSF	77.60	77.52	76.44	74.17
ResNet (150 epochs)	53.87	53.06	53.56	49.94
ResNet (1500 epochs)	69.07	72.11	68.44	66.84
Catch22	77.33	76.33	76.78	74.47
RDST	84.53	87.22	84.11	83.47

5 Summary

In this thesis, the aim was to determine whether motion sensor time series data recorded during selfie capture sequences could be utilized to support Candour's facial recognition engine in detecting possible video injection-based (and other) spoofing attempts, and to evaluate to which extent the handling patterns are person-specific and usable for verifying individual users. To this end, a data set was collected from 30 individuals, comprising 375 selfie capture motion sequences. Magnetometer signals could not be fully utilized due to heading-related context leakage in the collection setup. Three benchmarks were conducted: (i) A comparison of various time series classification algorithms to determine which of them best distinguish individual users, (ii) a one-class anomaly detection setting trained on only 10 sequences per user to assess detection of zero-effort impostors and (iii) a spoof-detection task using the same techniques to differentiate mimicked spoof attempts from *any* genuine users.

The user classification results were better than expected, with very low EER scores obtained. Best results could be obtained by using a stacked, 2-window configuration, including 200 first samples after the user opens the selfie camera, and a window of 50 samples before and 150 after the selfie is captured. The best-performing time series classification algorithm WEASEL+MUSE reached an impressively low mean EER of 1.07% across 5 repeats when using all 9 motion sensors, combined with an FRR of 1.40% and FAR of 7.01% using an universal, pre-set threshold. A similarly impressive result with an EER of 1.29%, FRR of 0.44% and FAR of 22.73% was achieved by using only the 3 accelerometer axes as input. The next-best algorithm, the lightweight feature-based catch22, reached an EER of 4.35% with an FRR and FAR of 1.11% 54.35% respectively when using the 3-axis accelerometer configuration, trailing far behind WEASEL+MUSE. WEASEL+MUSE's strong performance in multi-axis and multi-sensor channel configurations highlights the advantage of using TSC algorithms that are designed for multivariate time series analysis, taking into account and extracting features related to the interplay between different channels.

In a strict classification setting, no single method fully dominated: MR-HYDRA, which performed poorly in the verification benchmark matched WEASEL+MUSE in classification accuracy. This highlights that high label prediction accuracy does not automatically translate to user verification performance, which depends on setting a threshold on the enrolled user's output score. The classifier type is decisive here because it determines the distribution and scale of the output class probability scores. MR-HYDRA outputs one-hot class probability scores, which made decision threshold setting practically impossible, as there was effectively two choices: 0 (accept all) or >0 (always accept the predicted class). Overall, low-granularity output scores make thresholding difficult: they leave less room for adjustments, and can cause wide swings in FRR/FAR. By contrast, classifiers that yield smoother, fine-grained and more evenly scaled scores provide finer percentage precision and more responsive and stable behavior under thresholding, which is critical for verification. WEASEL+MUSE's dominance could thus partially also be explained by its high-resolution class score distributions, as the benchmarked MUSE implementation uses an L2-regularized LogisticRegression (class-weighted, max iterations = 5000) that produces smooth and continuous-valued scores rather than coarse vote fractions or straight up one-hot outputs, making threshold tuning more stable.

The 10-shot semi-supervised user verification experiment yielded interesting results. The state-of-the-art anomaly detection methods did not fare as well as expected, while simpler, classical approaches yielded the best results. An FRR of 0% combined with an FAR of 79% was achieved with the plain Euclidean 3-NN, indicating that 1 out of 5 zero-effort impostors could be rejected while keeping the risk of false rejections minimal. A similar result was achieved using 3-NN DTW, with an FRR of 0.87% and an FAR of 74.77%, correctly identifying 1 in 4 zero-effort impostors while keeping the FRR under 1%. While these FAR figures are still high, zero-effort impostor attacks are fairly rare, and would be rejected by the facial recognition engine with practically full certainty.

The genuine-vs-spoof anomaly detection experiment yielded satisfactory results that indicate that such a motion data analysis step could provide high benefits with little risk. This is where the state-of-the-art WTSAD method ROCKAD lived up to its promise, as more genuine samples were available for the training. The easiest spoof type to detect were the static sequences, in which the phone simply sits still on a surface, as an FRR of 0% combined with an FAR of 0% was achieved. The second easiest were the handheld spoofs, in which the camera is pointed to a face displayed on the screen of another device. The biggest challenge were the "delayed" spoofs mimicking live-morphing video injection attacks with a 1.5-3 second delay. For those, the best FAR achieved was 67.78% (combined with an FRR of 0%), which means that about 1 in 3 would be correctly rejected.

5.1 Quality of Results

The results showed that an analysis for motion sensor-based time series data can be a strong and reliable addition to spoof detection, and potentially also in user verification.

The main weakness of the collected data was the fact that all samples were collected in a sitting position. In daily use cases, the selfie capture would likely be regularly also performed in a standing position, and (more rarely) a supine position. Other, likely rarer scenarios involve performing the selfie capture while walking or traveling in a moving vehicle.

Another weakness comes from the fact that all sequences were collected in one sitting. While a large portion of the subjects took small breaks between and switched their posture/position between the sequences, a large amount of sequences may have been more identical than if they were collected "naturally" in multiple separate sessions. Though, the users with more uniform time series patterns typically had higher FRR scores than those with more variation, as the per-user rejection thresholds were set lower.

5.2 Deployment Considerations

The TSC user-verification results are encouraging, but the dataset comprises only 30 users. At production scale with thousands of users, maintaining a closed-set, multiclass model becomes impractical. Further work is needed on how to translate the strengths of these classification algorithms into more scalable and production-ready solutions.

The 10-shot user verification approach would be a simpler and a more lightweight solution, but significantly less reliable with its high FAR values. However, it should be kept in mind that the experiment measured performance on zero-effort impostors, which would very likely be caught with Candour's facial recognition engine alone anyway.

Device heterogeneity must also be considered. In deployment, many budget devices will contain lower-quality motion sensors, and some sensors (particularly gyroscopes) may be unavailable. Tablets should also be treated as part of the mobile device population, and their usage patterns likely differ significantly from smartphones, as they are typically heavier and handled with two hands.

5.3 Possible Future Work

Possible future work could include collecting sequences in standing and supine positions, as well as walking. TSC algorithms could be trained to detect the position/orientation of the user before the anomaly detection or user verification step is applied. Another angle could include employing sensor fusion techniques to combine the information from each of the 3 IMU sensor outputs into a single data channel for each axis rather than naively adding individual channels in parallel as input as was done in this thesis.

As of this writing, a patent application covering aspects of the findings of this thesis has been filed by Candour Oy. The next step is to further refine the approaches benchmarked here and officially begin development of a motion analysis engine, with the goal of piloting an experimental version of it in production in 2026.

References

Alexandersson, Á., Steingrímsdóttir, T., Marque, C., Terrien, J., & Karlsson, B. (2015). The icelandic 16-electrode electrohysterogram database. *Scientific Data*, *2*, 150017. https://doi.org/10.1038/sdata.2015.17

Almeida, F., Pereira, R., & Santos, J. (2022). Use of filters to smooth out signals collected through mobile devices: A review. *Applied Sciences*, *12*(13). https://doi.org/10.3390/app12136579

Android Developers. (2023). *Motion sensors* [Web page]. Google LLC. Retrieved July 11, 2025, from https://developer.android.com/develop/sensors-and-location/sensors/sensors_motion

Android Developers. (2024). *Sensors overview — sensor coordinate system (figure 1)* [Figure] [In *Sensors overview*]. Google LLC. Retrieved September 20, 2025, from https://developer.android.com/guide/topics/sensors/sensors_overview#sensors-coords

Android Developers. (2025). *Sensors overview* [Web page]. Google LLC. Retrieved July 11, 2025, from https://developer.android.com/develop/sensors-and-location/sensors/sensors_overview

Apple Inc. (2025). *CMDeviceMotion* [Developer documentation]. Apple Developer. Retrieved September 5, 2025, from https://developer.apple.com/documentation/coremotion/cmdevicemotion

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*(3), 606–660. https://doi.org/10.1007/s10618-016-0483-9

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

Bringmann, K., Fischer, N., van der Hoog, I., Kipouridis, E., Kociumaka, T., & Rotenberg, E. (2023). Dynamic dynamic time warping. *arXiv preprint arXiv:2310.18128*. https://doi.org/10.48550/arXiv.2310.18128

Cabello, N., Naghizade, E., Qi, J., & Kulik, L. (2024). Fast, accurate and explainable time series classification through randomization (r-STSF). *Data Mining and Knowledge Discovery*. https://doi.org/10.1007/s10618-023-00978-w

Carta, K., Barral, C., El Mrabet, N., & Mouille, S. (2022). Video injection attacks on remote digital identity verification solution using face recognition. *Proc. 13th Int. Multi-Conf. Complexity, Informatics and Cybernetics (IMCIC)*, 92–95. https://www.iiis.org/CDs2022/CD2022Spring/papers/ZA639OX.pdf

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, *41*(3), 1–58. https://doi.org/10.1145/1541880.1541882

Chen, K.-H., Yang, J.-J., & Jaw, F.-S. (2016). Accelerometer-based fall detection using feature extraction and support vector machine algorithms. *Instrumentation Science & Technology*, *44*(4), 333–342. https://doi.org/10.1080/10739149.2015.1123161

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., & Keogh, E. (2019). The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, *6*(6), 1293–1305. https://doi.org/10.1109/JAS.2019.1911747

DeepFaceLive contributors. (2024). *DeepFaceLive (version 3.1.0)* [Computer software]. Retrieved September 17, 2025, from https://github.com/iperov/DeepFaceLive

Dempster, A., Nguyen, T., & Webb, G. I. (2024). QUANT: A minimalist interval method for time series classification. *Data Mining and Knowledge Discovery*. https://doi.org/10.1007/s10618-024-01036-9

Dempster, A., Petitjean, F., & Webb, G. I. (2020). Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, *34*(5), 1454–1495. https://doi.org/10.1007/s10618-020-00701-z

Dempster, A., Schmidt, D. F., & Webb, G. I. (2023). HYDRA: Competing convolutional kernels for fast and accurate time series classification. *Data Mining and Knowledge Discovery*, *37*(5), 1779–1805. https://doi.org/10.1007/s10618-023-00939-3

Dimaratos, A., & Pöhn, D. (2023). Evaluation scheme to analyze keystroke dynamics methods. *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP)*, 357–365. https://doi.org/10.5220/0011626100003405

Estrela, P. M. A. B., de Oliveira Albuquerque, R., Amaral, D. M., Giozza, W. F., & de Sousa Júnior, R. T. (2021). A framework for continuous authentication based on touch dynamics biometrics for mobile banking applications. *Sensors*, *21*(12), 4212. https://doi.org/10.3390/s21124212

Fatima, K., Schuckers, M., Cruz-Ortiz, G., Hou, D., Purnapatra, S., Andrews, T., Neupane, A., Marshall, B., & Schuckers, S. (2024). A large-scale study of performance and equity of commercial remote identity verification technologies across demographics. *arXiv*. https://doi.org/10.48550/arXiv.2409.12318

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: A review. *Data Mining and Knowledge Discovery*, *33*, 917–963. https://doi.org/10.1007/s10618-019-00619-1

Fulcher, B. D., & Jones, N. S. (2017). hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. *Cell Systems*, *5*(5), 527–531. https://doi.org/10.1016/j.cels.2017.10.001

Guillaume, A., Vrain, C., & Elloumi, W. (2021). Random dilated shapelet transform: A new approach for time series shapelets.

Huang, X., Zhang, Y., Ma, Y., Zhang, F., & Wang, C. (2023). Sensor-based wearable systems for monitoring human motion and posture: A systematic review. *Micromachines*, *14*(10), 1990. https://doi.org/10.3390/mi14101990

Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., & Forestier, G. (2025). Look into the lite in deep learning for time series classification. *International Journal of Data Science and Analytics*. https://doi.org/10.1007/s41060-024-00708-5

ISO/IEC. (2021). *Information technology — biometric performance testing and reporting — part 1: Principles and framework*. International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).

Jain, A. K., Ross, A., & Prabhakar, S. (2004). An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, *14*(1), 4–20. https://doi.org/10.1109/TCSVT.2003.818349

Jain, A. K., Ross, A. A., & Nandakumar, K. (2011). *Introduction to biometrics*. Springer. https://doi.org/10.1007/978-0-387-77326-1

Keogh, E., & Kasetty, S. (2003). On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, *7*(4), 349–371. https://www.cs.ucr.edu/~eamonn/Data_Mining_Journal_Keogh.pdf

Leimhofer, J., Petrovic, M., Dominik, A., Heider, D., & Hegerl, U. (2025). Cross-platform availability of smartphone sensors for depression indication systems: Mixed-methods umbrella review. *Interactive Journal of Medical Research*, *14*, e69686. https://doi.org/10.2196/69686

Lima, W. S., Souto, E., El-Khatib, K., Jalali, R., & Gama, J. (2019). Human activity recognition using inertial sensors in a smartphone: An overview. *Sensors*, *19*(14), 3213. https://doi.org/10.3390/s19143213

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. *Proceedings of the 2008 IEEE International Conference on Data Mining (ICDM)*, 413–422. https://doi.org/10.1109/ICDM.2008.17

Liu, M., Geißler, D., Bian, S., Zhou, B., & Lukowicz, P. (2025). Assessing the impact of sampling irregularity in time series data: Human activity recognition as a case study. *arXiv*, *2501.15330*. https://doi.org/10.48550/arXiv.2501.15330

Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., & Jones, N. S. (2019). Catch22: CAnonical time-series CHaracteristics selected through highly comparative time-series analysis. *Data Mining and Knowledge Discovery*, *33*(6), 1821–1852. https://doi.org/10.1007/s10618-019-00647-x

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). LSTM-based encoder–decoder for multi-sensor anomaly detection. *Proceedings of the 2016 ICML Workshop on Anomaly Detection*.

Middlehurst, M., Ismail-Fawaz, A., Guillaume, A., Holder, C., Guijo-Rubio, D., Bulatova, G., Tsaprounis, L., Mentel, L., Walter, M., Schäfer, P., & Bagnall, A. (2024a). Aeon: A python toolkit for learning from time series. *Journal of Machine Learning Research*, *25*(289), 1–10. http://jmlr.org/papers/v25/23-1444.html

Middlehurst, M., Schäfer, P., & Bagnall, A. (2024b). Bake off redux: A review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, *38*, 1958–2031.

Mohamed, M., Shrestha, B., & Saxena, N. (2017). Smashed: Sniffing and manipulating Android sensor data for offensive purposes. *IEEE Transactions on Information*

Forensics and Security, *12*(4), 901–913. https://doi.org/10.1109/TIFS.2016.2620278

Pennsylvania State University. (2023, October 16). *Deepfakes expose vulnerabilities in certain facial recognition technologies*. Penn State News. Retrieved October 1, 2025, from https://www.psu.edu/news/information-sciences-and-technology/story/deepfakes-expose-vulnerabilities-certain-facial

Qiao, M., Zhai, Y., & Wang, Y. (2025). Contrastive learning for continuous touch-based authentication. *CoRR*, *abs/2504.17271*. https://arxiv.org/abs/2504.17271

Sae-Bae, N., & Memon, N. (2022). Distinguishability of keystroke dynamic template. *PLOS ONE*, *17*(1), e0261291. https://doi.org/10.1371/journal.pone.0261291

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *26*(1), 43–49. https://doi.org/10.1109/TASSP.1978.1163055

Schäfer, P., & Leser, U. (2018). Multivariate time series classification with WEASEL+MUSE. *Proceedings of the 3rd ECML PKDD Workshop on Advanced Analytics and Learning on Temporal Data (AALTD)*, 1–15.

Schmidl, S., Wenig, P., & Papenbrock, T. (2022). Anomaly detection in time series: A comprehensive evaluation. *Proceedings of the VLDB Endowment*, *15*(9), 1779–1797.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems 13* (pp. 582–588). MIT Press.

Shen, Z., Li, S., Zhao, X., & Zou, J. (2022). MMAuth: A continuous authentication framework on smartphones using multiple modalities. *IEEE Transactions on Information Forensics and Security*, *17*, 1450–1465. https://doi.org/10.1109/TIFS.2022.3160361

Sitová, Z., Sedenka, J., Yang, Q., Peng, G., Zhou, G., Gasti, P., & Balagani, K. (2015). HMOG: New behavioral biometric features for continuous authentication of smartphone users. *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 147–158. https://doi.org/10.1145/2750858.2805828

Sørbø, S., & Ruocco, M. (2023). Navigating the metric maze: A taxonomy of evaluation metrics for anomaly detection in time series. *Data Mining and Knowledge Discovery*, *38*(3), 1027–1068. https://doi.org/10.1007/s10618-023-00988-8

Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A. K., Sonne, T., & Jensen, M. M. (2015). Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15)*, 127–140. https://doi.org/10.1145/2809695.2809718

Suvorkin, V., Garcia-Fernandez, M., González-Casado, G., Li, M., & Rovira-Garcia, A. (2024). Assessment of noise of MEMS IMU sensors of different grades for GNSS/IMU navigation. *Sensors*, *24*(6), 1953. https://doi.org/10.3390/s24061953

Tax, D. M. J., & Duin, R. P. W. (2004). Support vector data description. *Machine Learning*, *54*(1), 45–66. https://doi.org/10.1023/B:MACH.0000008084.60811.49

Theissler, A., Wengert, M., & Gerschner, F. (2023). Rockad: Transferring ROCKET to whole time series anomaly detection. In B. Crémilleux, B. Hess, & S. Nijssen (Eds.), *Advances in intelligent data analysis xxi: 21st international symposium on intelligent data analysis (ida 2023), louvain-la-neuve, belgium, april 12–14, 2023* (pp. 419–432, Vol. 13876). Springer. https://doi.org/10.1007/978-3-031-30047-9_33

Vincent, J. (2022, May 18). *A deepfake attack reportedly fooled a facial recognition test used by banks* [News article]. The Verge. Retrieved September 20, 2025, from https://www.theverge.com/2022/5/18/23092964/deepfake-attack-facial-recognition-liveness-test-banks-sensity-report

Wang, Z., Yan, W., & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*, 1578–1585. https://doi.org/10.1109/IJCNN.2017.7966039

Xie, H., Gu, T., Tao, X., Ye, H., & Lv, J. (2014). Maloc: A practical magnetic fingerprinting approach to indoor localization using smartphones. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, 243–252. https://doi.org/10.1145/2632048.2632057

Yamane, T., Kimura, M., & Morita, M. (2025). Effects of sampling frequency on human activity recognition with machine learning aiming at clinical applications. *Sensors*, *25*(12), 3780. https://doi.org/10.3390/s25123780

Zhang, S., Hou, X., Wang, Z., Li, L., Liu, A., & Liang, X. (2022). Deep learning in human activity recognition with wearable sensors: A review on advances. *Sensors*, *22*(4), 1476. https://doi.org/10.3390/s22041476

Zhang, Y., Chang, S., & Jones, D. A. (2023a). Range-based equal error rate for spoof localization. *Proceedings of Interspeech 2023*, 0001–0005. https://doi.org/10.21437/Interspeech.2023-1234

Zhang, Y., Yu, S., Zhao, W., & Plataniotis, K. N. (2023b). *Advheat: Adversarial head-turn attack against deepfake detectors* [Preprint]. arXiv: 2305.12345. Retrieved September 20, 2025, from https://arxiv.org/abs/2305.12345

Zhang, Y., Song, X., Yang, L., Qiao, X., Hao, D., & Zheng, D. (2022). Assessment of features between multichannel electrohysterogram for differentiation of labors. *Sensors*, *22*(9), 3352. https://doi.org/10.3390/s22093352

Zhou, L., Fischer, E., Tunca, C., Brahms, C. M., Ersoy, C., Granacher, U., & Arnrich, B. (2020). How we found our imu: Guidelines to imu selection and a comparison of seven imus for pervasive healthcare applications. *Sensors*, *20*(15), 4090. https://doi.org/10.3390/s20154090

A Extra Experiments

An additional experiment converted the top three TSC methods into per-user binary classifiers (one-vs-rest). For each user, a separate model was trained with two classes: (i) impostor, comprised of sequences from all other users, and (ii) genuine, comprised only of that user's sequences. The per-user decision threshold τ_u was set on the genuine scores with a target of 0% FRR on the training split. The results (presented in Table A1) use all three accelerometer axes with two concatenated 200-sample windows.

Table A1: Per-user binary verification (all 3 acc axes; double 200-sample windows). 5 outer folds, 3 inner folds. Means ± SD over folds. τ_u set to achieve 0% FRR on train.

Method	EER (%)	FRR@τ_u (%)	FAR@τ_u (%)
MUSE	3.83 ± 1.82	6.56 ± 3.78	11.58 ± 1.38
QUANT	5.83 ± 1.33	7.67 ± 1.27	21.97 ± 0.63
Catch22	9.73 ± 1.04	8.56 ± 2.85	25.20 ± 2.57

It should be noted that this setup has a substantial class imbalance (impostor:genuine ≈ 29:1) during both training and evaluation. Operationally, this approach can be production-friendly by drawing the impostor class by randomly sampling sequences from background users, but this would require training and maintaining a classifier for each individual user.

Table A2 reports two small additional experiments. LITETime (Ismail-Fawaz et al., 2025) is a lighter variant of InceptionTime that matches its accuracy. It was trained with default settings except that the number of epochs was reduced to 500. The earlier ResNet run with 150 epochs used in the main benchmark did not provide a fair representation of deep learning methods, so a more state-of-the-art model was included here for completeness. QUANT was rerun with higher output granularity by increasing the number of estimators in the ExtraTreesClassifier from 200 to 1000, to test the effect of increasing the output score resolution.

Table A2: Multi-class verification (all 3 ACC axes; double 200 sample windows). 5 outer folds, 3 inner folds. Means ± SD over folds. τ set to achieve 1% FRR on train.

Method	EER (%)	FRR@τ (%)	FAR@τ (%)	Acc (%)
LITETime	3.95 ± 1.35	1.81 ± 1.89	26.12 ± 1.20	78.67 ± 3.27
QUANT	4.42 ± 1.08	1.00 ± 0.61	58.32 ± 4.89	77.07 ± 5.28

Despite the long training time (about 14 hours on a MacBook Pro M3 with GPU acceleration), LITETime was highly competitive in both EER and FRR/FAR and would place second in the original user verification setting under the same sensor and window configuration. Only 500 epochs were used while the default is 1500, so further gains are possible with more compute. These results indicate that deep learning methods are competitive after all for user verification, and by exploiting the multivariate structure of multi-channel time series they can be strong contenders alongside WEASEL+MUSE.

QUANT also shows a modest gain (about 0.7% in EER), indicating that increasing the output score resolution alone does not yield dramatic improvements, but can provide an incremental and measurable benefit.

B Larger Percentile Band Plots

For detailed inspection, the six plots from Figure 12 are displayed here on a larger scale.

Figure B1: Percentile bands around capture for accelerometer x-axis. The dashed vertical line marks the selfie-capture moment.

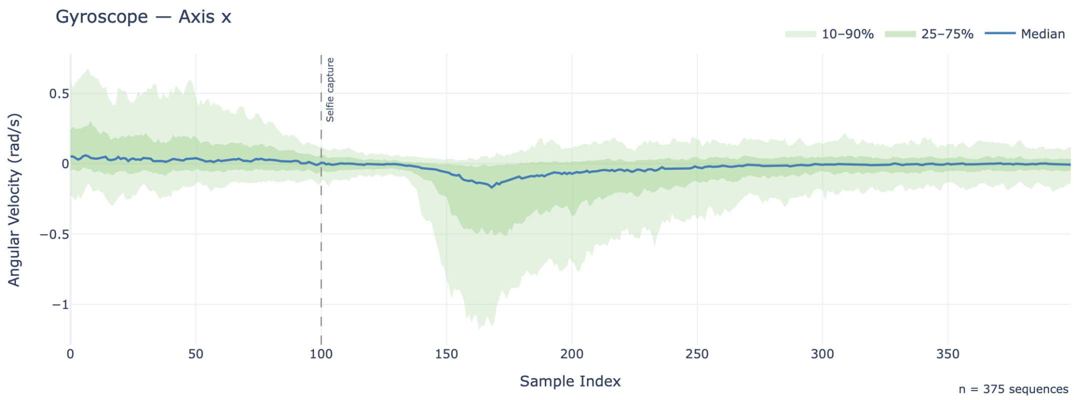


Figure B2: Percentile bands around capture for gyroscope x-axis. The dashed vertical line marks the selfie-capture moment.

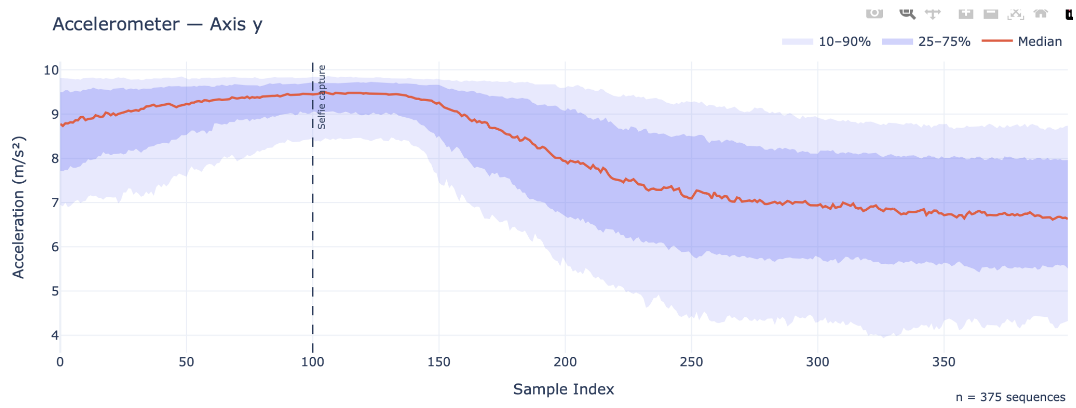


Figure B3: Percentile bands around capture for accelerometer y-axis. The dashed vertical line marks the selfie-capture moment.

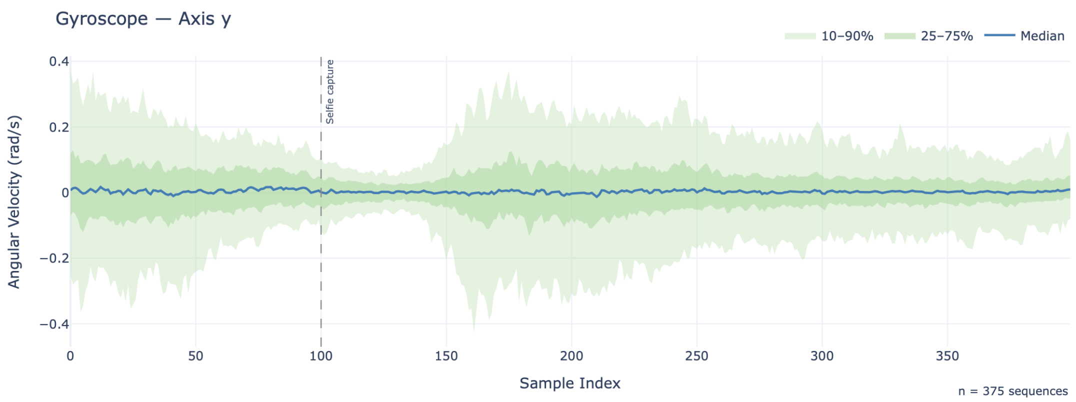


Figure B4: Percentile bands around capture for gyroscope y-axis. The dashed vertical line marks the selfie-capture moment.

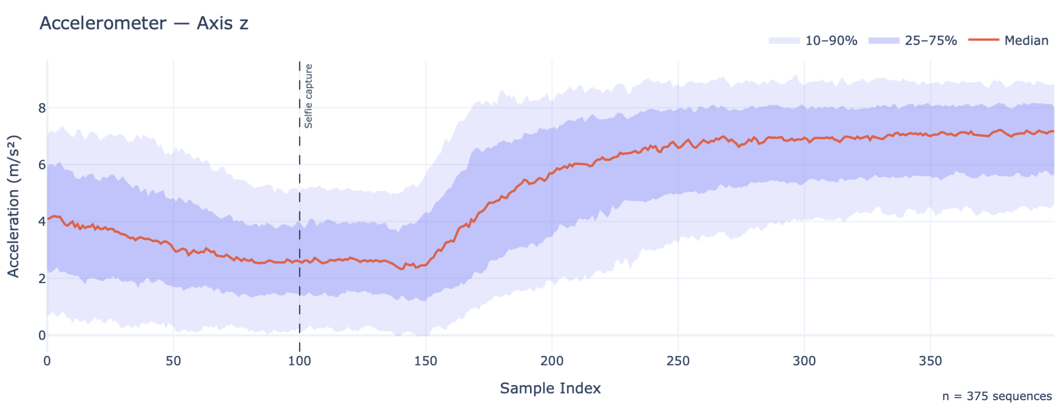


Figure B5: Percentile bands around capture for accelerometer z-axis. The dashed vertical line marks the selfie-capture moment.

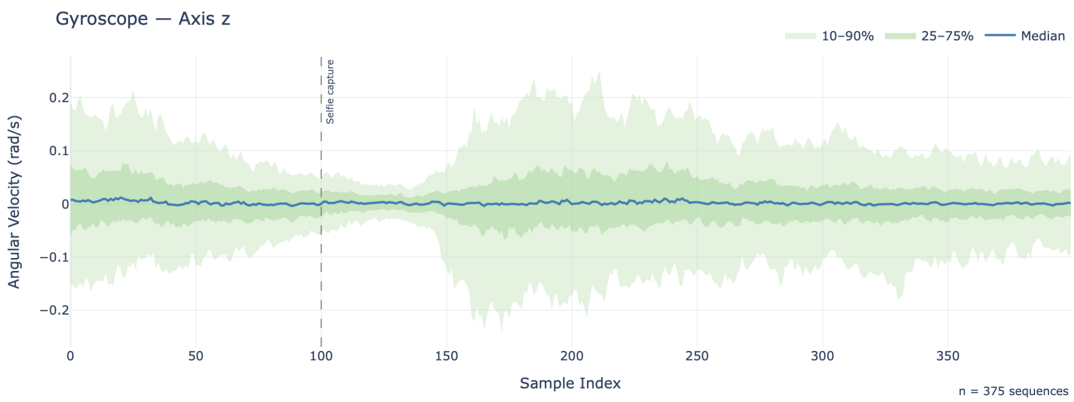


Figure B6: Percentile bands around capture for gyroscope z-axis. The dashed vertical line marks the selfie-capture moment.