TU WIEN Informatics

Impact of Time Discrepancy on Machine Learning Models for Human Activity Recognition

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Media and Human-Centered Computing

eingereicht von

Patrick Trollmann, BSc
Matrikelnummer 11907096

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Inf. Dr.sc.techn. Florian Michahelles
Univ.Ass. Dr.-Ing. Florian Wolling, MSc
Mitwirkung: Projektass. Dipl.-Ing. David Kostolani, BSc

Wien, 15. März 2025

Patrick Trollmann   Florian Michahelles

Informatics

Impact of Time Discrepancy on Machine Learning Models for Human Activity Recognition

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media and Human-Centered Computing

by

Patrick Trollmann, BSc
Registration Number 11907096

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Inf. Dr.sc.techn. Florian Michahelles
Univ.Ass. Dr.-Ing. Florian Wolling, MSc
Assistance: Projektass. Dipl.-Ing. David Kostolani, BSc

Vienna, March 15, 2025

Patrick Trollmann   Florian Michahelles

Erklärung zur Verfassung der Arbeit

Patrick Trollmann, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 15. März 2025

Patrick Trollmann

Danksagung

Ich möchte gerne meinen aufrichtigen Dank an folgende Personen aussprechen, ohne die die Fertigstellung meiner Diplomarbeit nicht möglich gewesen wäre.

Florian
Ich bin zutiefst dankbar für meinen Betreuer Dr.-Ing. Florian Wolling, MSc für seine kontinuierliche Hilfe, seine Führung und Ratschläge durch komplexe Themen hindurch, welcher mich durch meine gesamte Arbeit durch hinweg begleitet hat. Seine Expertise hat dazu beigetragen den richtigen Weg zu zeigen und die Qualität dieser Arbeit zu verbessern.

David
Ich möchte außerdem Dipl.-Ing. David Kostolani, BSc danken, welcher mir wichtige Tipps und Tricks zum Thema Machine Learning gezeigt hat, und so auch dazu beigetragen hat meine Arbeit zu verbessern.

Rita
Ein weiterer und aufrichtiger Dank geht an meine Freundin und Partnerin, welche mich mit endloser Geduld unterstützt hat und mich in den vielen durchgearbeiteten Nächten stets motiviert hat.

Meinen Eltern
Nicht zuletzt bin ich auch meinen Eltern zu einem großen Dank verpflichtet. Ihr unerlässlicher Glaube an meine Fähigkeiten und ihre finanzielle Unterstützung waren Säulen, auf denen ich meine akademischen Leistungen aufbauen konnte.

VSC
Die präsentierten Rechenergebnisse wurden [zum Teil] am Vienna Scientific Cluster (VSC) erzielt.

Diese Diplomarbeit ist das Ergebnis von gemeinsamen Anstrengungen und der Unterstützung von wahnsinnig wunderbaren Menschen, und ich bin ihnen für all ihre Beiträge zu meiner akademischen Laufbahn aufrichtig dankbar.

Acknowledgements

I would like to express my sincere gratitude to the following people, without whom the completion of my diploma thesis would not have been possible.

Florian
I am deeply grateful to my supervisor, Dr.-Ing. Florian Wolling, MSc, for his continuous help, guidance and advice through complex topics, who has accompanied me throughout my entire work. His expertise has helped to point me in the right direction and improve the quality of this thesis.

David
I would also like to thank Dipl.-Ing. David Kostolani, BSc, who showed me important tips and tricks on machine learning and thus also helped to improve my work.

Rita
Another sincere thanks goes to my girlfriend and partner, who supported me with endless patience and always motivated me during the long nights of hard work.

My parents
Last but not least, I also owe a huge debt of gratitude to my parents. Their indispensable belief in my abilities and their financial support were the pillars on which I was able to build my academic achievements.

VSC
The computational results presented have been achieved [in part] using the Vienna Scientific Cluster (VSC).

This diploma thesis is the result of the combined efforts and support of incredibly wonderful people, and I am sincerely grateful for all their contributions to my academic career.

Kurzfassung

Der Bereich der menschlichen Aktivitätserkennung mit tragbaren Sensoren kann das tägliche Leben in mehrfacher Hinsicht beeinflussen. Dabei wird jedoch ein wichtiger Faktor oft vernachlässigt. Die Zeitsignale von verschiedenen Sensoren in Sensornetzwerken sollten synchronisiert sein, um somit eine bessere Genauigkeit zu gewährleisten. Praktisch existieren jedoch gewisse Zeitunterschiede zwischen einzelnen Sensoren. Diese werden unter anderem durch Laufzeitdifferenzen in der Datenübertragung oder durch Ungenauigkiten von den verbauten Quarzen verursacht. Diese Diplomarbeit befasst sich mit den verschiedenen Arten von Zeitunterschieden, im Speziellen bei Sensordaten in Form von Zeitreihen. Dafür wird eine Pipeline entwickelt, mit welcher verschiedene Arten von Zeitunterschieden auf Zeitsignale von mehreren inertialen Messeinheiten angewandt werden können. Mit diesen kontrolliert veränderten Zeitsignalen wird ein Machine Learning Modell trainiert. Bei dem Machine Learning Modell wird dabei auf ein bekanntes und für Aktivitäten aus dem täglichen Leben entwickeltes Modell aufgebaut. Um eine allgemeinere Aussage über mögliche Effekte und Auswirkungen von Zeitunterschieden treffen zu können, werden mehrere Machine Learning Modelle, welche sich in deren Komplexität unterscheiden und verschiedene Längen für die Eingangssignale besitzen, verwendet. Basierend auf den Genauigkeitsverlusten wird rückgeschlossen wie stark sich verschiedene Zeitunterschiede auf die Leistung und die Erkennung auswirken. Diese Arbeit soll einen Beitrag zum Verständnis der Auswirkungen von zeitlichen Diskrepanzen leisten.

Die Forschungsfragen konnten mit den Ergebnissen der Arbeit erfolgreich belegt werden. Minimale Taktverschiebungen, ab bereits 5 Zeiteinheiten, führen bei den untersuchten Machine Learning Modellen bereits zu Verlusten in der Genauigkeit um bis zu 12,56% und zu einem verringerten F1-Score um bis zu 22,43%. Die maximale Taktverschiebung zwischen den Signalen der verschiedenen Sensoren sollte deshalb kleiner als 0,06 bis 0,15 Sekunden sein. Am stärksten wirken sich Zeitunterschiede während des Trainings eines Machine Learning Modells aus. Die Frequenzabweichung von gängigen Quarzoszillatoren wirkt sich nicht auf die Leistungsfähigkeit der Modelle aus. Diese reduziert sich erst bei einer Abweichung der Frequenz von 2000 ppm. Jedoch bringen Frequenzabweichungen eine akkumulierende Taktverschiebung mit sich, welche die Genauigkeit der Modelle reduziert. Bei einer Abtastfrequenz von 33 Hz und einer Frequenzabweichung des Oszillators von $\pm 100\,ppm$ beträgt der Taktversatz nach einer Stunde bereits 0,36 Sekunden. Deshalb ist eine regelmäßige Synchronisierung des Taktes erforderlich, um eine Taktverschiebung und somit Verluste in der Leistungsfähigkeit zu vermeiden.

Abstract

Human activity recognition with wearable sensors can impact daily life in many ways. However, one important factor is often neglected. The time signals from different sensors in sensor networks should be synchronized to ensure better accuracy and data analysis. In practice, however, there are certain time discrepancies between individual sensors. These are caused, among other things, by runtime differences in data transmission or by inaccuracies in the quartz crystals used. This diploma thesis deals with the different types of time differences, particularly sensor data in the form of time series. Therefore, a pipeline is developed to implement different types of time differences to time signals from multiple inertial measurement units. A machine learning model is trained with these controlled, modified time signals. The machine learning model is based on a known model developed for activities of daily living. To make a more general statement about the possible effects and impacts of time differences, several machine learning models are used, which differ in their complexity and different lengths of the input signals. Based on the accuracy losses, it is concluded how strongly time differences affect performance and recognition. This work aims to contribute to the understanding of the effects of different temporal discrepancies.

The research questions were successfully answered with the results of the work. Minimal clock skews, from as little as 5 time units, already lead to losses in accuracy of up to 12.56% and a reduced F1-score of up to 22.43% in the machine learning models investigated. The maximum clock skew between the signals of the different sensors should, therefore, be less than 0.06 to 0.15 seconds. Time discrepancies have the greatest influence during the training of a machine learning model. Clock drifts of common quartz oscillators do not affect the performance of the models. This is only reduced at drifts of 2,000 ppm. However, clock drifts result in an accumulative clock skew, which reduces the accuracy of the models. With a sampling frequency of 33 Hz and a clock drift of the oscillator of $\pm 100\,ppm$, the clock skew accumulates to 0.36 seconds in the span of one hour. Therefore, regular synchronization of the clock is necessary to avoid clock skew and, thus, losses in performance.

Contents

CHAPTER 1

Introduction

A wide variety of devices are worn by most people nowadays [PKI+20, OSK+21]. These devices include smartphones, smartwatches, and headphones. For greater practicality, these devices are connected wirelessly to each other. [BAL09] While headphones, for example, were still largely wired a few years ago, today, most are wireless. In addition to their primary function, such as playing music, these devices can also often track certain activities of the human body. Headphones recognize when they are taken out of the ear. Smartwatches automatically recognize when a person is running and independently record workouts.

This variety of devices and sensors can also be used together to determine more precise activities of the human body [BAL09]. However, as they are connected wirelessly, this presents some challenges. Among other things, one issue is that the devices worn on the body are often from different manufacturers and based on different standards. Each of the devices worn has its own clock. These different clocks of the physically separate processors run asynchronously. Many devices only use local time stamps when recording data and are started at unidentifiable times. All of these causes pose some challenges and prevent synchronous recording of data. An example of the use of numerous body-worn sensors is shown in Figure 1.1. All these devices can be used to capture human movement, track vital parameters, and fulfill some other tasks as well. The illustration shows a person who uses a smartphone, which, among other things, can record movements and often serves as a central server for all other sensors worn by a person. The smartwatch and smart ring could, for example, record many vital parameters, such as heart rate, pulse, body temperature, and stress levels. A smart headset can not only play music but also track the position of the head using built-in sensors. In addition to other accelerometers, smart shoes also have pressure sensors that can measure body weight, detect foot misalignments, and analyze gait. The number of body-worn sensors just mentioned is not unusual. Due to the wide range of different standards, they use different time standards and scales [JWL+24, AQAAC17]. All this leads to different types of discrepancies in their time bases. These time discrepancies could, as a result, impact the accuracy of machine learning models and negatively influence the recognition of human activities. [WSL+19, XZSS12]

Likewise, inaccurate behavior monitoring could lead to misinterpretations of human activities. If the proportion of incorrectly recognized activities is too high, it makes no sense to create statistics or interpretations of activity sequences in different daily routines. It is also hard to generalize models to be used on any user, after being trained on some specific datasets with an unknown and inconsistent time discrepancy [WSL+19, XZSS12].

Figure 1.1: Several sensors worn by a person, consisting of a smartwatch, a headset, a smart ring, a smartphone, and smart shoes.

1.1 Problem Statement

Previous research indicates that a synchronization accuracy of less than one second may be sufficient for human activity recognition (HAR) and especially for activities of daily living (ADL) [BAL09]. The paper also states that a synchronization of less than 0.1 seconds is not necessary. This statement was made 15 years ago. There is still a critical gap in understanding the systematic impact of different time discrepancies on machine learning models used in HAR, particularly those employing advanced techniques like neural networks and deep learning. Inaccurate temporal data leads to inaccurate models and weak classification performance [WSL+19, XZSS12]. Depending on the application, this can have serious or potentially even fatal consequences for the people who depend on the devices. In the context of fall detection [CKC+05, YSMS14], an unrecognized fall due to temporal discrepancies results in a rescue chain that is not triggered.

State-of-the-art research often assumes that the sensors used are synchronized, although this is not the case, or that the lack of synchronization has so little effect that it is negligible. Many datasets are manually synchronized afterward through post-processing, as synchronization of several portable, wireless inertial measurement unit (IMU) sensors in real-time is complex. For real-time synchronization, for example, magnetometers and pulse generators [GLD$^+$24] or context markers [WSL$^+$19] are used.

A certain degree of synchronization is also possible through a master-slave system among the sensors in which several data packets with local timestamps of the devices are sent back and forth and the time of ping and response is measured at the same time. But even with such a system, there is usually a certain time discrepancy, and the additional effort of regular synchronization means that fewer important sensor data can be transmitted. [SWBT20]

To analyze the effects of different time discrepancies on machine learning models for the detection of ADL, the following **research questions** were developed:

1. How do time discrepancies impact the performance of machine learning-based human activity recognition?
2. Which temporal accuracy is required for accurate human activity recognition using machine learning models?

This research will yield insights into the relationship between time discrepancies and performance in HAR using machine learning. The outcomes will include a comprehensive understanding of how different time discrepancies affect different activities and machine learning models. It also provides recommendations for the temporal accuracy required for specific HAR tasks.

1.2 Methodology

There are only a few studies on the time discrepancies of time signals from wearable sensors. This thesis will investigate how time discrepancies, in general, affect the recognition of human activities and which maximum deviations between the sensor channels should not be exceeded. The aim is to analyze how time discrepancies impact recognition and the magnitude of these effects for different time discrepancies. The proposed research approach is to create an understanding of the current state of research, to develop a testbench on which different types of time discrepancies can be tested for their effects, and, based on these results, to derive insights into the required synchronization of signals from IMU sensors for HAR. This diploma thesis aims to address the existing research gap by conducting a systematic investigation of the impact of time discrepancies on machine learning models in HAR.

The first step will be to find ADL, which require several sensors for successful detection. For example, many ADL are performed similarly by one hand, then it depends on what the other hand is doing and whether the person is sitting, standing, or walking. Furthermore, an existing dataset must be found as a basis. This dataset must contain a wide range of body-worn sensors and also reflect many ADL. The activities mentioned above also depend on whether they are available in a dataset. The dataset must also be synchronized and well-labeled to be usable for training of machine learning models. A certain degree of synchronization is necessary in order to be able to apply controlled time discrepancies and to analyze the differences in the results between the synchronous original time signals and the intentionally modified time signals.

A testbench will then be developed with which various time discrepancies can be tested. The selected, synchronized dataset will be adapted for further processing to serve as suitable input for a machine learning model. Adaptation will include reducing the dataset to meaningful data that is useful for this work. For example, only sensor signals worn on the body will be used. Time discrepancies will then be applied to the individual signals of the adjusted dataset. If an IMU contains several sensors, it is considered a single unit. The terms IMU and sensor will be used synonymously for this one unit in the following work. It is assumed that there is no time discrepancy between the time signals of the accelerometer and the gyroscope, nor between the individual 3D axes of these, as they are always sent as one data block.

Different types of time discrepancies will be used. These vary in **type**, i.e. whether they involve a constant, temporal shift in the time domain or a continuous change of frequency in the frequency domain, in **magnitude**, i.e. different sizes of temporal shifts and frequency changes are applied, in **amount** and **distribution**, i.e. how often the time discrepancies are applied to the various signals and how they are distributed, and combinations of the different types.

These data streams will then be fed into a machine learning model as input. An existing and widely used model, which is primarily used for human activity recognition, shall serve as the machine learning model. However, it is not the aim of this diploma's thesis to develop a perfectly functioning machine learning model. Rather, the differences in the outcome will be analyzed and evaluated when various time discrepancies are applied. To ensure generalization, several models with different complexity and temporal length of the input will be used.

CHAPTER 2

Fundamental Background

The following chapter aims to provide a basis for a general understanding of the various complex topics discussed. It specifically deals with the meaning and origin of time discrepancy, how signals are sampled, and a general overview of important key features of the machine learning algorithms used.

2.1 Time Discrepancy

Time discrepancy is the difference or discrepancy between multiple time signals such as between two clocks. In everyday life, for example with a wristwatch, a time discrepancy of a few milliseconds or even a few seconds is irrelevant or probably not even noticeable to most people. However, even a small time deviation can have a negative effect on the results of multiple wearable systems. Particularly in systems where the number of wearable sensors changes dynamically, the time discrepancy must be as small as possible [MH15, Ma18]. Newly added devices can otherwise introduce an unknown time discrepancy and thus interfere with the entire system.

There are different types of time discrepancies between several data streams. The different effects of time discrepancies were analyzed by Xu et al. [XGMW17]. They propose two different models to describe time discrepancy and its distribution. In a case study [XGMW17], the authors demonstrated the usability of the two used models. A model with an adaptive frequency strategy based on the features of the data can guarantee the completeness of the data and also reduce redundancy. They emphasize that systems with different wearable devices should work with minimal time discrepancy to achieve the best possible results. [XGMW17, XGM17, Ma18]

Time discrepancies, in general, occur when the timestamps of data originating from different devices do not match, even though the data was theoretically collected simultaneously at exactly the same point in time. One of the reasons for this is that the different devices that send data use different variants for sending, which can be subdivided into two categories: static frequency and dynamic frequency. Static frequency means that the device transmits data packages according to a fixed schedule, while dynamic frequency indicates a dynamic change in the intervals between two data packages, and the timing of the next data collection depends on various factors. There are also various factors, such as clock drift, network latency, and processing delay, which also influence the time discrepancy. The impact of time discrepancy can be subdivided into four main aspects of data timestamps: timestamp scale inconsistency, timestamp frequency inconsistency, timestamp drift, and timestamp absence. Timestamp scale inconsistency means that different wearable devices provide timestamps at different timescales. For example, the devices are expected to provide data at exactly timestamp 13:10, precisely to the very minute. Device 1 provides data at timestamp 13:10, which has an accuracy of one minute. However, device 2 provides data at timestamp 13:10:05.125, which has an accuracy of one millisecond. The data appears as if it has been collected at the same time if the timestamp is considered down to the minute. However, if accuracy is required on the basis of seconds, milliseconds, or even more precisely, device 2 has a time discrepancy of 5 seconds and 125 milliseconds. Device 1 has an unknown time discrepancy, which may be minimal or non-existent but can also be close up to a minute as there is no more precise measurement of time. Depending on the requirements of a system, such a time discrepancy can cause difficulties and problems for the successful design of a system. [XGMW17]

Timestamp frequency inconsistency means that different devices send their data at different frequencies. For example, device 1 has a fixed frequency for sending data of 15 seconds. However, device 2 has a dynamic frequency and sends the data between 10 and 55 seconds. One possible reason is that transmitting the data is seen as a low-priority task by the processing unit since it is rather time-consuming and the device focuses more on the acquisition of new data. This makes it very difficult to get any data at the same time from all the devices. Most wearable devices generate their own timestamp with internal clocks. There are various reasons, such as temperature variations, why these clocks can be affected by clock drifts. Temperature changes can occur both inside the appliance and in the external environment. This clock drift leads to timestamp drift. Timestamp absence addresses the problem of wearable devices not having an absolute timestamp. For example, devices may have a relative timestamp beginning from their first initialization. A solution is to have an external server providing an absolute timestamp for the device. However, this results in an unknown time discrepancy consisting of network latency and processing delay. [XGMW17]

2.1.1 Clock Skew

The clock skew is a temporal shift of the original clock by a specific, constant factor. The shift of the clock has a direct effect on the timestamps of the device. As explained in Section 2.1, this phenomenon of time discrepancy is known as timestamp absence [XGMW17]. This shift can have various causes, including the dependency on other devices for the generation of timestamps. This results in a clock skew due to unknown network latency. Both a positive and negative clock skew can be seen in Figure 2.1.

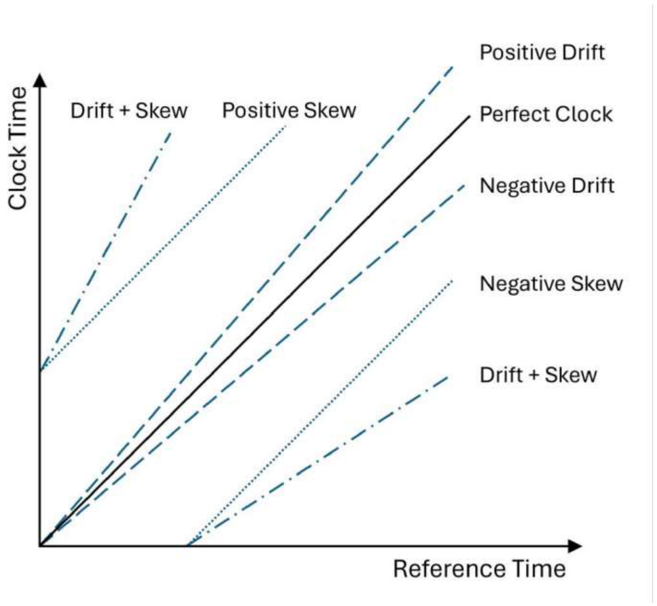


Figure 2.1: The following are the various types of clock inaccuracies: positive and negative clock drift, as well as positive and negative clock skew. It should be noted that any combination of these two inaccuracies also exists. Not all combinations are shown in this graphic to maintain clarity.

2.1.2 Clock Drift

The clock drift is a shift in the frequency domain of the original clock by a specific, constant factor. This drift also has a direct effect on the timestamps of the device. This phenomenon of time discrepancy is also known as timestamp drift [XGMW17]. The reasons for these shifts are usually inaccuracies in the built-in quartz oscillator but also external factors that affect the device. Both a positive and negative clock drift can be seen in Figure 2.1. The clock drift is also an additive error, which increases the error over time. This is particularly problematic with wearable devices, which are often worn continuously and are constantly operating and recording measurements. This additive error also leads to an additional clock skew. Therefore, combinations of the two time discrepancies are also shown in the figure.

2.1.3 Main Factors for Clock Inaccuracies

The clock inaccuracy of sensor signals is due to the large number of oscillators' instabilities. On the one hand, there are production-related inaccuracies and inaccuracies due to aging and noise, but also direct influences from environmental factors, such as frequency changes due to fluctuations in temperature, magnetic fields, acceleration, and supply

voltage [Vig92]. The frequency drift is a composition of the aging of the oscillator itself as well as the changes in the environment [Vig92]. The higher the required accuracy in the frequency range, both in terms of temperature stability and aging, the higher the production costs [Har20]. This is why manufacturers usually decide which oscillators to use depending on the application [Har20]. To reduce costs, just sufficient accuracy is often used for consumer products. Typical accuracies are highly dependent on the cost of the oscillators, but can be categorized as follows: $\pm 500\,ppm$ for complementary metal-oxide semiconductor (CMOS), $\pm 20\,ppm$ for external crystal oscillators (XO), and $\pm 5\,ppm$ for temperature-compensated crystal oscillators (TCXO) [TAA19].
The possible time synchronization of wireless sensor networks is heavily dependent on the selection of the clock used in the sensors. Solutions to prevent time discrepancies are available to some extent. However, research is dedicated to other, larger problems, as the effects of time discrepancy are usually assumed to be negligible, or are simply accepted because the effects are not directly recognizable. Furthermore, really exact synchronization in a sensor network tends to be rather complex. Various approaches focus on regular calibration of the devices to improve accuracy, on specially developed synchronization algorithms, or on the use of time servers, which are regularly queried to update the time and help with synchronization. The problem with time servers for obtaining an exact time is the unknown transmission rate. One solution is the use of Network Time Protocol (NTP). It allows devices to synchronize their clocks over a network. To be able to use these established products and protocols, either Internet access is required or a separate time server must be set up. Hardware solutions often involve the development of more accurate and higher-quality crystals for the system clock, but this usually makes the systems considerably more expensive. For this reason, compromises are made in order to accept a slight time delay and, at the same time, keep costs low and keep the networks open for dynamic connections from a wide variety of sensors. [TAA19, XGMW17, XGM17]

2.2 Resampling of Time Signals

Constant step size between individual data points is needed to train models. This makes it easier to find patterns and is even a prerequisite for many neural networks. Unintentional irregular sampling distorts the signal by increasing or decreasing the intervals between samples and thus changes the frequency components in the signal. Filters can only be applied to regularly sampled data. However, signals of real sensors in a real environment cannot always provide constant time steps. There are various reasons, including inaccurate sensor clocks, variable transmission times, and fluctuating processing times. [Hal20]

First, the time series is aligned to get fixed timestamps. If there are several values between two new points in time, there are various methods for generating the new values: mean, sum, min, max, last value, and next value. The method depends on the respective use case and the data used. It is therefore often useful to analyze the data in advance so that the characteristics and behaviors of the available data are known. [Hal20]

Then resampling, also known as sampling rate conversion, is used to change the effective sampling rate of a discrete signal sequence. The resampling of time signals consists of two main operations: increasing the sampling rate and decreasing the sampling rate. The increase of the sampling rate is also referred to as interpolation and consists of two steps. Zero values are inserted between the existing samples in order to increase the number of samples. The signal is therefore upsampled. The signal is then low-pass filtered to smooth it and remove the spectral images created by the upsampling. The sampling rate is then reduced. This step is referred to as decimation and also consists of two steps. The signal is low-pass filtered to limit the bandwidth and prevent aliasing effects. Samples are then discarded, which reduces both the number of samples and the sampling rate. By discarding samples, the signal is downsampled. Resampling can be linked to a certain data loss, as new data points need to be generated between existing data points, and existing data points need to be discarded. [Lyo11]

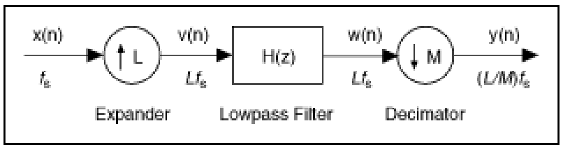


Figure 2.2: A rational resampling filter with a lowpass filter and the rational factor of L/M. [Nat23]

Resampling is necessary if direct sampling of a continuous signal with the desired sampling rate is not possible. For example, this is required for data from sensors with different sampling rates. The combination of upsampling and downsampling is used when a non-integer change in the sampling rate is required. Instead of changing the sampling rate with a single factor only, a desired rational number L/M can be implemented by interpolation with an integer factor L followed by decimation with an integer factor M. Since the ratio L/M can be obtained as exactly as desired with the correct choice of integers L and M, the sampling rates can be changed in practice with almost any factor. This process of resampling is also referred to as rational resampling. For example, an audio signal with 48 kHz can be resampled to 44.1 kHz to be used for audio CDs. Here, the factors for upsampling are L=147, and for downsampling M=160, so the signal is downsampled to a lower frequency than the sampling frequency of the original signal since M > L. An illustration of a rational resampling filter is shown in Figure 2.2. Between the expander and the decimator is a lowpass finite impulse response (FIR) filter. The length of the filter depends on $max(L, M)//gcd(L, M)$ [Sci25], where gcd is the greatest common divider. The filter is used for anti-aliasing. [Lyo11, Nat23]

2.3 Machine Learning

Machine learning is a rapidly advancing field in computer science characterized by algorithms that simulate human intelligence through learning from environmental data. These algorithms use input data to perform specific tasks without explicit programming for predetermined outcomes. Instead, they adjust their structure through iterative processes to improve task performance. This adjustment process, called training, involves providing input data along with the corresponding desired results, allowing the algorithm to optimize itself to achieve these goals and generalize findings to new, previously unseen data. [ENM15, Bel22]

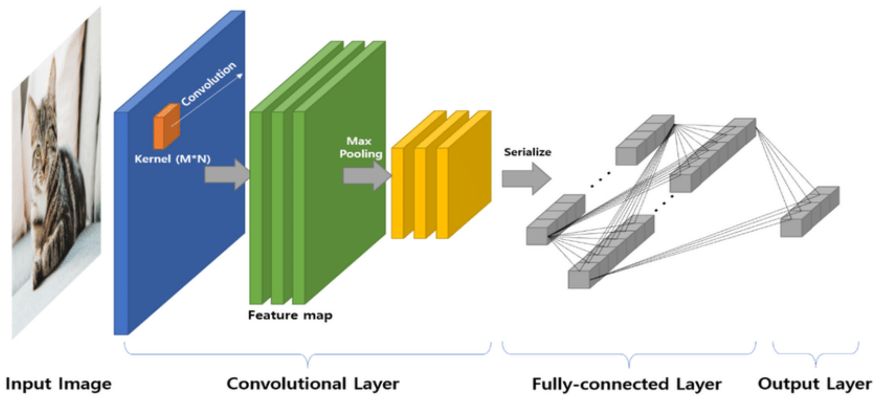


Figure 2.3: General structure of a 2D convolutional neural network used for image classification. [KGH20]

2.3.1 Convolutional Neural Networks

Generally, convolutional neural networks are used for the classification of images and object recognition or detection based on an image input. In feature engineering, features are extracted by convolutional layers with multiple kernels. In Figure 2.3, a kernel with $M \times N$ type is used. This figure shows the general structure of a 2D convolutional neural network. The extracted features are processed into vectors after multiple convolutions are performed. The fully connected layer receives these vectors, which are used as input for iterative learning. Through this learning process, a machine learning model can derive a classification result, which is emitted in the output layer. In Figure 2.4, a convolution neural network for one-dimensional data is shown. Instead of a kernel with $M \times N$ type, a $1 \times N$ type kernel is used here since the input is in the form of a vector. The feature maps are also extracted as 1D data. The fully-connected layers have the same structure as the general convolutional neural network. [KGH20]

Layers serve as fundamental building blocks. A layer typically receives weighted input data, applies a set of transformations, and forwards the resulting output to subsequent layers. To make them easily comparable within the network, layers are typically uniform, employing a consistent type of activation function, convolution, pooling operation, or other transformation. The initial and final layers are designated input and output layers, while those between are referred to as hidden layers. Convolution is a mathematical operation that defines a rule for combining two pieces of information or functions. The input data, or feature map, is convolved with the convolution kernel, resulting in a transformed feature map. The kernel extracts specific features from the input data and

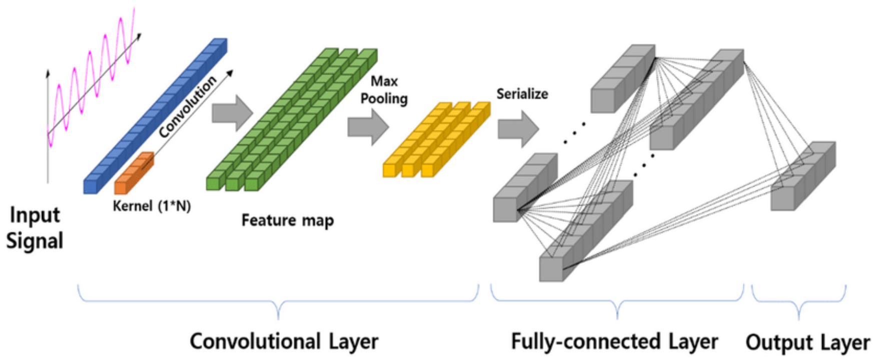


Figure 2.4: General structure of a 1D convolutional neural network used for signal pattern recognition. [KGH20]

is often interpreted as a filtering mechanism. Convolution serves as a bridge between the spatial/time domain and frequency domain. It can also be used to model the diffusion of information. While various interpretations exist in which areas convolution can be used, cross-correlation interpretation, which views convolutional filters as feature detectors, is currently the most practical for deep learning. The pooling layers reduce the size of the data streams. Nearby values are combined into one single representative value. Convolutional neural networks use multiple convolutional layers. These layers have learnable parameters that are adjusted automatically to filter input for the most useful information and extract this information. Multiple layers are usually used to filter the input data for more abstract information in each layer. [Kim17, ON15, SW11]

2.3.2 Evaluation Metrics

Several evaluation metrics are used to compare machine learning models and assess their performance. These are defined below.

The accuracy indicates the number of correctly classified samples over the total number of samples. The correct classifications are the combination of the true positives and the true negatives. A perfect model has zero false positives and zero false negatives and thus has an accuracy of 100%. Accuracy is one of the default evaluation parameters for evaluating models. However, it is only meaningful if all classes are equally distributed across the data set. If a class only occurs in a small percentage of the dataset and is always predicted incorrectly, the accuracy would still be high. [SW11]

The edit distance, also known as the Levenshtein distance, generally measures the minimum number of operations required to transform one sequence of characters into another. The smaller the edit distance, the more similar the two sequences are. In the context of HAR, sequences of sensor data, and therefore motion sequences are considered here, rather than sequences of characters. The lower the edit distance, the more similar the two movements are. The edit distance is therefore used to quantify the similarity of two movement sequences and thus helps to classify activities. [DSY24]

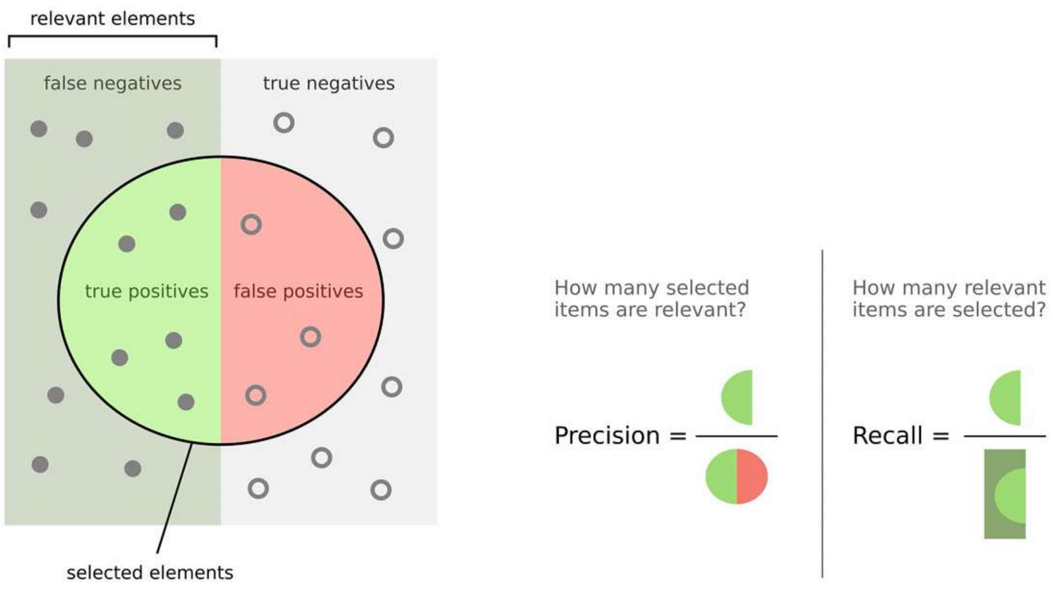


Figure 2.5: Visualization of precision and recall. [KM21]

Precision measures how accurate the positive predictions of a model are. It is the ratio of true positive predictions over the total number of positive predictions. Recall measures how effective a model is in identifying all relevant instances of a dataset. More specifically, it is the ratio of true positive predictions over the sum of true positive and false negative predictions. Both the precision and the recall of the models are important metrics for evaluating models. The F1-score combines both of these metrics. It is the harmonic mean of precision and recall and thus evaluates the overall performance of a machine learning model. The F1-score can take values between 0 and 1. It is made up of the ability to correctly classify the positive examples, i.e., precision, and the ability to find all positive examples, i.e., recall. A high F1-score indicates that a model has both a high precision and a high recall. Therefore, the model classifies a few false positives on the one hand and a few false negatives on the other hand. With a low F1-score, either many false positive samples or many false negative samples are classified. [SW11]

Using the threshold value k, the F1-score can be refined to determine when a value is classified as positive. F1@0.1 is a low threshold value and therefore more instances are classified as positive. This can increase the recall but also reduce the precision. In contrast, F1@0.5 uses a high threshold value. As only very reliable predictions are classified as positive here, the precision is higher. However, this may also reduce the recall, as many positive values are not recognized. Further analyses of false positives and false negatives are also carried out using confusion matrices. [DSY24, ZVVC22]

2.4 Activity Recognition

Activity recognition aims to automatically recognize and classify activities carried out within a specified time interval. Human activity recognition is often used to recognize ADL. Various sensors, including IMUs and cameras, are used to track the activities. Convolutional neural networks are used to classify the activities. [NTAgA18, ZVVC22]

2.5 Activity Segmentation

Activity segmentation aims to divide a continuous activity sequence into individual, homogeneous segments and assign pre-defined labels to each of these segments. It can predict which action class is taking place at each frame of a video or at each time step of a sensor signal. A signal can, for example, be split up based on temporal changes in sensor data. As a prerequisite, activity segmentation generally uses activity recognition, which detects which activity is taking place, while activity segmentation finds out the exact time when the activity takes place. [DX17, vAKLS23] An example of the interaction, or rather an extension of the two methods, is that activity recognition can use accelerometer data to recognize that a person is walking. Activity segmentation can then be used to recognize when the person started running, stopped, and took a break. Many studies use temporal convolutional networks to classify the input data [FG19, LFV$^+$16, LT18].

The main difference between activity recognition and activity segmentation is that activity recognition classifies the activity, while activity segmentation recognizes the exact start and end time of an activity in a continuous data stream. Therefore, activity segmentation determines when the activities actually take place. [ZVVC22]

CHAPTER 3

Related Work

The following chapter focuses on the review of related work and literature on the topics of synchronization in general, synchronization methods, and HAR especially using body-worn sensors. Papers on HAR datasets and Temporal Convolution Networks are also analyzed.

3.1 Synchronization of Body-Worn Sensors

David Bannach et al. [BAL09] have developed an automatic event-based synchronization method for several wearable sensors. They point out the importance of synchronization since asynchronous systems can run further and further apart in time, which results in poor activity recognition. This means that many actions that would otherwise be recognized can no longer be detected. The other problems and origins of non-synchronization in wearable devices are also discussed. Wearable devices aim to be very lightweight and have a long battery life. By removing receiving elements in the devices, both can be achieved, but this makes it difficult to synchronize the device. In the experiments carried out, a synchronization error of 0.3 s was achieved with their developed event-based synchronization. They conclude that good synchronization is a key element for good performance and that it is not trivial to achieve high synchronization performance in real applications. [BAL09]

Wang et al. [WSL+19] have also developed an approach for the clock synchronization of multiple, independent sensors. They focused on a hybrid approach based on NTP and context markers. They also point out that synchronization is one of the biggest challenges in detecting human activity in multiple sensor systems. The approach consists of three separate steps. First, the time drift of the receiver was recorded for 15 hours using NTP. This requires internet access. Then, offline, context markers were created by hitting the fist against a table and in the last step the context markers were used to align several sensor's clocks with each other and by using the NTP recordings the synchronization of the clocks is maintained over longer periods of time. The hybrid approach reduced the timing errors to 20 ms over a duration of 15 hours. The NTP alone had a synchronization error of 250 ms and the event-based approach had a synchronization error of 1310 ms. [WSL+19]

There are many more synchronization approaches when using wireless IMU sensors. All of them try to limit the time discrepancy between the sensors. Gilbert et al. [GLD+24] use a magnetometer-based method to synchronize four to eight IMUs with built-in magnetometers. With an electromagnet, an external, electromagnetic pulse is generated. The magnetometer of the single IMUs detects and measures the pulse and therefore, is able to apply some synchronization of the clock. Additionally, they also used a hard gyroscopic kinetic event to be able to align the data manually, similar to the approach of Bannach [BAL09]. Other approaches with magnetometers require either fixed distances to the pulse generator or the removal of the sensors and placing them in a box to synchronize them. In particular, placing them in a box limits the sensors, as only a certain device size is possible. The process is also very time-consuming, especially with many sensors. With this approach, the sensors to be synchronized only need to be within the range of the portable electromagnetic pulse generator, but they can remain on their attachments and do not need to be plugged into something. It was shown that synchronization with a maximum error of 36 ms could be achieved with this aligning method. [GLD+24]

3.2 Human Activity Recognition with Body-Worn Sensors

Bulling et al. [BBS14] describe the concept of the Activity Recognition Chain (ARC), as well as the research challenges that HAR shares with general pattern recognition. They argue that multiple sensors help to better recognize multiple daily activities. However, the positioning of the sensors is also crucial. Sensors on the legs are of little help in recognizing the movement of the arms. Furthermore, they state that there is no definition of human activities that would help to formulate a problem statement. Many activities in behavioral monitoring cannot be defined beforehand, since the execution of an activity is very different among different users. Therefore, sensors and their placement have to be selected carefully to design person-independent features and achieve satisfactory results. An ARC includes steps for data acquisition, signal pre-processing, and segmentation. First, the raw signals are processed and divided into segments. This is followed by feature extraction and selection, where the most important features are extracted from the previously created segments. Finally, training and classification are carried out using the selected features. Another challenge is also stated, which is the annotation of training data with ground truth labels. This is a rather expensive task since it is either annotated in real time [BWG12] or everything is labelled manually in a very time- and resource-expensive post process. [BBS14]

For a successful HAR, multiple sensors are needed to track the movement of a person and therefore be able to identify an activity. A lot of research uses acceleration data from wearable sensors. A sampling frequency of at least 20 Hz is required to identify

ADL correctly [BKV+97, BI04]. Piezoresistive accelerometers with a resonant frequency f_0 of 1200 Hz and frequency response of 0-600 Hz were used by Bouten et al. [BKV+97] to accurately measure frequencies of different human body movements. The resonant frequency of an accelerometer is the frequency at which the system reacts most sensitively to external vibration. Even small vibrations or accelerations can lead to large spikes in the accelerometer's output signal. [May22] Therefore, all frequency components should be well below f_0. A sampling frequency f_s of 76.25 Hz was used by Bao et al. [BI04] to recognize activities using a decision tree classifier, which was more than sufficient. They also mention that there are minor variations in the sampling frequency based on the clock accuracy. Ling Bao et al. from 2004 suggests that the problem of clock inaccuracy and resulting changes in sampling frequency is already known. However, to the author's knowledge, there exists no relevant research on how time discrepancy, specifically triggered by frequency changes, affects various types of machine learning or decision tree classifiers. Although, there is a lot of research being done on the synchronization of wearable sensors.

The two studies carried out by Jay Chen et al. [CKC+05] and Won-Jae Yi et al. [YSMS14] show how body-worn sensors can be used in everyday life in special areas of HAR. Both deal with the automatic and reliable detection of falls using acceleration sensors. The detection of a fall is based on a combination of a sudden sharp increase in acceleration data with an abrupt stop, which indicates the impact on the floor, and gyroscope data, which notices a change in orientation from an upright, standing position to a lying position. The focus here was not only on the sole detection of this special form of ADL, but also on the avoidance of false alarms, i.e. the avoidance of false positives, for example if the person only sits down. The second [YSMS14] also connects additional sensors, such as its own temperature sensors and an ECG sensor, to form a larger sensor network that is controlled by the smartphone. All the external sensors are connected via Bluetooth. In both studies there are several wirelessly connected sensors, but neither addresses the issue of synchronization or poor accuracy that could be caused by non-synchronization of the sensors. [CKC+05, YSMS14]

Daniel Roggen et al. [RCR+10] and Lukowics et al. [LPB+10] describe the implementation and approach for the Opportunity dataset [RCND+10]. They used 15 networked sensor systems with 72 individual sensors. Among other things, 7 IMUs and 12 3D acceleration sensors were fixed to the participants' bodies. In addition, there are various object sensors and environment sensors. A motion jacket with acceleration, gyroscope, and magnetic field sensors is attached to the upper body of the subjects. Multiple wireless microphones are placed on different body positions as well. Around 25 hours were recorded from 12 participants for the dataset. ADL are categorized into different levels of activities. Around 30.000 individual actions are annotated on the low level, such as opening a drawer or picking up a knife. The highest level has around 200 actions, such as getting up or preparing breakfast. There are also other ADL, such as: preparing a coffee, drinking a coffee, preparing a sandwich, opening a door, or unloading the dishwasher. A temporal decomposition of the activities can be seen in Figure 3.1. [RCR+10, LPB+10]

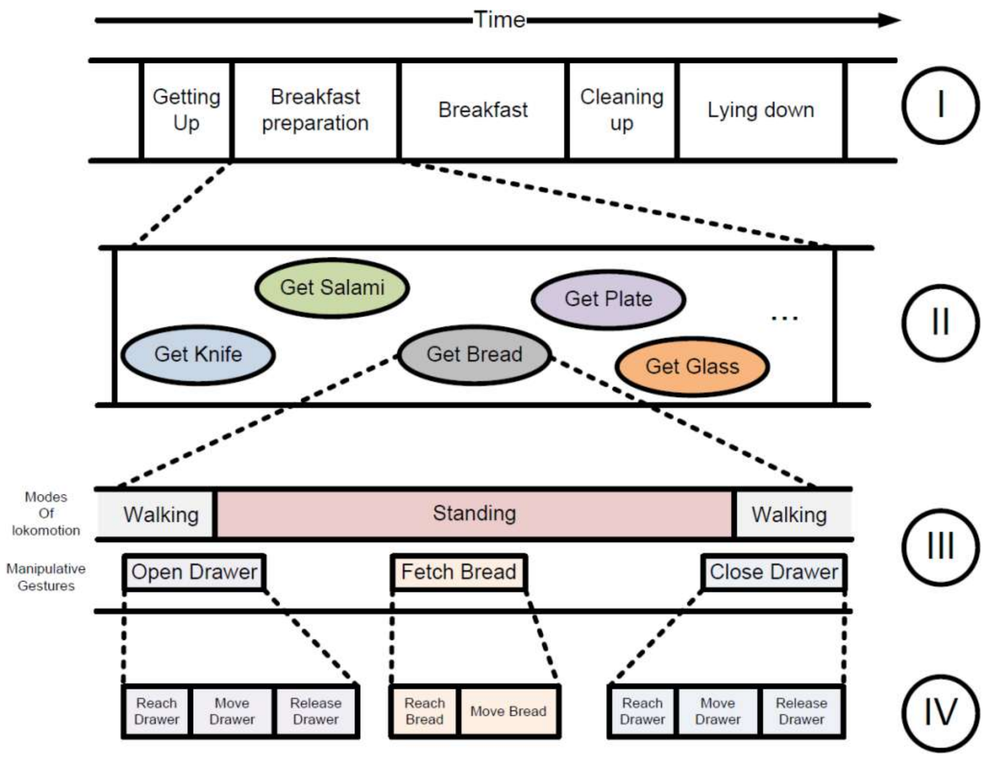


Figure 3.1: Temporal decomposition of activities. [LPB+10]

3.3 Machine Learning Methods in Context of Human Activity Recognition

Marius Bock et al. [BHMVL21] demonstrate a basic concept and best practices to create a HAR system with deep learning. The Deep Learning Activity Recognition Chain (DL-ARC) is presented and explained. It consists of four main parts: data collection and data analysis, followed by preprocessing, neural network architecture and training, and finally validation and testing. The data collection can be done in various ways, the other sections stick to some specific pattern. The preprocessing deals with various issues in the data, like missing values, which can be solved by interpolating the existing data points, or different sampling rates. The network must then be sufficiently trained with the training data. Finally, the results have to be validated and tested. Here it is also important to check whether there is an overfitting [SW11], which is an undesirable behavior in machine learning. It occurs when the machine learning model provides accurate predictions for training data but not for new data. This effect appears when the model cannot be generalized and fits too closely to the training dataset. Overfitting can occur for various reasons. The size of the training data is too small and does not contain enough data samples to accurately represent all possible input data values, or the training data contains large amounts of irrelevant pieces of information, known as noisy data. One solution to avoid overfitting is to stop early. This involves interrupting the training phase before the machine learning model learns the noise in the data. [BHMVL21]

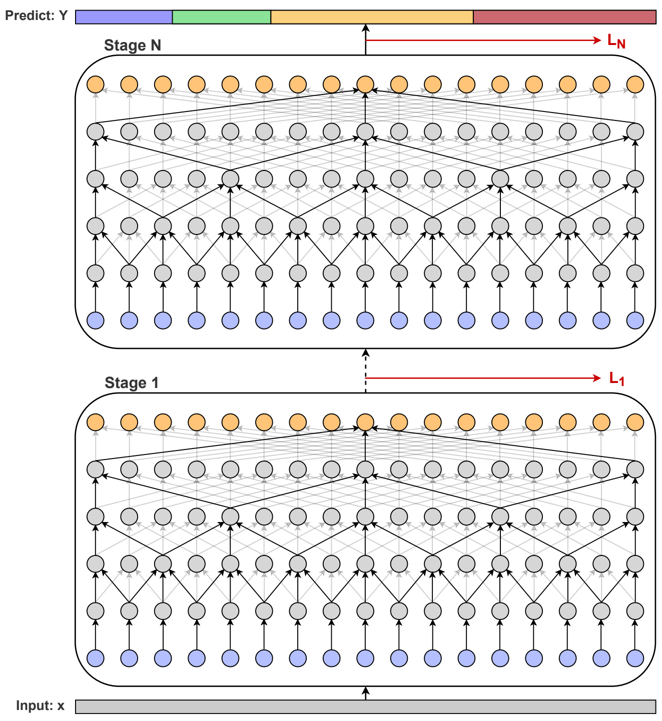
Anna Ferrari et al. [FMMN21] have also presented the pipeline and techniques used for successful HAR using data segmentation. Input data is first adapted using preprocessing and then segmented with different types before the features are extracted from it to finally make a classification of the HAR performed. In data segmentation, the data is divided into windows. There are different approaches for the windows. Sliding windows have a fixed size and there is no gap between the windows used, i.e. all possible data samples are utilized. However, overlapping can be used with sliding windows. Another possibility is activity-defined windows, where windows are placed on detected activity changes. The third option is event-defined windows. With these, the windows are placed around detected events. The overlapping of sliding windows has the advantage that noise at both ends of the window can be reduced. [FMMN21]

Understanding human activities and being able to track them is a main topic in ubiquitous computing. To be able to track and categorize different activities on body-worn IMU sensors are used. Xu et al. [XZSS12] categorized 14 different human activities with sensors on various body locations. They chose a more independent approach. Most sensors in other HAR systems are attached to specific body positions. Here, however, the sensor position is recognized using a developed framework. The same framework is also able to recognize the activity being performed. This means that each sensor can be attached to one of the seven defined positions on the body and therefore it is not necessary to check, for example, whether sensor x is really on the right wrist. [XZSS12]

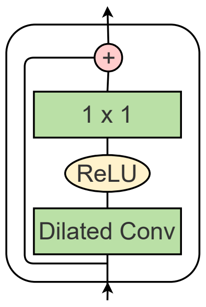
The MS-TCN for Action Segmentation is able to classify action segments in untrimmed videos as well as in time series data. Previous approaches for this segmentation used sliding window approaches [RAAS12, KSB14, OVS14] or hybrid approaches using Markov models [KGS15, LRVH16, RKG17]. They both achieve results with high accuracy but both are rather slow [FG19]. The aim is to derive each image's class label from the video images. An overview of the MS-TCN is shown in Figure 3.2a. At each stage, an initial prediction is generated, which is then refined by the subsequent stage. At each stage, a series of dilated one-dimensional convolutions are applied to the activations of the preceding layer. Subsequently, a loss layer is incorporated following the completion of each stage. Figure 3.2b shows a single convolutional layer with ReLU activation. [FG19]

A suitable temporal length or the length of the window, which is used as input for the convolutional neural network, is important to obtain useful results. The length is different depending on the type of input, and also depending on the activities that are to be recognized. For very slow activities that are carried out over more than 20 seconds, a temporal length of less than one second will probably not provide the best results. In the following paragraphs, some papers that use the MS-TCN are analyzed for their temporal length and purpose.

For activity recognition of ADL using acceleration data, such as walking, a 512 sample window was used at a sampling frequency of 76.25 Hz resulting in a window of 6.7 seconds. [BI04]



(a) Overview of the MS-TCN.



(b) Overview of a single layer in the MS-TCN.

Figure 3.2: MS-TCN. [FG19]

For activity segmentation with MS-TCN [FG19] of ADL of older persons using IMUs, a window length of 2 seconds at 250 Hz (used for a modified physical performance test (mPPT)) up to 16 seconds at 32 Hz (used for bathroom activities) was used. This results in a temporal window size of 500 to 512 samples. [ZVVC22]

In another study on ADL with MS-TCN, different temporal windows of 3, 7, 17, and 31 frames were used. For the datasets 50 Salads [SM13] and GTEA [FRR11], 31 frames provided the best results and 3 the worst. Larger temporal window sizes were also tried, resulting in much higher computational costs. [KHA+22]

Several window sizes of 1, 2, 4 seconds at a sampling frequency of the IMUs of 64 Hz, i.e. 64, 128, 256 samples, were used to train the models during gait recognition for people with Parkinson's disease. Finally, 256 samples were used as the optimal sample size. [YFG+24a]

An overview of various temporal lengths used in other papers is shown in Table 3.1. Not only a reasonable choice of temporal length for the network is important, but also the parameters, that influence the properties of the network and also change the number of trainable parameters, and so also the complexity of the network itself. The original MS-TCN [FG19] uses 4 stages, 10 layers, 64 feature maps, 50 epochs, and a learning rate of 0.0005. The datasets 50 Salads [SM13], GTEA [FRR11] and Breakfast [KAS14] are

Reference	Sensor Type	Application	Fs	Window Length	Window Samples
[BI04]	IMU	ADL	76.25 Hz	6.7 s	512
[ZVVC22]	IMU	mPPT	250 Hz	2 s	500
		Bathroom Activities	32 Hz	16 s	512
[KHA+22]	IMU & Video	ADL			31
[YFG+24a]	IMU	Gait Recognition	64 Hz	4 s	256

Table 3.1: Listing of several works that use different sampling frequencies and window lengths, also referred to as temporal lengths.

applied as input here. Only video signals are used as input to the network. [FG19] Most of the other research also uses the same learning rate, but the remaining parameters vary.

The MS-TCN is also often used without specifying the used parameters and settings in detail [DSY24, KHA+22]. Therefore, only the standard parameters can be assumed. Also the so-called “vanilla” MS-TCN [SYN+23] is used, which can be interpreted as the standard MS-TCN comprising 4 stages, 10 layers, and 64 feature maps. Different networks are often compared with regard to their accuracy and F1 values, but the properties and accuracy of the networks also depend on their complexity and characteristics.

Two separate MS-TCNs are used in a study on eating and drinking habits. The first MS-TCN is responsible for video features and has (N x 3072) features. The second MS-TCN is responsible for IMU features and has (N x 18) features. The IMU data is composed of 3 IMU sensors with 6 features each. Both MS-TCNs use 4 stages and 10 layers with 64 nodes. The learning rate is 0.0005. The IMU data was recorded at 100 Hz and the videos at 30 FPS. [KNN23]

The drinking behavior of people was investigated using two IMUs with 6 features each. The sensor signals have a sampling frequency of 64 Hz. However, these were downsampled to 16 Hz, as otherwise there would have been a lot of redundancy and unnecessarily high computation costs. The MS-TCN was adapted to 2 stages, 9 layers, and 128 filters per layer with a learning rate of 0.0005. The receptive field in this case was 1023 sample points. [WKDR+22]

To identify jumps in volleyball, an IMU with a sampling frequency of 100 Hz and a camera with 20 fps were used, which were fed into the MS-TCN as input. The number of layers was set to 12, as volleyball jumps are short activities. The temporal length of the input is 40 seconds. The length of the input should always be longer than the receptive field. One to five stages were trained to compare the results, with four stages providing the best results. [SDBV+23]

Different ADLs were trained with the MS-TCN for more than 100 iterations using several datasets with IMU data recording bathroom activities. The parameters were 256 filters, 12 layers at a sampling frequency of 100 Hz and 7 layers at a sampling frequency of

6.25 Hz, as well as 6 stages. This results in a receptive field of 81.91 seconds at 100 Hz and a receptive field of 40.8 seconds at 6.25 Hz. [ZVVC22]

Another action segmentation using video datasets uses 3 stages with 10 layers each with 64 features. For the 50 salads dataset, the model was trained for 100 epochs with a learning rate of 0.0005, and for the Breakfast dataset for 100 epochs with a learning rate of 0.0001. [vAKLS23]

An MS-TCN implementation for gait recognition in people with Parkinson's disease has 8 layers and a learning rate of 0.0005. The time signals of the 5 IMU sensors have a sampling frequency of 64 Hz and the videos 30 fps. [YFG+24a]

Usage	Parameter	Input-type	Dataset	Edit	Acc	F1 (@10,@20, @50)		
Original MS-TCN Implementation [FG19]	Stages: 4, Layers: 10, F-Maps: 64, Epochs: 50	Video	50Salads	67.9	80.7	76.3	74.0	64.5
			GTEA	79.0	76.3	85.8	83.4	69.8
			Breakfast	61.4	65.1	58.2	52.9	40.8
Eating and Drinking [KNN23]	2 MS-TCN (Video & IMU), Stages: 4, Layers: 10	Video & IMU	custom	67	83	75	72	60
Drinking Behavior [WKDR+22]	Stages: 2, Layers: 9	IMU	custom			95.6	95.5	94.3
Bathroom Activities [ZVVC22]	Stages: 6, Layers: 7-12	Video & IMU				81.6	80.5	70.5
Action Segmentation [vAKLS23]	Stages: 3, Layers: 10 F-Maps: 64, Epochs: 100	Video	50salads	85.8	90.3	86.4	90.4	91.3
Gait Assessment with Parkinson [YFG+24b]	Stages: 4, Layers: 8, Epochs: 50	IMU	custom				0.50	
Gait Recognition with Parkinson [YFG+24a]	Stages: 2, Layers: 8, Epochs: 50	IMU	custom [YFG+24b]				0.76	

Table 3.2: Listing of several papers that use the MS-TCN, with the parameters of the network and the results of it.

CHAPTER 4

Implementation

To address the identified, current research gap concerning the influence of time discrepancies on machine learning model performance in the literature, the next logical step is to design, develop, and implement a comprehensive experimental framework, which will be further referred to as a testbench. This testbench facilitates rigorous evaluation of the impact of temporal discrepancies on a widely utilized and representative machine learning model.

A synchronized dataset serves as the foundation for these investigations. This dataset is used to train the machine learning models under different conditions. It must be mentioned directly that the selected dataset was synchronized by means of post-processing. However, the time signals of the individual sensors used in this dataset are synchronized with each other, as they are connected to a separate computing unit. The dataset is also labeled in great detail, which makes it suitable for machine learning, and it is also used in many other papers.
Subsequently, a series of controlled experiments are conducted, wherein varying magnitude, distribution, and types of time discrepancies are systematically introduced into the dataset. These temporally changed datasets are then used to train the machine learning model in various runs.

The results obtained from the various machine learning runs are analyzed at the end in a comparative analysis to assess the extent to which time discrepancies impacted model performance. This analysis aims to clarify the specific influence of these impacts.
To enhance the generalizability of the findings and mitigate potential biases arising from the choice of a single model architecture, the experiments were repeated using a diverse set of machine learning models with varying levels of complexity.

4.1 Utilization of Opportunity Dataset

The Opportunity Activity Recognition Dataset [RCR+10, RCND+10], further referred to as Opportunity dataset, is a large and widely used dataset [BBS14, BMVL23, SM13, ZVVC22]. It contains a particularly large number of atomic activities, over 27,000. In total, 72 sensors in 15 wired and wireless network systems recorded movements and changes on the subjects' bodies, on objects, and in the environment. 12 subjects were used, providing over 25 hours of sensor data for ADL. The dataset not only contains these numerous sensor data, but the activities carried out are also labeled at different levels. There are labels for locomotion in which a distinction is made between stand, walk, sit, lie, and no data (nothing). Furthermore, there are high-level activities that describe the entire activity currently being carried out: relaxing, coffee time, early morning, cleanup, and sandwich time, and low and medium-level activities for smaller and shorter activities.

The dataset itself contains only four subjects with six runs each. The first five are ADL in a natural sequence. Each subject could move freely and decide for themselves when to do what. The sixth run is a "drill" run in which all activities are performed according

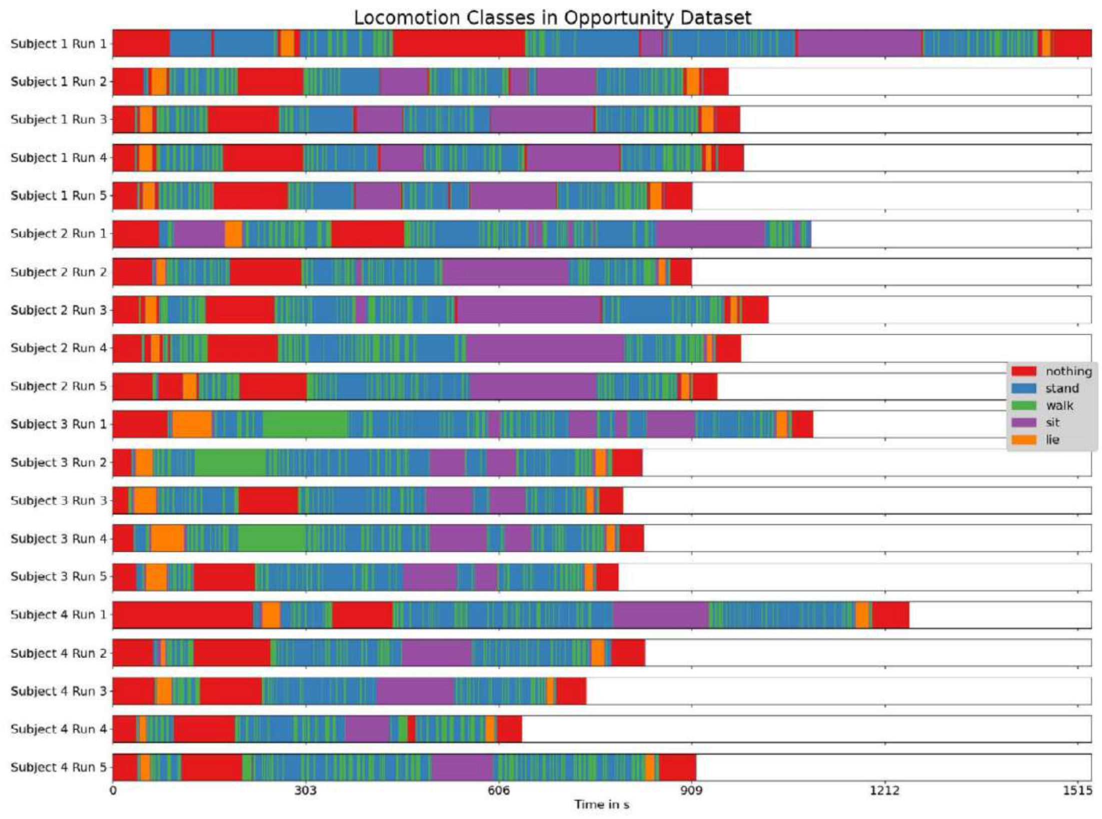


Figure 4.1: Visualization of the timeline of the various locomotion activities in the Opportunity dataset.

Subject	Run	total	stand	walk	sit	lie	nothing
1	1	50702	44.14%	12.9%	14.71%	2.22%	26.02%
1	2	31892	32.26%	20.42%	19.8%	4.37%	23.15%
1	3	32500	31.94%	18.55%	23.39%	3.98%	22.14%
1	4	32699	32.08%	19.63%	21.22%	3.06%	24.0%
1	5	30048	30.79%	17.45%	22.39%	4.05%	25.32%
2	1	36170	33.95%	19.54%	26.93%	2.47%	17.13%
2	2	30010	30.57%	19.46%	24.23%	2.74%	23.0%
2	3	33984	33.65%	18.59%	23.52%	2.69%	21.55%
2	4	32543	27.34%	22.72%	26.07%	2.27%	21.59%
2	5	31313	28.57%	20.13%	20.96%	3.7%	26.65%
3	1	36302	40.04%	26.37%	15.61%	7.01%	10.97%
3	2	27457	41.65%	30.11%	13.76%	5.17%	9.31%
3	3	26448	39.97%	17.15%	17.78%	5.72%	19.37%
3	4	27541	32.99%	32.75%	17.65%	7.84%	8.77%
3	5	26204	39.52%	17.1%	16.46%	5.72%	21.2%
4	1	41260	37.49%	15.85%	12.84%	3.88%	29.94%
4	2	27588	35.21%	18.23%	14.71%	3.18%	28.67%
4	3	24550	34.48%	15.78%	16.99%	4.53%	28.22%
4	4	21216	32.53%	22.93%	11.82%	3.71%	29.02%
4	5	30249	34.39%	30.25%	11.02%	3.08%	21.26%

Table 4.1: Total number of all occurring locomotion activities, as well as the percentage shares of the individual locomotions for all subjects and runs.

to a predefined script [RCR+10]. This run is not used. The sequence of the individual activities and also the length of the single activities is the same, as the script is the same for all subjects. This could lead to a bias of the machine learning algorithm, as the model adjusts to certain activity sequences.

The labels for locomotion are used. They are visualized for the individual runs of the four subjects over time in Figure 4.1. The percentage breakdown of the total occurrences of the individual locomotion activities is shown in Table 4.1, as well as the total number of samples. It is visible that the different runs in the dataset also have very different lengths. It can also be seen that there are clear differences in the occurrence of different activities. There is no equal distribution of the individual activities. For example, the locomotion activity “sit” is present once over a longer period of time, but otherwise only very rarely. In a visual analysis, it appears as if the “stand” activity is very dominant and is represented most frequently in every run, while the “lie” activity is only performed very rarely and there is therefore less data for this activity on which to train. This observation can be confirmed by the percentage breakdown of the individual activities in the table. A cumulative visualization of the locomotion activity occurrences is shown in Figure 4.2. This view provides further valuable insights into the time sequences of the activities in the individual runs of the dataset.

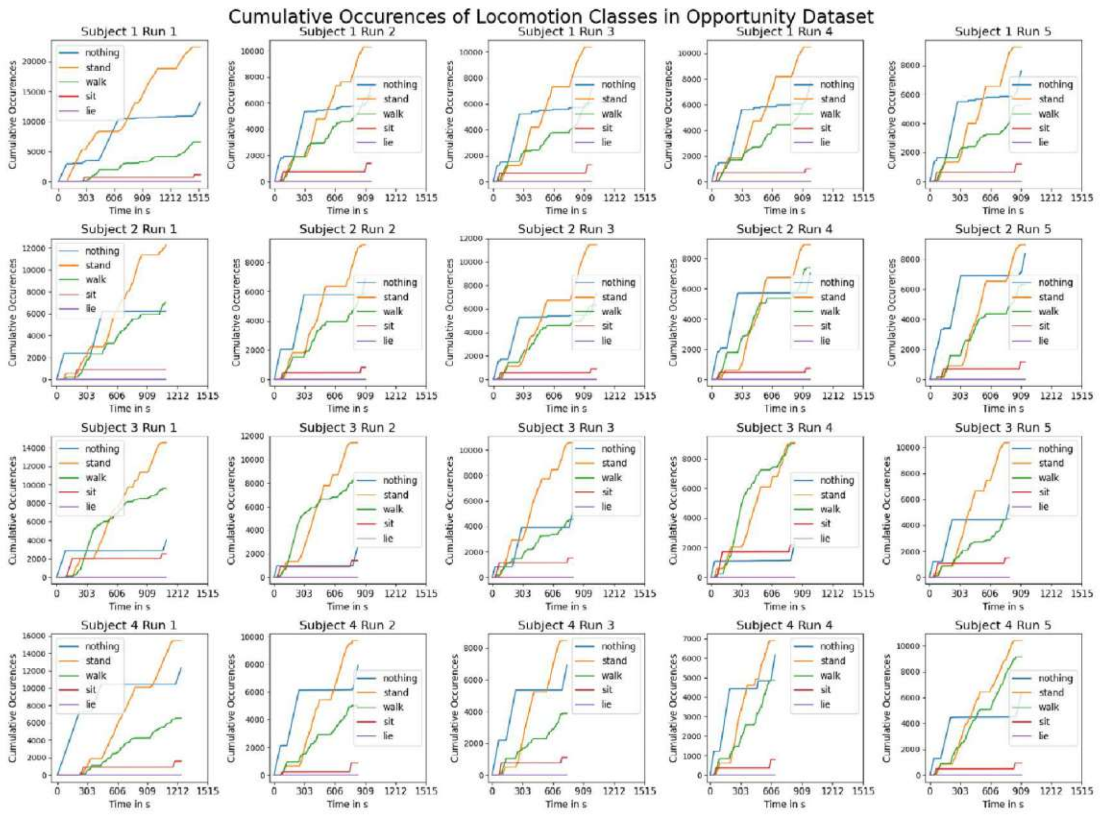


Figure 4.2: Visualization of the cumulative occurrences of the different locomotion activities over time in the different runs of the Opportunity dataset.

In order to make the Opportunity dataset usable for use in the testbench, it is adapted to the required input format for the machine learning model. From the dataset, 20 data files (.dat) are used: four subjects with five runs each. Each of these files has 255 columns. The first column is the timestamp in milliseconds. Columns 2 to 243 contain time data of various sensors, both body sensors and object and environment sensors. The last columns, 244 to 250, contain the labels for each timestamp. There are also files to explain the labels and columns. This means that a data file contains both the feature data and the ground truth data. However, the machine learning algorithm requires the features and ground truth data separately as input. In addition, splits are established and the data files are divided into test data and training data. Furthermore, a mapping is required to establish a connection between the labels as an integer value and as a string value.

In a Python notebook, the original Opportunity dataset is adapted to an 'opportunityAdapted' dataset. The activities used are defined here, i.e. the values that the labels can assume. Additionally, the data files and the columns to be used are described. The drill files are skipped here.

The data files are read in and filtered according to the columns used. Some sensors provide NaN values during the recording of the runs. However, the sensors used always provide consistent values and each data point also has a label, which is why all data can be used. As no values are missing, there is no need to interpolate between missing values. For the last few data points of each run, all sensors provide NaN values, which is why these last values are discarded.

The sensor data used is saved as NumPy files (.npy). NumPy is a standard binary file format to store single NumPy arrays on a disk [Num25]. This file can be loaded directly by Python without the need for complex data conversion. The ground truth data is saved as text files (.txt). Four splits are created in order to be able to perform a cross-validation later. Since the data for testing the machine learning algorithm must be independent of the training data, all runs of a subject are always used either for training or testing.

This means that 15 files are used for training and 5 files for testing. This results in a training-to-test ratio of 75:25. The splits are numbered so that the split number is always the number of the subject that is used for testing, i.e. split one uses subject one for testing and subjects two to four for training.

Subsequently, a specific subset of the existing sensor channels is used. Since the focus of this thesis is on recognizing ADL using sensors worn on the body, to ensure the broadest possible generalization, only acceleration and gyroscope data will be used. These are the most common [ASZ17, DSL19, Lau07, HEKA21, BHV13].

The Opportunity dataset has also been synchronized using manual alignment. Ideally, there would be a large, labeled, synchronized dataset, but none could be found in relevant research, which is why the Opportunity dataset was chosen. However, some of the body-worn sensors are attached to a custom motion jacket [SRO+08]. This jacket has five RS485-networked XSense IMUs. Since these five sensors are not connected wirelessly like the others but via cable and are first collected together in a processing unit and then sent to a laptop, it can be assumed that these sensors are synchronized and therefore best suited for my work. [RCR+10, SRO+08]

This means that the following sensors are used, each with 3-axis acceleration and 3-axis gyroscope data. One of the sensors (`IMU BACK`, Column: 38-43) is located on the back of the subjects. Two others (`IMU RUA` Column: 51-56 and `IMU LUA`, Column: 77-82) are placed on the upper arms. The last two IMUs (`IMU RLA`, Column: 64-69 and `IMU LLA`, Column: 90-95) are located on the two lower arms.

This results in a total of 30 different data columns of the Opportunity dataset which are used. As a consequence, 30 different features can be utilized for the machine learning algorithm.

4.2 Time Discrepancies

The two different time discrepancies clock skew and clock drift are applied to the synchronized signals in order to generate controlled shifts in the time and frequency domain. This allows any combination of clock skews as well as clock drifts to be applied in order to test their different effects on the machine learning algorithm, if there are any and which kind of effects there are.

In the Fundamental Background, Section 2.1, four main aspects were mentioned which have an impact on the time discrepancy of signals. Although, timestamp scale inconsistency is problematic, as an unknown inaccuracy comes into play due to the different time bases, this can be regulated by certain device standards and regulations for wearable sensor systems and, therefore, is not considered further here. The same applies to timestamp frequency inconsistency. In timestamp absence, there are no absolute timestamps, but only relative timestamps. One solution is to request timestamps from another device, but in this case, there is an unknown offset in the timestamps caused by network latency and processing delay. This is one of the two time discrepancies, whose effects are analyzed in more detail and is referred to as "clock skew" in the following.
The timestamp drift results in shifts in the frequency range. This is the second time discrepancy which is analyzed in more detail. It is referred to below as "clock drift".

4.2.1 Clock Skew

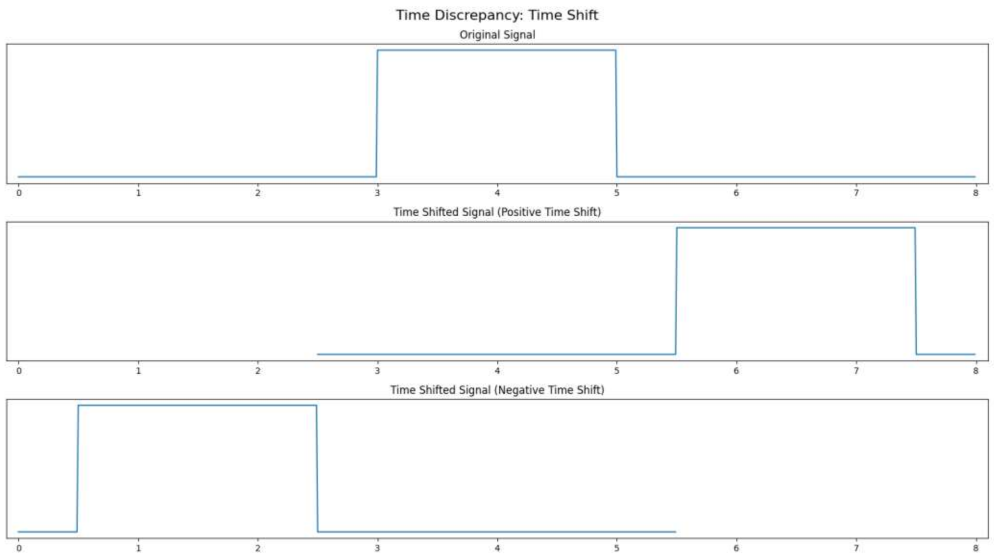


Figure 4.3: Abstract illustration of a positive and negative clock skew on a signal.

Since most body-worn sensors are connected wirelessly for practicality and convenience, there is often a network latency and processing delay. This leads to a constant time discrepancy and the signals from the different sensors are misaligned. In many applications, there is an unknown time discrepancy, which can lead to poorer performance. The clock skew and its implementation are explained in more detail below.

Due to various delays in a sensor system, a signal can be shifted by a constant, positive clock skew, see Figure 4.3. Under specific conditions a negative clock skew can also occur.

With a processor with several cores working in parallel, a signal can be sent earlier than that from other sensors, which handle the sending task later. If all, but one sensor in a multiple-sensor network, have a positive clock skew, this one sensor has a negative clock skew relative to the others. For this reason, a negative clock skew is also illustrated and applied as a clock skew in the subsequent sections.

In Figure 4.3 another aspect is visible as well. If a time segment is used, as shown in this example from 0 to 8, then a positive clock skew of 2.5 takes place. Then there is no or only invalid data from 0 to 2.5, as the recording of the data has not yet started here. The situation is similar to a negative clock skew, where there is no valid data in the range from 5.5 to 8. In this case, the time range must be restricted to ensure that only valid data is used.

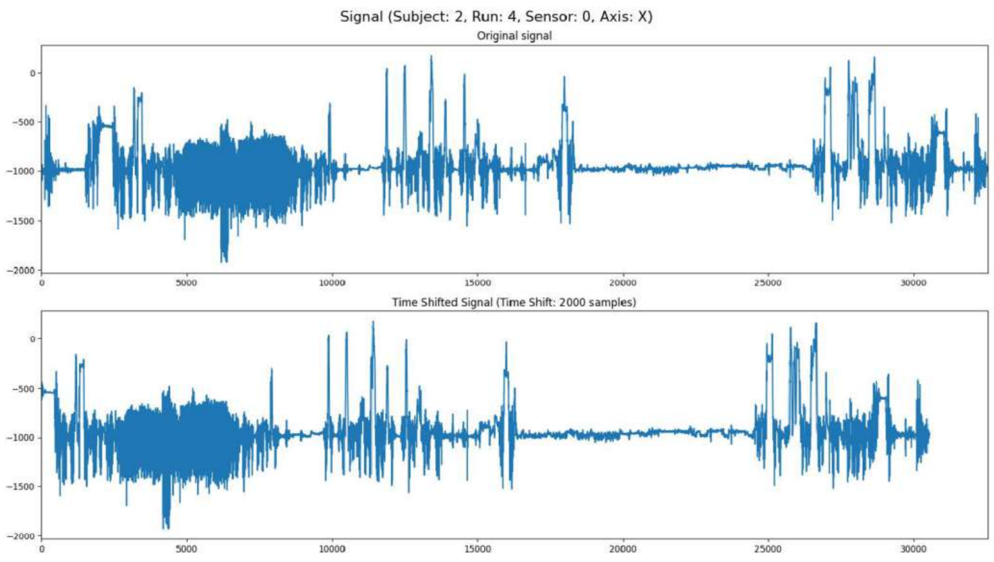


Figure 4.4: Visualization of the x-axis of sensor 0 from the entire run 4 of subject 2. The original time period is shifted by 2,000 samples due to an applied clock skew.

Since the Opportunity dataset contains sequences lasting up to 30 minutes and it does not make sense to use such long input data for the machine learning model, only small, random segments from each run are taken to train the model. If only the 15 files were used once, but as a whole, there would be very little training data on which the machine learning model can be trained. However, this individual data would be over 30 minutes long, which would be very challenging for the individual layers of the model to train on. Figure 4.4 shows the x-axis of sensor zero from the entire fourth run of the second subject. Here, a positive clock skew of 2,000 samples is applied. In Figure 4.5, a zoomed excerpt of the same signal is visualized so that the clock skew of the time section is better visible. For the training of the machine learning model, for example, these 2,000 sample long time periods are used. In order to be able to shift these sections without having no or only invalid data, further samples must be available after the section in the case of a positive

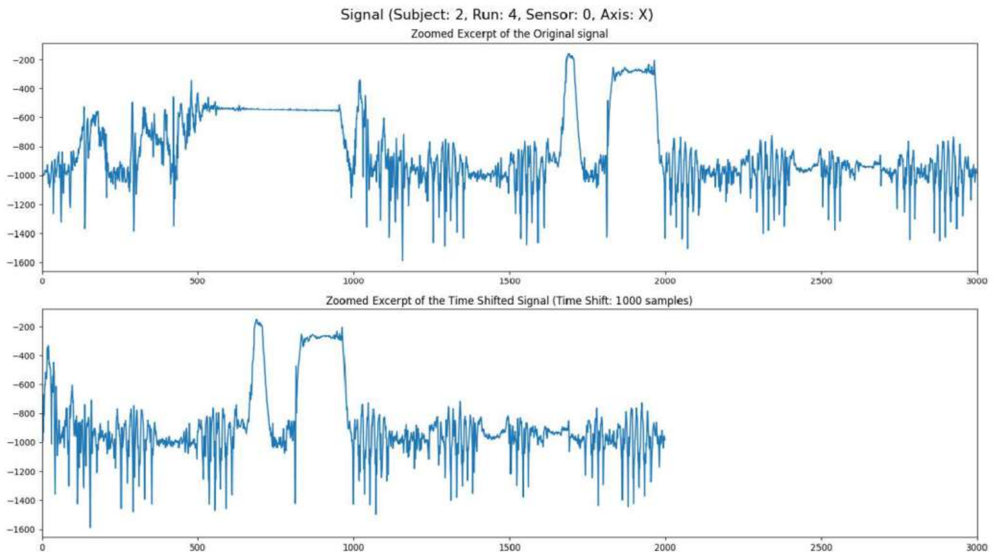


Figure 4.5: Visualization of the x-axis of sensor 0 from a zoomed excerpt of run 4 of subject 2. The original time period is shifted by 1,000 samples due to an applied clock skew.

clock skew. In the case of a negative clock skew, data must exist beforehand. This is also shown in the figures. If this were not the case, the time segment, and therefore also the temporal length used as input for the machine learning model, would be shortened. For model training, however, the same temporal length is always used. Depending on the size of the temporal length, the last few samples of the run cannot be used with a positive clock skew and the first few samples of the run cannot be used with a negative clock skew.

The clock skew is implemented using index shifting of NumPy arrays. For example, an array with temporal samples is shifted by plus 50 samples. The array indices are then shifted by 50 samples in the positive direction. In order to avoid changing the original array, an additional array is used which, however, has fewer array entries. It is important to note that the shifted signal has fewer samples based on the size of the shift. If a signal is recorded from time 0, there are no or only invalid values before this. With a positive clock skew, the signal is “shifted to the right”. The resulting gap could only be filled with values before 0, see also Figure 4.3. However, as there are no values there, the array of the shifted signal must be reduced in size in order to have only valid values. The opposite is true for a negative clock skew. If a signal is only recorded up to a certain point in time, there are no more valid values afterward. The signal is “shifted to the left”. As there are no valid values after the end of the signal, the array of the shifted signal must also be reduced by the size of the shift. For the reasons just mentioned, the initial array is always larger than the array at the end, which then only has a size equal to the temporal length and is, therefore, suitable as input for the machine learning model.

4.2.2 Clock Drift

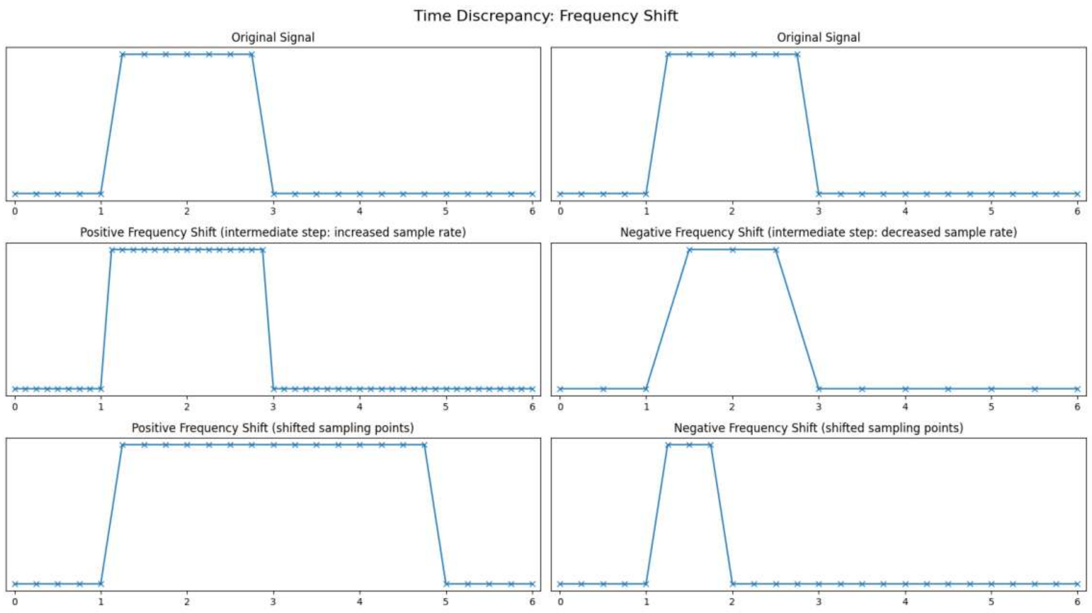


Figure 4.6: Abstract illustration of a positive and negative clock drift on a signal.

Most wearable devices generate their timestamp locally. The timestamp is generated using the device's clock. Quartz crystals are built into most devices, which have a certain variation due to environmental changes or their intrinsic properties [TAA19]. This leads to a time discrepancy in the data and the data from several devices are no longer synchronized. As the effects of a shift in the frequency range on machine learning algorithms are not known, the clock drift is examined in more detail here and explained below.

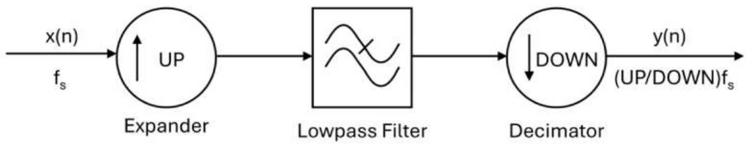


Figure 4.7: A rational resampling filter with a lowpass filter and the rational factor of *up/down*. c.f. [Nat23]

The clock drift within a sensor unit results in a drift in the timestamps. This shift in frequency can be either triggered by a positive factor or by a negative factor, see Figure 4.6. A positive clock drift is implemented as follows, see flowchart in Figure 4.7. The signal is upsampled by a factor *up*, then filtered and downsampled again by a factor *down*. This results in a new sample rate of *up/down* times the original sample rate. If the *up* factor is greater than the *down* factor, there is a positive clock drift, as the sampling rate is

increased. By increasing the sampling rate, there are more sampling points in the same period of time. However, these sampling points are then stretched back to the original width of the section. This then leads to an overall stretching of the signal.
If the factor *up* is smaller than the factor *down*, there is a negative clock drift, as the sampling rate is reduced. Due to the low sampling rate, there are fewer sampling points in the same period of time. However, as each sampling point has a specific timestamp, the sampling points are downsampled to these timestamps. Consequently, the signal is also downsampled.

In general, this means that a positive clock drift results in a stretching of the signal, see left column of Figure 4.6, and a negative clock drift results in a squeezing of the signal, see right column of Figure 4.6.

Figure 4.8 shows a section of the x-axis of sensor 0 from the fourth run of the second subject. A negative clock drift of 100,000 ppm is applied here. Clock inaccuracies are usually referred to as parts per million ($1\,ppm = 1 \cdot 10^{-6} = 0.000\,001$), as the frequency deviations of most crystals are in this order of magnitude. However, a very large shift was chosen here to better illustrate the clock drift. The 100,000 ppm corresponds to a factor of 0.1, which means that the original section of the signal is now reduced by a factor of 0.9, resulting in "more signal" in the same period of time.

The clock drift is implemented using Rational Resampling to convert the sampling frequency to another frequency, see Section 2.2. This method can achieve a non-integer

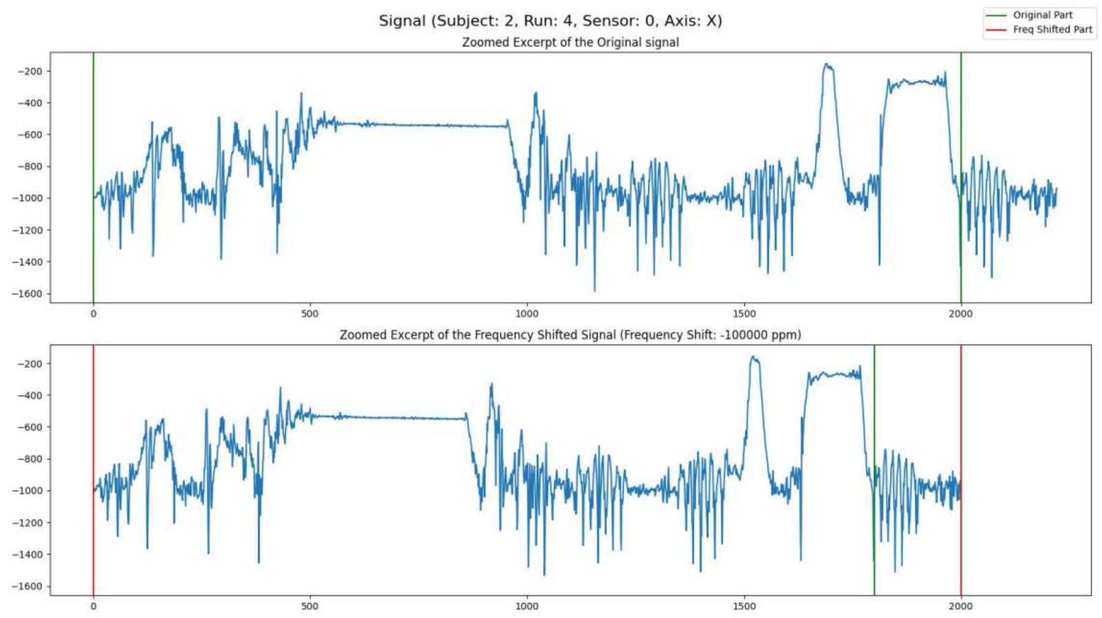
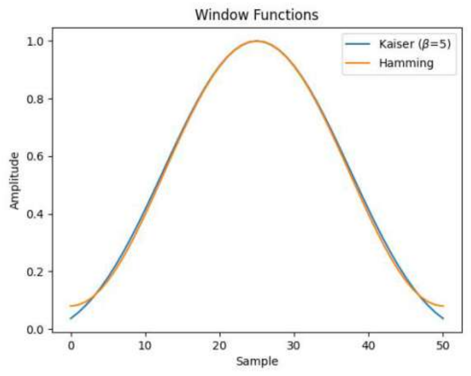


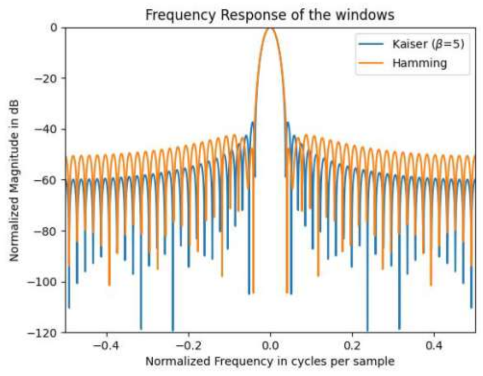
Figure 4.8: Visualization of the x-axis of sensor 0 from a zoomed excerpt of run 4 of subject 2. The original time section, marked green, is reduced by a factor of 0.1 (equals 100,000 ppm) due to an applied clock drift, marked red.

change in the sampling rate. This is done by first upsampling, filtering, and then downsampling. The *resample_poly* function from the *SciPy* library is used in the implementation.

If a signal is downsampled by a certain factor, it must also be taken into account that the shifted signal can have fewer values after the clock drift is applied. However, this is only the case with a negative clock drift. With a positive shift, when the signal is upsampled, sufficient values are available, or rather, there are even more values than needed. In this case, the array must be shortened to satisfy the expected temporal length of the machine learning model input. Therefore, with a negative shift, the length of the array must be larger at the start of the resampling since the data shrinks during the resampling process.



(a) The Kaiser window with $\beta = 5$ and the Hamming window.

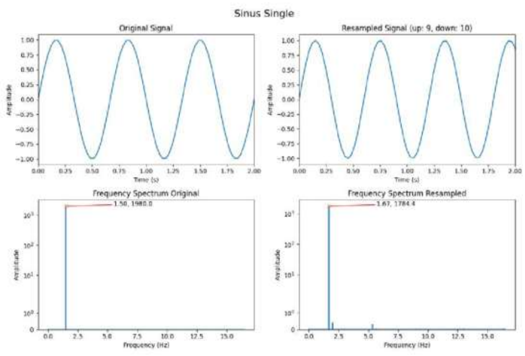


(b) The frequency response of both the Kaiser window with $\beta = 5$ and the Hamming window.

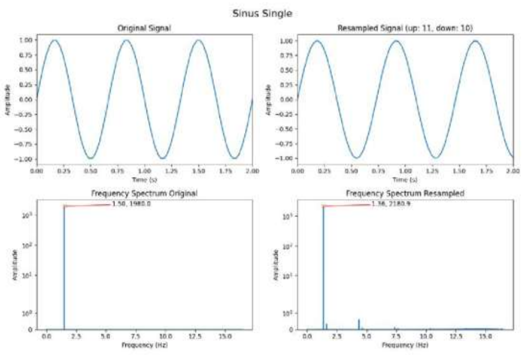
Figure 4.9: Comparison of the windows and the frequency response of Kaiser window with $\beta = 5$ and Hamming window.

SciPy stands for Scientific Python and is a scientific computation library. It uses *NumPy* and provides more utility functions, especially for signal processing (*scipy.signal*). The function *resample_poly* upsamples a signal x by a given factor, applies a zero-phase low-pass FIR filter, and then downsamples the filtered signal by a given factor. [Sci25] This results in a new sample rate of $upsample\_factor/downsample\_factor$ times the sample rate of the original signal.
The standard window for the filter design of the linear-phase low-pass FIR filter is the symmetrical Kaiser window with the shape parameter $\beta = 5$ [KK66]. The Kaiser window forms an almost optimal window using the 0^{th}-order modified Bessel function. It maximizes the energy in the main lobe and requires only two parameters: the length$(M+1)$ and the shape parameter β [OS10]. The Kaiser window with $\beta = 5$ is similar to a Hamming window [Sci25], see Figure 4.9. The length of the filter is $numtaps = 2{*}(10{*}max\_rate)+1$, where $max\_rate = max(upsample\_factor, downsample\_factor)$.

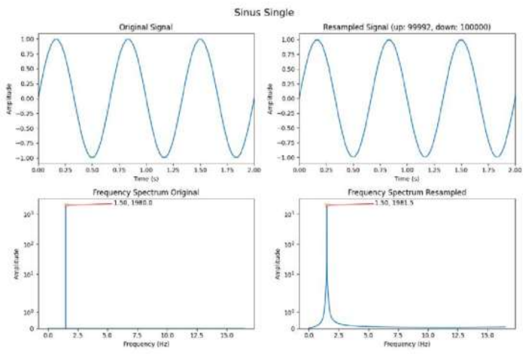
For example, a clock drift of -50 ppm is used. This results in the $upsample\_factor = 1,000,050$ and the $downsample\_factor = 1,000,000$. The clock drift should always be specified in ppm, as this is the most common form in which clock drifts are specified, especially for quartz oscillators, but also for entire systems. As only integer values can be used for the filter parameters, the upsample and downsample factors must be selected to be this large in order to achieve a change in ppm. In the example just mentioned, this results in a length of the Kaiser windows of 400.001 samples. Therefore, the signal is resampled with the help of the resample function using polyphase filtering. Resampling the signal results in the signal being either compressed or stretched. As a result, the signal is either given fewer or more samples, which also leads to more or fewer array indices. Therefore, the signal is trimmed to the correct length before it can be used.



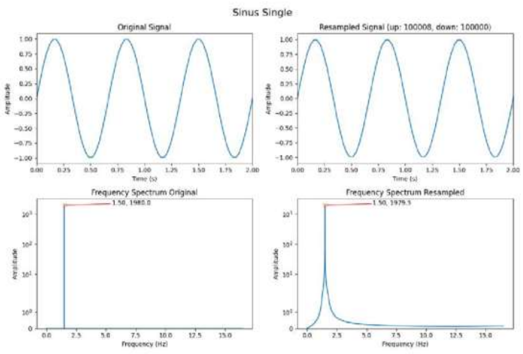
(a) Single sinus with upsample factor = 9 and downsample factor = 10



(b) Single sinus with upsample factor = 11 and downsample factor = 10



(c) Single sinus with upsample factor = 99,992 and downsample factor = 100,000



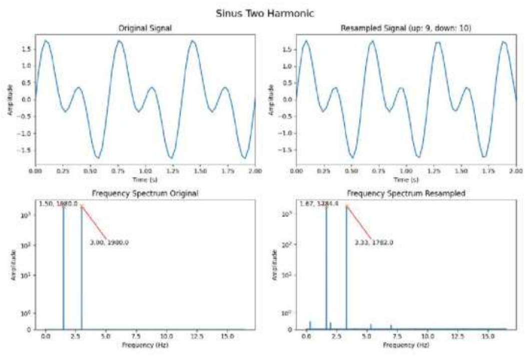
(d) Single sinus with upsample factor = 100,008 and downsample factor = 100,000

Figure 4.10: Time domain and frequency domain visualization of clock drift using rational resampling on a single sinus with different upsample and downsample factors. In addition to the dominant frequency of the signal, the top two figures also show additional small frequency components. These may be caused by slight noise during the resampling process. The successful resampling and, thus, change in the frequency of the peaks is visible. The lower two figures show a wider main lobe caused by the use of the Kaiser window. But no additional frequency components appear.

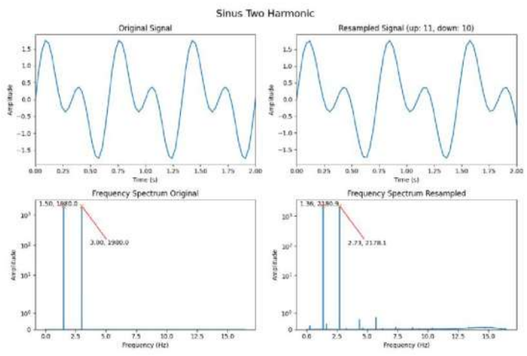
4.2.3 Functionality Verification of the Rational Resampling Function

To check the functionality of the rational resampling function and to ensure that it can be used to simulate a clock drift even with complex signals, different sine wave curves are generated and resampled before the function is used with the dataset itself.

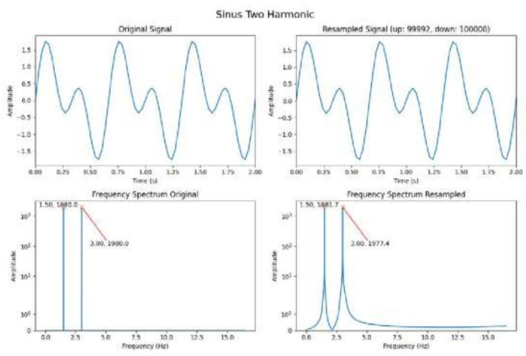
Figure 4.10, Figure 4.11, and Figure A.1 each show one or more sine waves with four different upsample and downsample factors. In all plots, the sampling frequency is 33 Hz, and the length of the signal is 120 s, of which only a short section is shown in the time domain to better visualize the sine wave and the change of it. The fundamental frequency



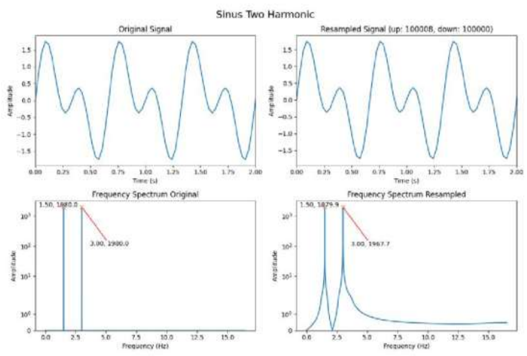
(a) Two harmonic sinuses with upsample factor = 9 and downsample factor = 10



(b) Two harmonic sinuses with upsample factor = 11 and downsample factor = 10



(c) Two harmonic sinuses with upsample factor = 99,992 and downsample factor = 100,000



(d) Two harmonic sinuses with upsample factor = 100,008 and downsample factor = 100,000

Figure 4.11: Time domain and frequency domain visualization of clock drift using rational resampling on two harmonic sinuses with different upsample and downsample factors. In addition to the two dominant frequencies of the actual signal, the top two figures also show additional small frequency components. These may be caused by slight noise during the resampling process. The successful resampling and, thus, change in the frequency of the peaks is visible. The lower two figures show wider main lobes caused by the use of the Kaiser window. But no additional frequency components appear.

of the signal is 1.5 Hz. In the subfigures with the factors 9/10 (a) and 11/10 (b), each upsample/downsample, the clock drift can be recognized very well in the time domain and in the frequency domain. A clock drift of $\pm 10\%$ should not occur in practice with any quartz, but these visualizations serve as a good visual aid. The other two subfigures with the factors 99.992/100.000 (c) and 100.008/100.000 (d) each have a clock drift of $\pm 8\,ppm$. Such a clock deviation is quite typical and thus realistic. However, since the deviation is so small, no difference can be seen in the time domain. The peaks in the frequency spectrum are only labeled to an accuracy of 10 mHz, but the deviation of $\pm 8\,ppm$ is in the 100 nHz range. This cannot be read from the labeling but the change is still there.

At the factor 11/10, peaks of other frequencies also occur several times. This is due to the fact that the change in sampling frequency caused by rational resampling is very large. However, this does not occur in the relevant ppm range.

4.3 Impact of the Single Sensor Channels

Before the various time discrepancies are applied to the different sensors, a verification is performed to ensure that all sensors contained in the dataset also have a general influence, effect, and impact on the result of the machine learning models. Applying different time discrepancies to sensor signals and determining their general influence on machine learning models is only possible if the sensors are also relevant. The knowledge gained from these tests is therefore an important part.
As the sensors are distributed over different parts of the body, it may be that one or more sensors are not moved much, as these parts of the body are not required for the ADL being performed. For example, if several sensors evenly distributed on a person's body are used to detect whether a person is typing on a keyboard or writing a letter, sensors on the feet will have only a negligible effect on the ability to distinguish between these two activities. This is because the person will most likely be seated during both of these activities and therefore the lower body will not or hardly move. However, sensors attached to the arms and hands will play a particularly important role in ensuring successful recognition. A failure of one of the arm sensors should therefore have a noticeable negative impact on the results of the model, while a failure of one of the leg sensors may not be noticeable at all.

In order to verify the influence of the individual signals used, a separate run of the testbench was created. In several runs, the samples of each individual sensor is set to zero once. This means that the sensor no longer has any effect on the training and testing of the model and therefore, no longer contributes to the results of the model since the samples of the sensor are always the same for all activities. To rule out random results, this is repeated with different temporal lengths and model complexities.

Three different temporal lengths were tested with two of the three model complexities, see Table 4.2. The "Sensor" column indicates which sensor signals were set to zero in the respective run, i.e. in this run this sensor did not influence the model and therefore had

Complexity	Length	Omitted Sensor	Accuracy	Edit	F1@0.1	F1@0.25	F1@0.5
medium	200	0	51.93	43.35	51.21	47.74	34.26
medium	200	1	31.33	33.1	17.56	13.26	8.02
medium	200	2	33.26	28.45	19.1	13.86	6.5
medium	200	3	57.66	26.02	28.24	23.27	11.52
medium	200	4	58.42	22.68	15.66	11.29	6.79
medium	200	-	**69.74**	52.1	53.53	49.62	39.4
medium	600	0	48.96	47.49	49.31	43.54	30.57
medium	600	1	28.62	35.79	16.99	13.71	9.39
medium	600	2	38.94	16.7	14.29	10.7	5.54
medium	600	3	43.25	45.84	23.29	17.96	10.48
medium	600	4	52.71	41.88	15.46	12.59	9.96
medium	600	-	**70.9**	52.31	52.42	49.4	40.62
medium	2000	0	49.48	56.99	52.62	47.93	32.97
medium	2000	1	27.76	33.83	19.01	15.02	11.18
medium	2000	2	28.71	29.77	17.65	12.94	6.71
medium	2000	3	52.23	36.61	30.28	24.31	11.47
medium	2000	4	55.64	43.86	19.11	15.26	10.63
medium	2000	-	**71.77**	55.72	59.82	56.91	46.37
shallow	200	0	50.32	28.16	36.84	33.67	23.24
shallow	200	1	29.26	18.71	12.93	9.31	5.39
shallow	200	2	32.48	13.0	9.68	7.2	3.58
shallow	200	3	51.14	11.37	13.59	10.01	4.61
shallow	200	4	52.72	12.31	7.27	4.59	2.38
shallow	200	-	**63.38**	43.38	48.88	45.65	34.62
shallow	600	0	46.02	29.72	36.16	32.02	20.15
shallow	600	1	29.33	30.91	16.09	12.48	7.99
shallow	600	2	29.4	8.89	6.71	4.45	2.0
shallow	600	3	48.41	17.31	17.76	14.36	5.66
shallow	600	4	54.01	13.18	8.02	5.91	3.0
shallow	600	-	**78.18**	39.66	43.0	40.71	32.25
shallow	2000	0	47.8	37.3	41.89	36.68	25.7
shallow	2000	1	26.81	32.27	13.58	11.03	6.96
shallow	2000	2	27.36	17.53	12.06	8.6	4.39
shallow	2000	3	48.71	18.88	18.71	14.4	5.77
shallow	2000	4	50.6	20.04	9.67	7.06	4.2
shallow	2000	-	**72.35**	38.09	45.39	42.51	33.96

Table 4.2: Overview of the accuracy, the edit distance, and the various F1-scores if one sensor is omitted in each case. In the rows in which the omitted sensor is “-”, all sensors are used normally. These rows serve as a reference value.

no influence on the results. The rows in which the sensor has the value “-” are reference values for the model in which all sensors were used normally without any sensor being influenced.

The reference model, in which all sensors were used, delivered the best results in all variants performed. The accuracy of the reference model is always over 10% better than the best model where one sensor is omitted. Although the accuracy of the model decreases to varying degrees when individual sensors are omitted, it decreases clearly for all sensors. This means that all sensors contribute to a proper detection of the ADL and are therefore relevant for a result with high accuracy.

For all models used, the accuracy of the model is lowest when sensor 1 is not used and almost as bad when sensor 2 is not used. Interestingly, these are the two sensors on the right arm, one on the right upper arm and one on the right forearm. The two sensors on the left arm, sensor 3 and 4, always, except once, have the least effect on reducing the accuracy of the network. This means that these two have the least influence on the decision as to which activity is currently being performed. However, the reduction in accuracy is also clearly present with approximately 11% to 30%.
Whether it can be deduced that the subjects studied were all right-handed and, therefore, the sensors on the right half of the body have a more important influence can only be hypothesized, as there is no information provided from the dataset. However, since the activities studied are all locomotion activities such as sitting, walking, and lying down, the influence of whether a person is right- or left-handed is assumed not to be important. The sensor attached to the back of the subject, sensor 0, also has a noticeable influence on the result. If it is not present, the accuracy is also considerably worse.

4.3.1 Application of the Discrepancies to the Sensor Signals

So far, it has been explained how the various time discrepancies can be applied to a single time signal that is independent of all others. Either a change in the time domain can be created, a change in the frequency spectrum, or both discrepancies can be generated simultaneously in order to create any desired changes.

During a run, a time discrepancy is currently applied to the time signals of one or more sensors. This time discrepancy can either be a shift in the time domain, a shift in the frequency domain, or both at the same time. Just like the time discrepancy, the affected sensors are predefined for a run. Any number of the five sensors can be selected and the discrepancy is applied individually to all six individual signals of the sensor. However, the discrepancy is a constant value and it is the same for all selected sensors. Whether and how useful this type of implementation is, will be clarified below.

In order to create a realistic environment, it does not seem reasonable to apply the same time discrepancy to several sensors. Instead, a shift of several sensor channels should be tested in any case. As it is unlikely that at most one sensor will be affected by one type of time discrepancy. In a non-synchronized sensor network in which the sensors are connected wirelessly, several time discrepancies of different sizes are quite possible.

Shifting several sensors at the same time can also produce a poorer machine learning model result for the same shift. However, if all time signals of the sensors are shifted by the same value and the same shift is applied during training and testing, the model is trained and tested with altered data, which may give worse results for other data, but for the training data and test data the result will be similar to no shift.

Therefore, instead of applying a single time discrepancy to the signals of one to five sensors, a probability distribution curve for the time discrepancy is used, with which different sized time discrepancies are applied to the signals of the five different sensors. One sensor should always be unaffected by any time discrepancy. As there are only five sensors, a probability distribution using a normal distribution does not seem ideal. Instead, a uniform distribution with a defined interval is used. Only 5 different time discrepancy values are used, which are distributed among the 5 different sensors. With such a small number, the use of a uniform distribution appears useful.

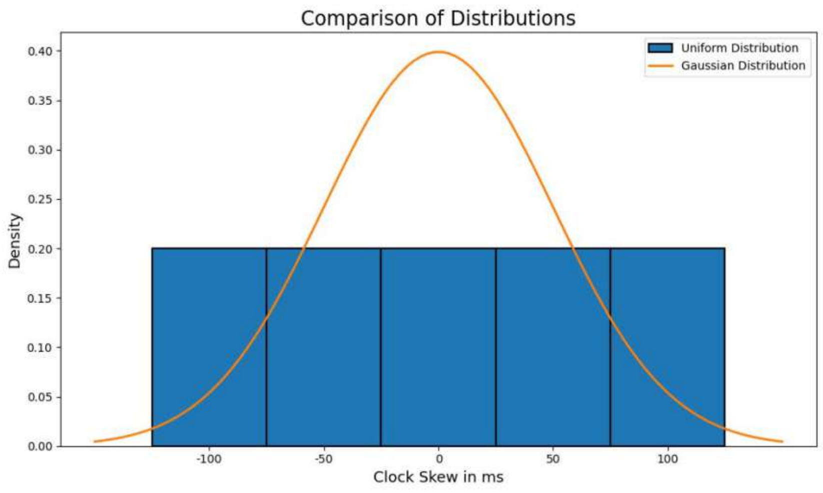


Figure 4.12: Comparison of Gaussian distribution and uniform distribution for a clock skew with a closed interval $\pm 100\,ms$

The uniform distribution used is explained using the example of a clock skew of 100 ms, see Figure 4.12. This also applies to the same extent to a shift in frequency. With a clock skew of 100 ms, a closed interval of $\pm 100\,ms$ is created, i.e. [-100,100]. The signals of two sensors are shifted by the two extreme values of +100 ms and -100 ms. Based on the uniform distribution used, one sensor is not affected by the clock skew. It is shifted by 0 ms. The signals from the remaining two sensors are distributed between the remaining two bins of the uniform distribution. This corresponds to exactly half of the two extreme values. As a shift is only possible by whole integer values, the middle values are rounded to the next integer value if necessary. This means that the following clock skews are used in this case: -100 ms, -50 ms, 0 ms, 50 ms, and 100 ms.

Five parameters for the testbench have been defined, which are varied to test a broad spectrum and finally make a broad statement. There are several different parameters for the machine learning model, which involve different models, see Section 4.5, and also different types and variants of the time discrepancy. If every possible variation of every time discrepancy was applied to all possible sensor distributions, it would increase the number of testbench runs. Since the five different time discrepancies are to be applied to all five different sensors, this is the stochastic problem of the number of possible events in a permutation without restrictions. This results in a number of possibilities of $n!$. Where n is the number of objects, with five different sensors, there are therefore 120 different ways of applying this to the various sensors for each individual time discrepancy that is to be examined.

Now, for example, 9 different machine learning models (m) are to be used, and the clock skews (cs) 0 ms, 100 ms, 200 ms, and 500 ms, as well as the clock drift (cd) 0 ppm, 10 ppm, 20 ppm, 50 ppm, and all 120 possible combinations of the time discrepancies (p). The total number of runs for the machine learning model, which must then be evaluated and analyzed, can be calculated as follows: $m \cdot cs \cdot cd \cdot p$. In this example that would be: $9 \cdot 4 \cdot 4 \cdot 120$. This results in a total of 17,280 runs. However, even more different models and even more different time discrepancies are to be tested, which would considerably increase the number of runs even more. It does not seem feasible to test the time discrepancies on all possible sensor combinations. Therefore, another experiment is being developed to determine whether a single, random sensor assignment is sufficient. If all 120 assignments are necessary or, if not, how many randomly selected sensor assignments are necessary to form a meaningful judgment. This is tested separately for both the clock drift and the clock skew. This means that a total of 240 machine learning models are trained and tested. The models for the clock drifts and clock skews are considered separately. The 120 possibilities are calculated in a random order and are then numbered consecutively.

The results are analyzed in a subsequent analysis. For this purpose, the runs are also randomly assigned again. The accuracies of the individual permutations are taken into account for deciding which number of sensor variations is sufficient to be able to make a meaningful statement.
For permutation 1, only a single, random sensor assignment is used. In permutation 20, for example, twenty random sensor assignments are considered and their accuracies are averaged. With permutation 120, all possible sensor assignments are taken into account and the average of all the accuracies is calculated. This means that pairs of different sizes are always formed for each permutation. For each possible permutation, as many of these pairs as possible are formed and their accuracies are averages, in order to subsequently calculate the mean and the standard deviation of the averaged accuracy pairs. Since not all permutation possibilities are always completely divisible by 120, not all, but always as many variants as possible are used, e.g. for permutation 17, 7 groups are formed, each with a size of 17. Within each of these 7 groups, the average of the accuracies is calculated. The mean and the standard deviations are then determined for these 7 groups. In permutation 17, one random combination is not taken into account, as only 119 combinations are required to form the groups.

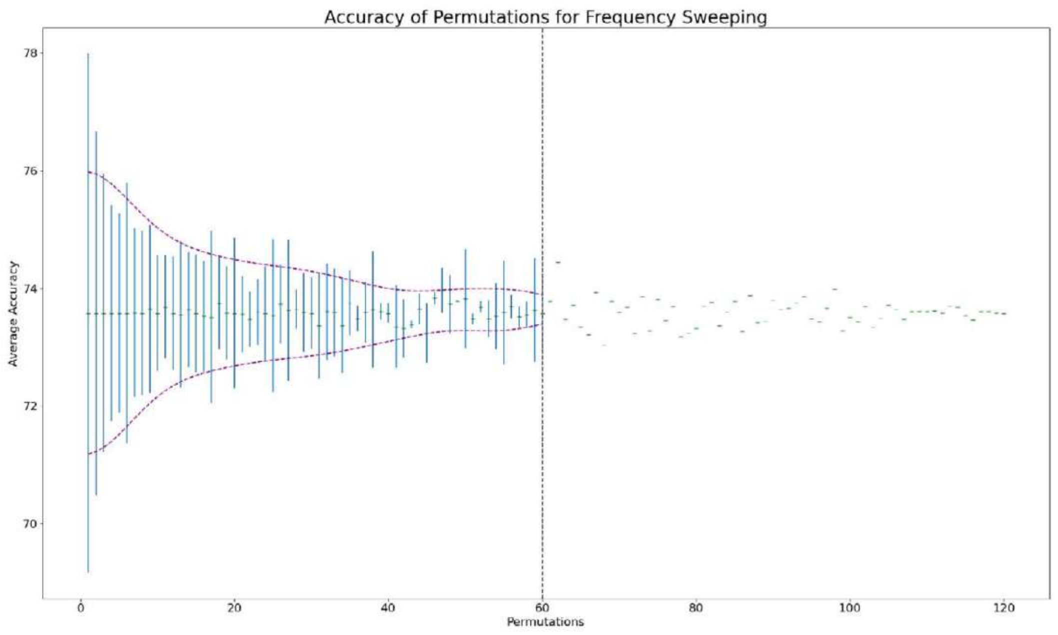


Figure 4.13: Mean and standard deviation of the averaged accuracy of all possible permutations for the sensors affected by a clock drift, with model complexity medium and a temporal length of 2,000 samples.

The mean and the standard deviation of the averaged accuracy for the respective number of permutations are shown in Figure 4.13, where the sensor signals are affected by a clock drift. The same is shown for a clock skew in Figure 4.14. The model complexity medium and a temporal length of 2,000 samples are used for this test run. In order to exclude random results, the same experiment is repeated again with a different temporal length of 500 samples. This produced similar results as the initially used temporal length. A comparable trend curve is visible.

Similar conclusions can be drawn for both the clock drift and the clock skew. The mean of the accuracies changes only slightly for all permutation variants. The greater the number of combinations over which the accuracy is first averaged, the more the mean fluctuates. From permutation 60, there is only one averaged accuracy value, so the mean of this is the same value as the averaged accuracy itself and there is no standard deviation. Fluctuations in the mean are more noticeable here, as there is no longer any proper averaging.

In order to identify a trend in the data, a Gaussian smoothing filter with a window size of 5 is applied to the data. In general, it is not necessary to calculate all possible permutations, as there are no unexpected results, such as a standard deviation that appears unexpectedly sized. The more permutations are considered, the smaller the standard deviation becomes. It can, therefore, be derived that a higher number of permutations also allows a more accurate conclusion to be drawn about the effects of the

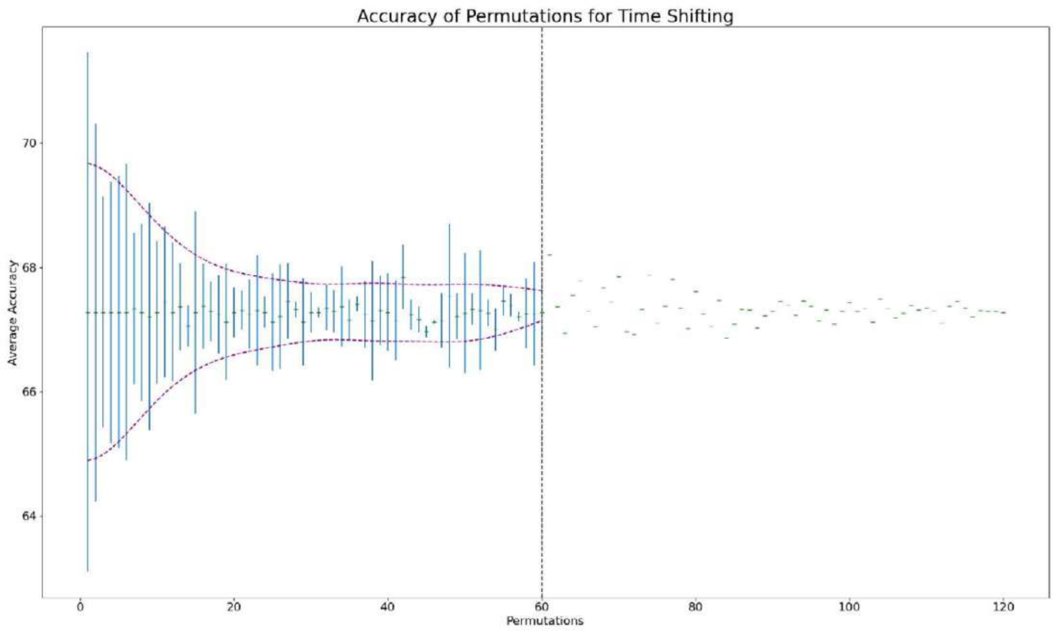


Figure 4.14: Mean and standard deviation of the averaged accuracy of all possible permutations for the sensors affected by a clock skew, with model complexity medium and a temporal length of 2,000 samples.

applied time discrepancy, which is not based on random values.
For the first values, the standard deviation declines very fast from over ±5% for the clock drift and from over ±3% for the clock skew. Therefore, if only a random distribution of the time discrepancy across the sensors were used, the significance would still be low. The trend line decreases very quickly for the first few values. In the case of clock drift, the second derivative has a zero crossing at the 6th permutation. For the clock skew, it is at the 8th permutation. Both second derivatives are thereafter greater than zero, which means that after the inflection point, they are left-curved and flatten out. After this, there are no further relevant inflection points for either function. A few more permutations are chosen to flatten the curve even more. The same number of required variants are selected for both time discrepancies for universality. The saddle point is reached at around 10 permutations for both clock drift and clock skew. From this point, the curve no longer continues to fall but begins to flatten out, and the changes in the standard deviations become smaller and smaller.
As a result, 10 permutations are used for the further testbench runs in order to be able to make a meaningful statement about the effects of the applied time discrepancies. The trend line continues to fall with a higher number of permutations, but only in the very low one percent range. The use of more permutations would considerably increase the number of variants and networks that need to be calculated, but would not contribute to a more precise statement.

4.4 Machine Learning Model

The MS-TCN [FG19] will be used as the machine learning algorithm and model for the temporal action segmentation of ADL. By using a temporal convolutional network, the features do not have to be extracted by hand, which would require some knowledge about the individual sensors and activities. Instead, the data is classified directly by the deep learning model. Although this network was initially developed for video input, it can also be used for data over time from IMUs.

4.4.1 Single-Stage Temporal Convolution Network

The single-stage TCN (SS-TCN) consists of a stack of several residual dilated convolution layers. The model can learn from larger time intervals without increasing the computational cost by using dilated layers. Each dilated convolutional layer is followed by a rectified linear unit (ReLU) operation. A residual connection is applied at the output to avoid the problem of vanishing gradient. The vanishing gradient problem occurs when training neural networks with a backpropagation [BJZP20]. As the network depth increases, the number of multiplications with which the gradients of previous weights are calculated increases. As a result, the gradient magnitude of previous layers shrinks exponentially and so does the influence of previous layers. [BJZP20] The first layer is a convolution layer with a 1x1 convolution. It is used to adjust the dimensions of the features from the input to fit the feature maps inside the TCN. The first layer is then followed by multiple layers with a dilated 1D convolution. [FG19]

Each single convolution layer is stacked to expand the receptive field. At the first layer, the dilation factor is 1. In each additional layer, it is doubled: 2, 4, 8, ... The receptive field for the single stage can be calculated as follows:

$$receptivefield = 2^{(L+1)} - 1 \tag{4.1}$$

Figure 4.15 shows an SS-TCN on the right using dilated convolutional layers. The total number of layers varies in the final testbench implementation as several models with different complexities to achieve a better generalization about the problem are used. The first layer is a 1x1 convolutional layer and the last layer of each single stage uses a *softmax* function.

4.4.2 Multi-Stage Temporal Convolution Network

The MS-TCN uses several stages in a row. Each stage, or more precisely each individual model, uses the output generated by the previous stage as input. This allows an improvement in the prediction per stage to be achieved. The first stage uses the features of the IMU as input, i.e. acceleration and gyroscope data. The next stage then receives the prediction of the first stage and then refines it. All further stages then do the same. The network can recognize dependencies between feature classes and can thus reduce the over-segmentation error. This happens because each stage outputs its initial prediction.

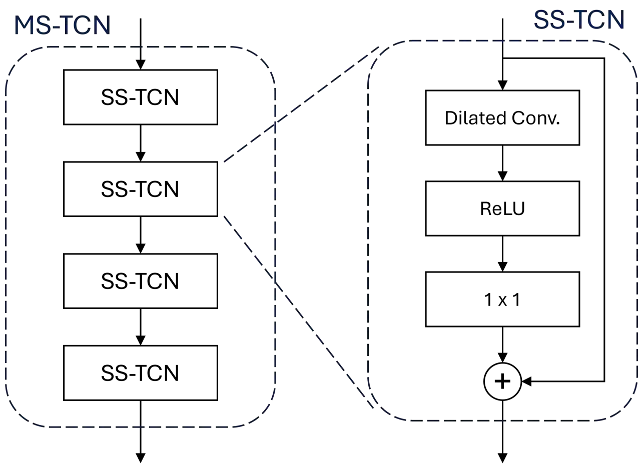


Figure 4.15: Overview of the MS-TCN and SS-TCN architecture. The MS-TCN consists of four stacked stages of single SS-TCN on the left. A dilated residual layer within the SS-TCN is shown in the right part.

In Figure 4.15, an illustration of the implementation of MS-TCN and SS-TCN is shown. The first stage has the only external input: the frame-wise features of the IMUs. After that, each stage uses the probabilities of the previous stage. Only the last stage provides an output to the outside, which is then used to classify the features.

4.4.3 Loss Function

The loss function quantifies the difference between the values predicted by the model and the actual values, the ground truth. The MS-TCN uses a combination consisting of a smoothing loss for refinement and the classification loss as the main part. The cross entropy loss measures the difference between the values predicted by the probability distribution of the model and the actual class labels. This is used here for the classification loss. In order to reduce over-segmentation errors, smoothing loss is also used to reduce precisely these errors.

4.5 Model Selection

In order to achieve a broader generalization, not only one model will be selected, but three base models with different complexities will be defined. The models are further categorized into the groups shallow, medium, and deep and are also referred to as such. The shallow models are the models with the lowest complexity, which primarily means the fewest layers and stages, and therefore also the least number of trainable parameters of the network.

The parameters for the models were selected by analyzing relevant papers and scientific work that also uses the MS-TCN model. Not only the parameters used, but also the

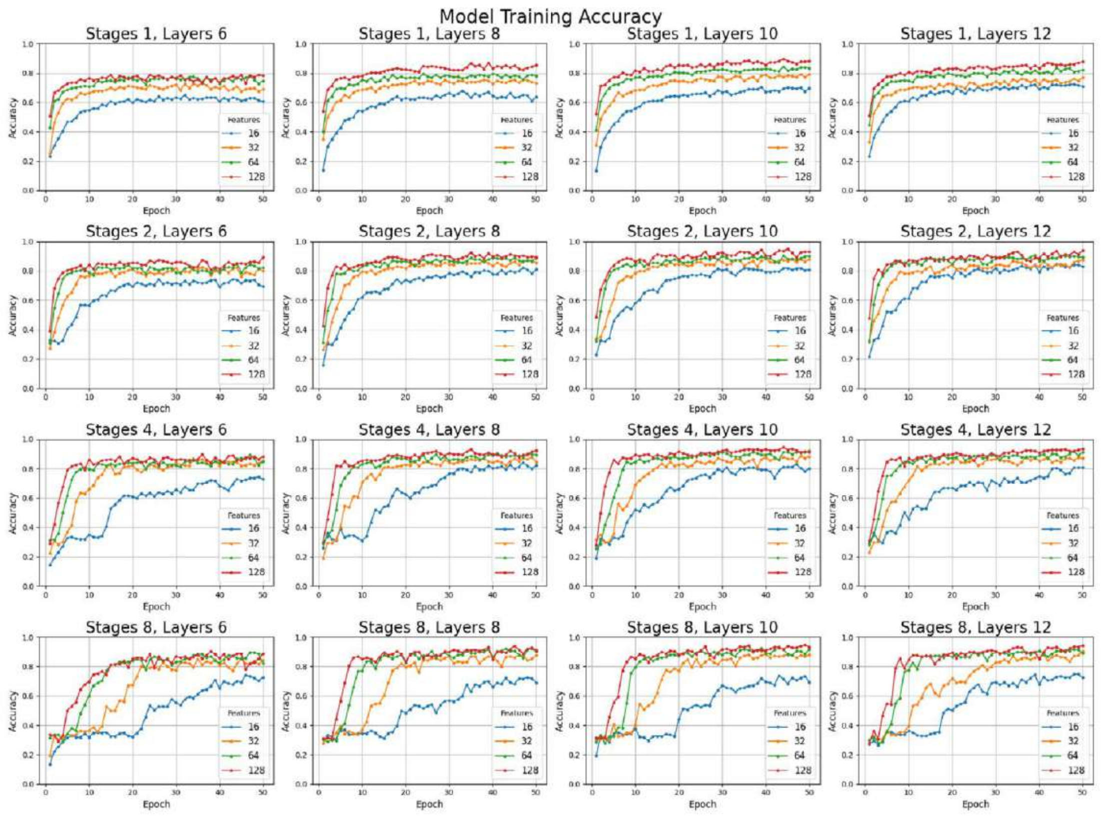


Figure 4.16: Training accuracy across training epochs for different variations of levels, layers, and features.

use case, the type of input, the datasets used, and finally the accuracy and F1-score achieved with the network were taken into account. The type of input refers to whether it is sensor data from IMUs or videos. The number of input features is also relevant, as many features require more complex models to achieve adequate results. The papers were selected to deal with one type of activity segmentation, ideally with ADL. A number of papers unfortunately do not specify all the properties of the networks used, or sometimes none at all.

To further support the decision for different model complexities, an overview of the model training accuracies is also created, see Figure 4.16. The training accuracy is a measure of how well a model has learned the patterns in the provided training data [SW11]. The accuracies during the training of 50 epochs are considered for different model settings. Here, it can be seen that 16 feature maps are probably too few to model the complexity of the inputs, as the accuracy only increases very slowly over more epochs. Combinations containing 10 or even 12 layers are probably too complex for the required application since only time signals are used as input and no video signals. As the complexity is increased, the time required for training and testing the model also becomes longer,

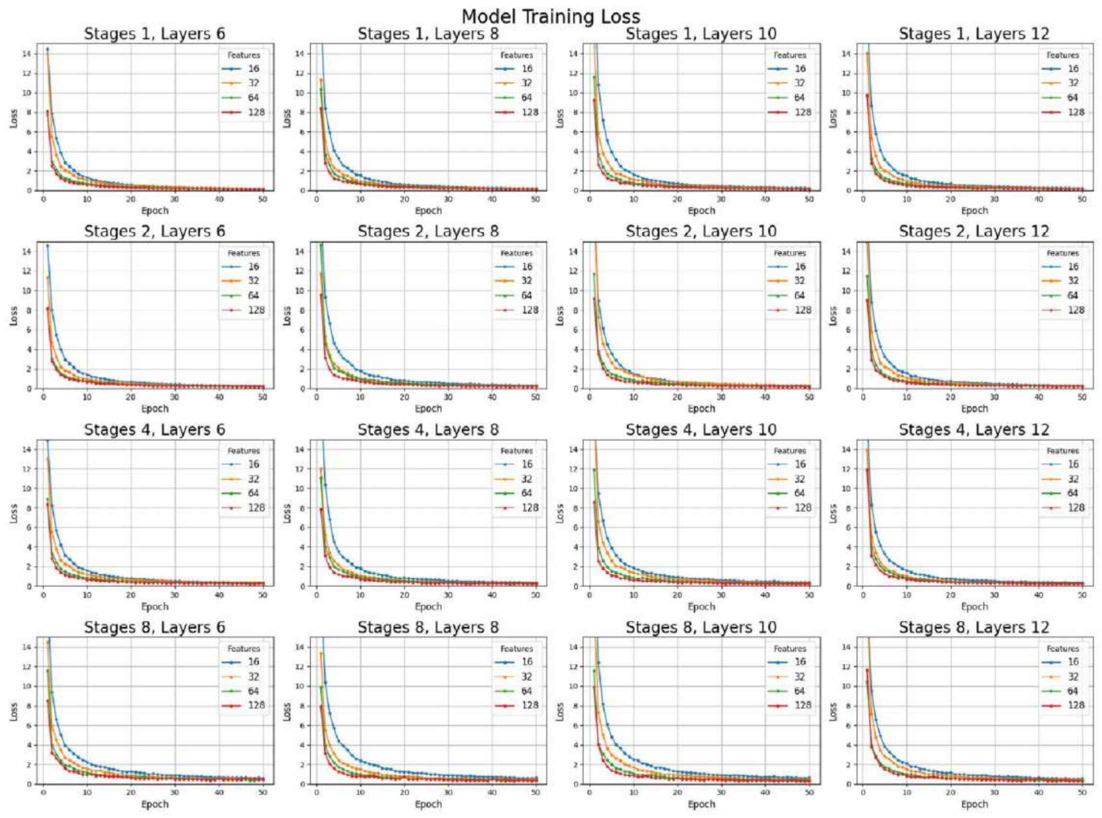


Figure 4.17: Training loss across training epochs for different variations of levels, layers, and features.

as there are far more trainable parameters. Such large layers also mean a very large receptive field, which also considerably increases the minimum length for the temporal length. It can also be seen that no more than 50 epochs are required to train the model. The accuracy peaks very quickly after just a few epochs.

In addition, an overview of the various losses in model training is also created over the training epochs, see Figure 4.17. The model loss shows how good a model is at predicting results and indicates the difference between the predicted and actual values [SW11]. In the first 10 epochs, the loss decreases for all models, and from epoch 10 to 20 the loss still reduces considerably. The models with fewer features perform worse than those with more features. The loss decreases visibly slower with a lower number of features.

Based on the created Table 3.2, a selection of parameters for several models was made. The models have the following parameters. Shallow models consist of 2 stages and 6 layers. Medium models have 3 stages, with the same number of layers. Deep models are equipped with 4 stages and 6 layers. All models have a learning rate of 0.005 and 128 feature maps.

But not only three different models are used. Rather, these three models are the overall category. The model groups are further divided using different temporal lengths. The temporal length is another important factor that defines a TCN model. It specifies how many individual, consecutive data points the model takes into account to predict and classify the features. The temporal length is particularly important with regard to recognizing temporal dependencies. It is also directly related to the receptive field. The receptive field of a neuron indicates the time range over which the neuron collects information or which input data influences a neuron. The time range here refers to how many samples are taken into account around the current sample. In the first layer of a convolution network, each individual neuron has only a small local receptive field. The receptive field of the individual neurons grows per layer. This means that a neuron in a later layer is influenced by more input data than a neuron in one of the first layers. If the temporal length is too short, the model may not recognize longer dependencies and the accuracy of the prediction may suffer as a result. A temporal length that is too long can lead to overfitting, which means that the model is then too closely adapted to the training data. The receptive field for the MS-TCN is calculated as described in Equation 4.1. The kernel size is 3 and the stride is 1 for all layers in all used networks. Only with these parameter settings, the given formula is valid. The temporal length must always be greater than the receptive field. Based on Equation 4.1, the models have the following receptive field r: shallow models: r = 127, medium models: r = 127, deep models: r = 511. This means that temporal lengths of at least 128 samples can be used for shallow and medium models. At least 512 samples are required for deep models. The Opportunity dataset has a sampling frequency of 33 Hz. Therefore, the smallest input length in seconds is 3.88 s and 15.52 s long.

The different temporal lengths in particular are intended to ensure that neither overfitting occurs nor that some activities are not recognized because the time interval is too short to be able to recognize them effectively. For the temporal lengths, the existing implementations in the related work were included in the selection again. In addition, the temporal lengths used should recognize ADL, as their duration is typically within this time span. A total of ten temporal lengths were then selected, with the following sample sizes: 198, 330, 528, 660, 825, 990, 1320, 1980, and 3300. As the receptive field is larger for the deep models, this model only has a total of seven temporal lengths. The various chosen combinations are shown in Figure 4.18.

4.6 Cross-Validation

Cross-validation is used to ensure the generalizability of the machine learning algorithm. Instead of splitting all data of the Opportunity dataset only once and randomly into training and test data, the data is split several times into different variants of training and test data. This should ensure an even more robust system. First, the various data in the dataset must be divided into training data and test data. A classic split ratio of 80/20 (training/test) [Jos22] seems reasonable at first. Since there are a total of 20 runs in the Opportunity dataset, this division is possible. However, the dataset has four subjects,

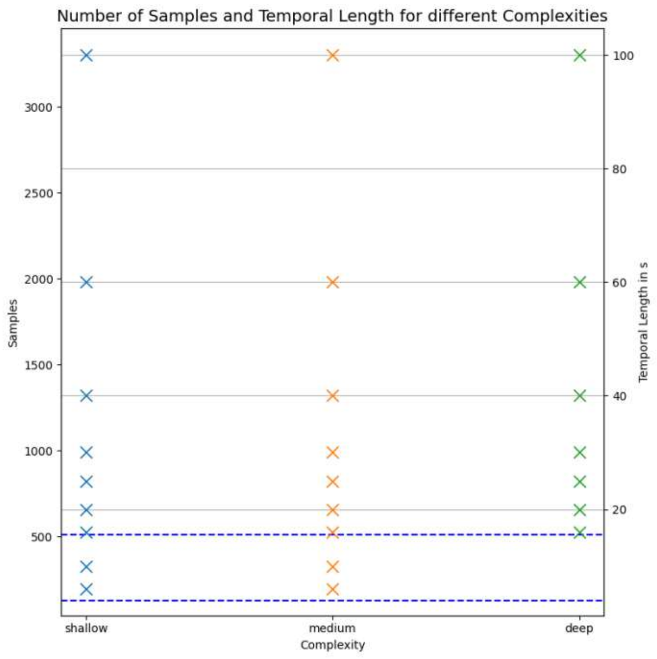


Figure 4.18: Number of samples used, as well as the temporal length in seconds, of the three different models.

each with 5 runs of ADL. Here, a division of 4 runs per subject for training data and the remaining run per subject for test data could be performed. This would basically be a leave-one-subject-out method [SW11]. However, such a division can also introduce a bias into the trained model, as every subject used in the test dataset is also present in the training dataset. A subject that is used for training should no longer be used for testing, as the testing of the model is then no longer independent. In this case, the machine learning model is trained precisely on specific users, which is equivalent to overfitting. This means that precisely these users can be recognized well. However, the model then has a drop in accuracy for other users that are completely unknown to the model.

For the reasons just mentioned, a distribution ratio of 75/25 is chosen instead. The data is split so that three subjects, with a total of 15 runs, are provided for the training data, and the remaining single subject, with 5 runs, is left for testing. The dataset is therefore split using 4-fold cross-validation. This means that it is divided into four equal parts, also known as folds. In each part, one fold, in this case, the 5 runs of one subject, is selected for testing. This process is repeated four times so that each subject is used once for testing. Such a part is referred to below as a split. There are splits one to four. The number of the respective split means the number of the subject that is used for testing, i.e. split one uses subject two, three, and four to train the model and subject one to afterward test the model.

In a further preliminary run of the testbench, the four different split variants are run through on several models with different temporal lengths. Two different temporal lengths are selected for each model complexity. The accuracy, the edit distance, and the different F1-score values are compared. Based on this data, see Table 4.3, split four is selected for the remaining parts of the work. This split performs best across all parameters used for evaluation. Split one delivers similar results, but in order to prevent the total number of models run at the end from growing exponentially, only one split is used further.

Complexity	Length	Split	Accuracy	Edit	F1@0.1	F1@0.25	F1@0.5
deep	2000	split1	76.82	56.23	55.47	53.78	45.32
deep	2000	split2	**80.77**	**61.4**	60.1	55.38	39.3
deep	2000	split3	73.68	60.2	60.4	56.27	44.93
deep	2000	split4	76.13	60.04	**61.32**	**58.79**	**47.91**
deep	600	split1	76.37	58.51	57.23	53.28	40.28
deep	600	split2	**81.18**	61.15	62.3	60.34	45.53
deep	600	split3	72.06	60.08	59.9	57.03	44.79
deep	600	split4	73.67	**68.25**	**65.47**	**61.87**	**50.84**
medium	2000	split1	**78.64**	46.22	47.77	45.78	39.19
medium	2000	split2	72.89	37.74	44.44	41.5	30.52
medium	2000	split3	69.24	52.21	52.31	49.15	36.98
medium	2000	split4	75.92	**57.94**	**55.39**	**51.82**	**39.35**
medium	600	split1	**78.49**	51.12	48.56	47.27	38.23
medium	600	split2	70.86	40.1	43.46	39.88	29.3
medium	600	split3	71.55	59.15	59.02	**56.47**	**45.76**
medium	600	split4	75.98	**56.55**	**59.1**	55.94	44.54
shallow	2000	split1	**77.09**	35.71	36.98	35.47	29.67
shallow	2000	split2	72.87	28.55	33.89	31.55	22.13
shallow	2000	split3	71.52	36.96	40.18	37.4	28.99
shallow	2000	split4	74.8	**42.09**	**46.95**	**44.59**	**35.62**
shallow	600	split1	76.27	**40.38**	40.09	38.13	33.74
shallow	600	split2	76.6	29.5	36.45	33.9	25.47
shallow	600	split3	72.05	34.65	**41.77**	**39.84**	29.4
shallow	600	split4	**78.91**	36.98	41.65	39.32	**30.6**

Table 4.3: Accuracy and F1-score overview for the four different splits with different machine learning model settings.

4.7 Model Training

The MS-TCN originally trains on video input data. However, time signals from various sensors are used in this work. This reduces the number of features needed to map the input correctly. The data used in the Opportunity dataset always consists of six individual time series that belong together, as these are the 3-axes of an accelerometer and 3-axes of a gyroscope, which are located within an IMU. This means that they are

considered as a single unit with identical time base. Five different sensors are used. This leads to a total number of 30 feature dimensions. The original implementation uses input data from the GTEA dataset, Breakfast dataset, and 50Salads dataset. These are all video datasets and therefore have far more features. The features here are also not much longer than about 1,000 samples. However, they contain more individual runs and more different subjects. This leads to more possible runs of the model to train it and also to test it. The individual runs are always used as a whole, i.e. in the order of magnitude of 1,000 samples as input for the model.
With a training/test ratio of 75/25, the Opportunity dataset only offers 15 possible runs for training. This is relatively few to be able to train a model well. In addition, the individual runs have up to 50,702 individual samples. At a sampling frequency of 33 Hz, this means a run length of more than 27 minutes. Using such a long signal as input for a model is not very effective. This can increase the over-segmentation of the model, since the model is also trained on exactly the sequence occurring in the training set and not on the pure detection of the activities. In addition, the lengths of the various runs in the Opportunity dataset vary considerably. The shortest run has a length of only 21,216 samples. The mean value of the lengths of the dataset is 31,554 samples with a standard deviation of 6,243. For the reasons just mentioned, the individual runs of the dataset are not used as input as a whole but are divided into smaller parts. Different input lengths, also called temporal lengths, are used, see also Section 4.5. Taking only one single section of each run would be too little data to be able to train the model well. This single section with a length of 132 to 3,300 samples would also be unable to cover all different kinds of the performed ADL. Therefore, several periods are selected from each run in order to be able to train the model with more data. A total of 50 randomly selected sections from each run are chosen to train the model.

The training itself is performed over 50 epochs. An epoch of a model is defined as a complete iteration of the entire training dataset. Every 10th epoch and the best epoch are saved. The best epoch of training is defined by the lowest epoch validation loss. If there is no further improvement in the validation loss over three consecutive epochs, the training is stopped early to avoid overfitting. A schematic flowchart of the entire workbench, including training, is shown in Figure 4.19.

4.8 Model Testing

The entire run is not used as input for the model at once, but the same temporal length from the training is also used for the input here. However, the test files are not divided up completely randomly and tested with these parts, but are taken piece by piece to facilitate subsequent validation. This simplifies the process of assigning the individual samples over time and later comparing the individual samples with the correct ground truth samples. The best training epoch of the model with the lowest epoch validation loss is used for testing. The predicted values for the test runs are saved.

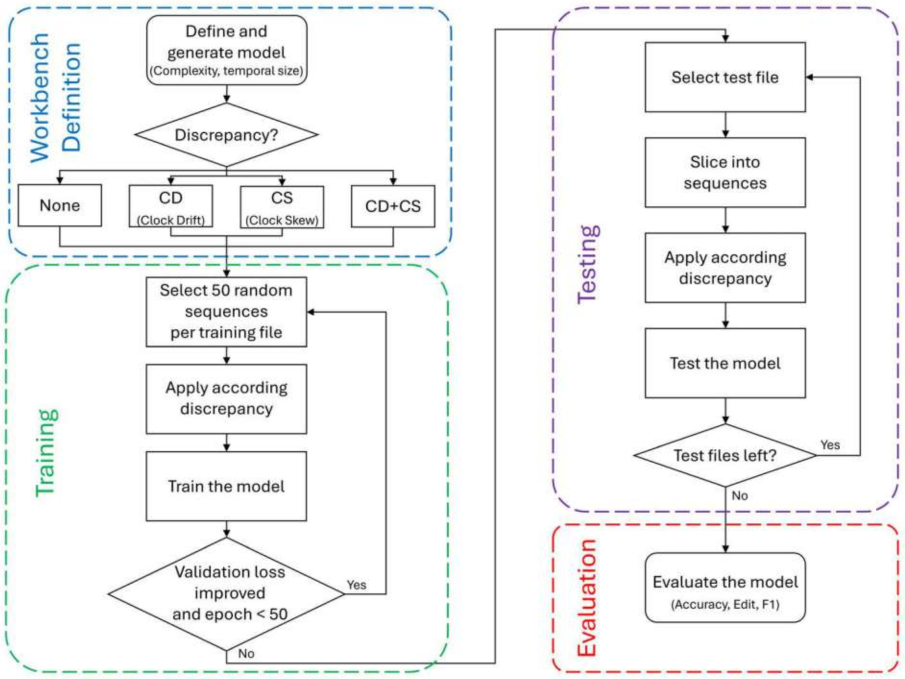

Workbench Definition
Define and generate model (Complexity, temporal size)
Discrepancy?
None
CD (Clock Drift)
CS (Clock Skew)
CD+CS
Training
Select 50 random sequences per training file
Apply according discrepancy
Train the model
Validation loss improved and epoch < 50
Yes
No
Testing
Select test file
Slice into sequences
Apply according discrepancy
Test the model
Test files left?
Yes
No
Evaluation
Evaluate the model (Accuracy, Edit, F1)

Figure 4.19: Flowchart of a single testbench run: First, the model and the time discrepancy are defined. Then, the model is trained and tested, and afterward evaluated.

4.9 Model Evaluation

To evaluate the results of the models, the predicted samples are compared with the ground truth samples. Based on this comparison, the accuracy, the edit distance, and the F1@0.1, F1@0.25, and F1@0.5-scores of the model are calculated. The calculated values are also saved again. The individual parameters and their relevance to the evaluation are defined in Subsection 2.3.2.

4.10 Implementation into a Testbench

So far, the individual components, such as the MS-TCN model used, the Opportunity dataset, and the applied time discrepancy method, have been explained separately. The following section explains how the entire testbench is structured in order to test various combinations of time discrepancies on the signals of different sensors and various complexities of the machine learning model and to analyze the results.

A rough subdivision can be made, as visualized in Figure 4.20, into the preparation for the input into the MS-TCN, i.e. the synchronized dataset, as well as the application of time discrepancies to it, the MS-TCN itself, and finally the evaluation of the predictions of the MS-TCN.

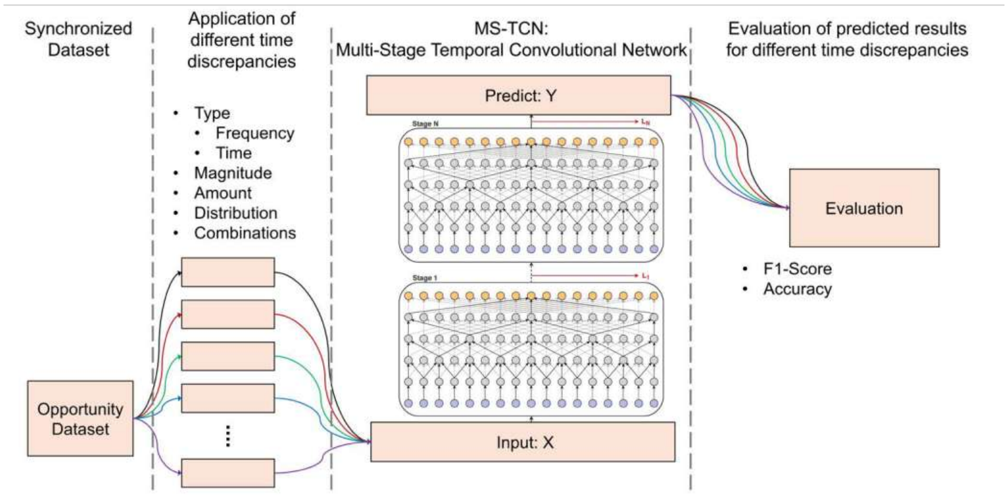


Figure 4.20: Abstract illustration of the testbench process.

4.10.1 Parameter Definition and Overall Structure

For the first testbench runs to find the most suitable split, to determine the number of permutations used, and also to find the parameters for the different complexities of the models, the various parameters required were defined in a separate file, which is loaded afterward and on the basis of which the runs are specified.

All folder structures are also pre-defined in the file. On the one hand, from the Opportunity dataset, so that the features, splits, ground truth files and the mapping can be read in correctly, and on the other hand, from the MS-TCN model, where the individual models and the results are stored. The folder and file structure are illustrated in Figure 4.21. The root folder contains the Python files required for the testbench as well as files for analyzing various data. The individual folders for the models, results, and pickle files are generated generically according to a specific structure. The folder name indicates which model complexity, which temporal length, and which split are used, as well as how large the shift is in frequency and time, and which sensor combinations are affected by time discrepancies. Additionally, there are parameters for the number of features and the sampling rate of the dataset.

Numerous parameters are also defined for the various MS-TCN models. These include the names of the models: shallow, medium, deep, and the temporal length used. However, in order to generate only meaningful models, the receptive fields of the three basic models are calculated based on the number of layers, and then only temporal lengths that are larger than the receptive field of the respective model are used. Each model is further defined by stages, layers, feature maps, batch size, learning rate, number of epochs, and how often random, individual parts are used per ADL run in each epoch for training. The various time discrepancies are also defined here.

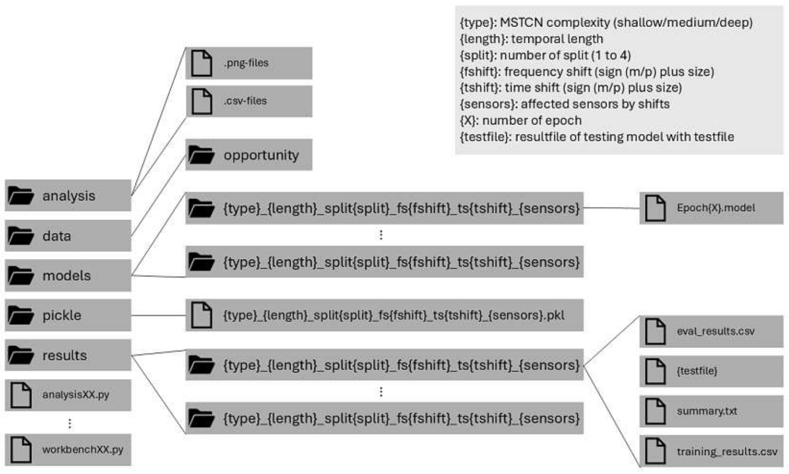

{type}: MSTCN complexity (shallow/medium/deep)
{length}: temporal length
{split}: number of split (1 to 4)
{fshift}: frequency shift (sign (m/p) plus size)
{tshift}: time shift (sign (m/p) plus size)
{sensors}: affected sensors by shifts
{X}: number of epoch
{testfile}: resultfile of testing model with testfile
.png-files
.csv-files
opportunity
analysis
data
models
pickle
results
analysisXX.py
⋮
workbenchXX.py
{type}_{length}_split{split}_fs{fshift}_ts{tshift}_{sensors}
⋮
{type}_{length}_split{split}_fs{fshift}_ts{tshift}_{sensors}
{type}_{length}_split{split}_fs{fshift}_ts{tshift}_{sensors}.pkl
{type}_{length}_split{split}_fs{fshift}_ts{tshift}_{sensors}
⋮
{type}_{length}_split{split}_fs{fshift}_ts{tshift}_{sensors}
Epoch{X}.model
eval_results.csv
{testfile}
summary.txt
training_results.csv

Figure 4.21: The file and folder structure of the testbench.

4.10.2 Execution of the Various Models

The previously defined parameters of the model, the dataset, and the time discrepancies are then read in and processed. One model after another is processed in a large number of loops. Where possible, PCs with a GPU that supports CUDA are used.

CUDA stands for Compute Unified Device Architecture. It is an API with which the computing power of GPUs can be used for general calculations and not just for displaying graphics. The advantages include the fact that GPUs are many times faster than CPUs when training neural networks [Sar24]. In addition, several GPUs can be used in parallel and the parallel calculations in a GPU can maximize the utilization of the hardware. [20225a]

Since the training and testing of the networks already take up the entire GPU, it is not effective to train several models simultaneously using multiprocessing. With Python multiprocessing, calculations can be performed with several CPU cores at the same time. Since the models are trained with CUDA, the GPU would make context switches between the processes, which would not improve parallelization. This would slow down the individual training of the network. Using CUDA Multi-Process Service (CUDA MPS), the GPU can also process several processes in parallel to a certain extent without performing context switching. However, when using CUDA in the normal way, in a single process, all parallel capacities of the GPU are already utilized, which is why both multiprocessing and MPS will not be used in the following. But for the sake of completeness, these have been listed. For the reasons just mentioned, the decision was made to train and test the individual models in series rather than in parallel.

First, the machine learning model is created. Here it is defined that torch should support CUDA and use it. Torch is an open-source deep learning framework. Then, the defined parameters for the model are set and the multi-stage model is created. The file paths are also defined here and all required folders are generated. A PyTorch summary is created and saved for the relevant model, in which the names of the layers, input and output shape, kernel shape, number of parameters, and number of operations are depicted. The training of the model is started, then. Following, the model is tested as well as evaluated, and finally, various variables are saved using Pickle before the next model is started with a different time discrepancy.

4.10.3 MS-TCN in the Testbench

The MS-TCN is created per iteration based on various predefined parameters. Among other things, the torch device is first set to CUDA so that the GPU can be utilized for the required computations. Then, the parameters that vary the complexity are taken from the parameter definition and are saved for the individual model. This also includes the name of the model, as well as the number of stages, layers, feature maps, batch size, learning rate, all required file paths, and other parameters. The actual multistage model is initialized with these settings, then. The first stage is created separately as a single stage model. All further stages are created as single stage models in a module list. The 1x1 input convolution, a module list of layers, and the output convolution are defined within the models. Once a model has been successfully prepared, a summary is created using torch.
The summary provides a simple overview of the model. The various stages and the layers which they contain are listed almost illustratively. These also contain the output shape of the respective layer or model, as well as their number of parameters. There is also the total number of parameters shown and how many parameters can be trained. An example of the summary is shown in Listing 4.1. Here, a model of complexity medium with a temporal size of 2,000 samples is used. The complexity medium means that the model is defined with 3 stages, 6 layers, and 128 feature maps.

Then, the MS-TCN is trained with the Opportunity dataset. Depending on the split selected, different subjects are used for training. Regardless of the split, 15 runs of ADL are always used for training and the remaining 5 runs of the fourth subject are used for testing. As the individual runs are all very long to be used as input for the MS-TCN as a whole, only a random part of the run is used to train the model using a previously defined temporal length. Randomly, 50 parts of the entire run are taken for training per epoch. In total, the model is trained for up to 50 epochs. However, the training is terminated early if there is no improvement in the validation loss over several epochs. This is to avoid overfitting the model to the training data. The model with the lowest training validation loss is saved separately, as it is then used for further testing and subsequent evaluation.

When testing the model, the same temporal length is used as during training. Once the model has been tested on the fourth subject, training, and testing are complete. When the model is then evaluated, the results are compared with the ground truth. This is based on the accuracy, the edit distance, and the three F1-scores which are calculated and saved.

Listing 4.1: Model Summary of a medium Model with temporal size of 2000

```
=====================================================================
Layer (type:depth-idx)                    Output Shape      Param #
=====================================================================
MultiStageModel                           [3, 10, 2000]     --
+ SingleStageModel: 1-1                   [10, 2000]        --
|    + Conv1d: 2-1                        [128, 2000]       3,968
|    + ModuleList: 2-2                    --                --
|    |    + DilatedResidualLayer: 3-1     [128, 2000]       65,792
|    |    + DilatedResidualLayer: 3-2     [128, 2000]       65,792
|    |    + DilatedResidualLayer: 3-3     [128, 2000]       65,792
|    |    + DilatedResidualLayer: 3-4     [128, 2000]       65,792
|    |    + DilatedResidualLayer: 3-5     [128, 2000]       65,792
|    |    + DilatedResidualLayer: 3-6     [128, 2000]       65,792
|    + Conv1d: 2-3                        [10, 2000]        1,290
+ ModuleList: 1-2                         --                --
|    + SingleStageModel: 2-4              [10, 2000]        --
|    |    + Conv1d: 3-7                   [128, 2000]       1,408
|    |    + ModuleList: 3-8               --                394,752
|    |    + Conv1d: 3-9                   [10, 2000]        1,290
|    + SingleStageModel: 2-5              [10, 2000]        --
|    |    + Conv1d: 3-10                  [128, 2000]       1,408
|    |    + ModuleList: 3-11              --                394,752
|    |    + Conv1d: 3-12                  [10, 2000]        1,290
=====================================================================
Total params: 1,194,910
Trainable params: 1,194,910
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 152.49
=====================================================================
Input size (MB): 0.24
Forward/backward pass size (MB): 80.35
Params size (MB): 4.78
Estimated Total Size (MB): 85.37
=====================================================================
```

4.10.4 Application of Time Discrepancies

The application of the various time discrepancies, also defined as parameters, to the individual signals only takes place within the MS-TCN. The problems and implementation are nevertheless explained in Section 4.2.

A clock skew manifests itself in a change in time, i.e., a shift of a signal by a constant temporal value, either forward in the case of a positive shift or backward in the case of a negative shift. The other applied time discrepancy is a change in frequency, which is usually due to clock inaccuracies. This means that the signal is shifted by a constant value in the frequency spectrum. This clock drift can also take two different types, either the signal is compressed or stretched by applying this form of time discrepancy. Both time discrepancies are seen in relation to an absolute, external reference system in which several sensors together form a sensor system consisting of numerous IMUs, which ideally should provide synchronized data. The time discrepancy is always applied to all six time signals of a single sensor together.

However, when applying the time discrepancies, this is not used on the entire time signal. Instead, it is only always applied to the time segment that is currently being used to train or test the model. For this reason, different datasets are not created with different types of discrepancies, and the original, synchronous dataset is not changed before the model training is started in order to subsequently use it for a time discrepancy. This is only done, once a random part of the entire signal has been selected to be used as input for the model.

This procedure is necessary because a change in frequency triggered by a clock inaccuracy, for example, indirectly results in a clock skew when using a time segment. See Subsection 2.1.2 for a more detailed explanation of the problem. If a signal $y_0(t) = s(f_0 \cdot t)$ with the actual frequency f_0 is stretched by a frequency Δf from time t=0, the signal $y_0(t)$ becomes a new signal $y_1(t) = s((f_0 + \Delta f) \cdot t) = x(f_0 \cdot t + \Delta f \cdot t)$. When a time segment is now taken from the signal y_0 from any point in time t_a and a time segment is also taken from the signal y_1 from the same point in time t_a, then the starting point of this time segment has a time offset by the frequency change Δf.

This means that if a signal has a frequency change from the beginning, every section that is used that does not start from time $t = 0$ but at some later point also has a time change at the same time. However, since clock skews and clock drifts are two different types of time discrepancies, and the effects of both are to be determined independently of each other, only the section itself is changed and not the entire signal. Nevertheless, combinations of the different time discrepancies are also used to depict the scenario just described.

4.10.5 Model Computations using High-Performance Computing - Vienna Scientific Cluster

In the previous subsections, the various parameters, their meaningfulness, and their different values were explained, and their importance was justified. The various parameters that vary in the different calculation runs are summarized below to provide an overview of the iterations.

Three different complexities of the machine learning model (shallow, medium, deep) are tested with nine different temporal lengths of the input signals (198, 330, 528, 660, 825, 990, 1320, 1980, 3300). The models with the complexity deep have a larger receptive field. Therefore, the two smallest temporal lengths (198 and 330) are not used. This results in a total of 25 different machine learning models. The time discrepancies are applied in four different areas (none, all, train, test). The following nine different clock skews in samples are utilized: 2, 5, 10, 20, 50, 100, 200, 500, 1000. The following ten different clock drifts in ppm are utilized: 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000. In addition, eight combinations of the two time discrepancies are also applied (clock drift | clock skew): (2 | 2), (10 | 10), (10 | 100), (100 | 10), (100 | 100), (500 | 500), (1000 | 1000), (2000 | 2000). This results in 27 different time discrepancies. These are each applied to ten random sensor variations. Altogether, 270 different time discrepancies are computed with different allocations to the sensors.

The temporal length for the input of the respective models corresponds to a sampling frequency of 33 Hz: 6 s, 10 s, 16 s, 20 s, 25 s, 30 s, 40 s, 60 s, and 100 s. The clock skew applied at the same sampling frequency corresponds to: 0.06 s, 0.15 s, 0.3 s, 0.61 s, 1.52 s, 3.03 s, 6.06 s, 15.15 s, and 30.30 s.

The different time discrepancies are then also applied in various ways. In an "all" run, the time discrepancy used is applied both when training the model and when testing the model. In the "test" run, the training is carried out with data that is not influenced by a time discrepancy, so the signals of these sensors are synchronous, but an artificial time discrepancy was added when training the model. In the "train" run, the time discrepancy used is present during training, but the sensors are synchronized during the testing of the model. In the "none" run, the sensors are synchronized during both training and testing of the model, i.e. no time discrepancy is applied. These runs also serve primarily as reference values for how accurately a model can be trained with synchronous data.

In the three variants in which time discrepancies are applied to the time signals of the different sensors, they are applied to all 270 possibilities. In the "none" runs, there is only one run, as no time discrepancies are applied here. This leads to a total of 811 different applications per model of time discrepancies and no time discrepancies.

This results in a total of 21,025 runs of the machine learning model. 25 of these models have no time discrepancy and serve as reference models. 21,000 runs are affected by various time discrepancies. In each case, 10 of these runs are affected by the same time discrepancies, but in different random sensor splits, of which the average is calculated in each case, which then leaves a total of 2,125 different time discrepancy variants. Each individual run takes between a few minutes to more than an hour to train, test, and evaluate the model, so it would require quite a long time to run more than 20,000 runs on just one computer.

These complex models, and particularly the large number of models to be calculated, would push a single computer beyond its limits. For this reason, high-performance computers are used for further calculations and various runs. Due to their parallel processing and much more powerful hardware than individual PCs, they are also able to calculate larger machine learning tasks in considerably shorter time.

The Vienna Scientific Cluster (VSC) [20225b] is one of the leading high-performance computing infrastructures in Austria and offers researchers powerful computing resources. The architecture of the VSC comprises a large number of nodes with powerful CPUs and GPUs. This environment simplifies the efficient execution and training of machine learning models. The workload manager Simple Linux Utility for Resource Management (SLURM) is used to execute and utilize the computing resources of the VSC. SLURM can be used to manage batch jobs and distribute them to the various nodes of the cluster. Therefore, the original version of the testbench is rewritten to provide interfaces to batch scripts. Various SBATCH parameters are defined, including the number of nodes, number of tasks, number of tasks in parallel on a single node, number of tasks for a single core, number of processors per task, number of GPUs, and the quality of service (QOS) and partition used. SBATCH starts a batch script for SLURM. The hardware used is selected with the partition and the QOS. This differs in the available number of nodes, architecture, number and type of CPU, number of cores per CPU, GPU, and the size of the RAM. The code line `#SBATCH -partition=zen2_0256_a40x2` is used to select a node with Zen2 CPUs and 2x NVidia A40 GPUs. The code line `#SBATCH -nodes=1` requests the number of nodes allocated to this job. Then modules are loaded. With these, the required libraries can be loaded. The exact version must be specified. An example for cuda: `module load cuda/11.8.0-gcc-9.5.0-67lesn4`. Afterward, the Python script for initializing the workbench is called with arguments.

Another point that helps to reduce the overall machine learning workload is the data parallelization offered by the VSC's resources. By using multiple nodes, many CPUs and GPUs can be used in parallel. These nodes can train, test, and validate multiple models in parallel without blocking each other or having to share resources. The clock skew is an array shift and, therefore, a very time-efficient operation that cannot be further optimized. With clock drift, however, the signal is resampled. The use of filters is more time-consuming. Since hundreds of resamplings are necessary for each training of a model, models with clock drift take longer to calculate. For this reason, all possible clock drifts were pre-calculated and saved in pickle files. If a specific sequence is to be resampled from a run, the precalculated value is loaded from the pickle file instead.

To save memory space, only every tenth epoch and the best epoch of the entire training run are saved. The accuracy and validation loss are saved for each epoch during training, as well as the number of correctly recognized activities and the total number of activities in each case. When testing the model, a file is created for each test subject with the individual predictions for each sample, which are then compared with the ground truth in the evaluation. Once the model has been successfully evaluated, the accuracy, the edit distance, and the three F1-scores are saved separately in a file.

The computational results presented have been achieved [in part] using the Vienna Scientific Cluster. [20225b]

4.10.6 Runtime Optimization for Time Discrepancy Calculations

Array values are modified when time discrepancies are applied. With clock skew, it is a simple index shift in the array by the number of samples of the respective clock skew. This operation is very time efficient. The clock drift is much more complex to calculate. The signal is upsampled, filtered, and downsampled again. Filter operations, in particular, can take a lot of time. This operation is not applied once but 750 times per epoch during the model's training in the case of a clock drift. Since several random sections are taken from each test file. This means that the training of the machine learning model takes a long time when a clock drift is applied.

Method	Epochs	Runtime	Accuracy
Original Implementation	29	15,978 seconds	63.07%
HDF5	16	27,563 seconds	61.92%
Pickle	16	2,114 seconds	64.95%

Table 4.4: Runtime comparison of the original implementation, HDF5 and pickle formats: The original implementation calculates how the time signal is modified by clock drift during training. In contrast, the other methods utilize pre-calculated data and access it during training.

In order to optimize the runtime, the signals were calculated and saved in advance with the applied clock drifts. When training the machine learning models, no predefined windows are used, but windows at random times in the signal. This means that for each file, every clock drift at every possible starting point was precalculated and saved. This results in a huge amount of data. In order to store these data volumes as efficiently as possible and to be able to call them up quickly, different file formats were compared with each other. The HDF5 and pickle file formats were selected for this purpose. HDF5 is a binary data format with which large amounts of numerical data can be stored [The25]. The original implementation was compared with the two pre-calculated file formats, see Table 4.4. A machine learning model with the complexity medium and temporal length 198 was used. The investigated clock skew is 20 ppm during the training of the model. In this comparison, random windows were taken for training. Therefore, they differ in the number of training epochs and in the accuracy. However, the runtime differences are clear. In contrast, a clock skew with 20 samples takes only 671 seconds to train with the same machine learning model.

HDF5 is a data format that can efficiently store thousands of datasets in one file. In the tests, however, pickle proved to be the fastest method, which is why all possible clock drifts are precalculated and saved with pickle.

CHAPTER 5

Evaluation

After completing the computations, a huge amount of results was collected from the various machine learning models. This data is analyzed and evaluated below and summarized in an understandable way to draw conclusions for the research questions and to better understand the existing problems with synchronization in HAR. The first two sections again explain the type of data used and how the data was collected. The last section is dedicated to evaluate the results of this diploma thesis.

5.1 Type of Data

One of the aims of this diploma thesis is to find out whether time discrepancy in general influences HAR, and in which ways the recognition is influenced. To achieve these goals, on the one hand, a dataset was used and prepared for my purposes, which contains exactly such HAR data, and on the other hand, a common machine learning network was implemented, which together belong to the most important part of the developed testbench.

The Opportunity Activity Recognition dataset used consists of more than 27,000 individual atomic activities and, thus, covers numerous complex ADL with a large network of many sensors. The five synchronized IMUs used on the motion jacket of the subjects are only a small part of the total of 72 sensors, which are composed of numerous sensor systems with body-worn sensors, object sensors, and other sensors in ambient space. In total, the dataset consists of more than 25 hours of sensor data. Each data point is also labeled in several ways, making it ideal for machine learning. Two different time discrepancies were applied to these synchronized data streams. They consist of constant changes in the time spectrum, which reflect the effect of timestamp absence that occurs in reality. An IMU only has a relative timestamp at the time at which the IMU was last started. Therefore, the sensor unit requests a timestamp from a reference system. However, unknown and variable processing delay times and network latency lead to delays and, thus, to an unknown clock skew of the data. The other time discrepancy investigated is a constant change in the frequency spectrum, which represents the effect of timestamp drift that appears in the real world. This effect is caused by a clock drift of the IMU worn on the body. In this case, the sensor unit itself generates a timestamp, but this is affected by a drift, which is usually caused by properties of the quartz crystal used for the clock or temperature fluctuations [XGMW17]. This leads to an unknown clock drift of the data. However, as these two time discrepancies often occur not only on their own but also in combination, this is also investigated. Especially when there is a clock drift, usually a clock skew of the data occurs at the same time.

The MS-TCN was used and implemented as the machine learning model in the test-bench. This network contains several stages, each of them performing dilated temporal convolutions and refining the first prediction stage by stage. The model is particularly suitable for temporal action segmentation and is utilized by numerous other publications [KNN23, WKDR+22, vAKLS23, YFG+24b, YFG+24a, DSY24, KHA+22, SDBV+23, SYN+23, ZVVC22] as a machine learning model for HAR. The time signals of the Opportunity dataset modified by controlled time discrepancies are used as input for the first stage of the MS-TCN.

5.2 Data Collection

The high-performance computing system of the Vienna Scientific Cluster was used to calculate 21,025 individual runs of the machine learning model. This allows the calculations to run faster than on an average desktop PC and, above all, on several nodes in parallel.

The results of the final testbench runs, some test runs, and further calculations for the final testbench were calculated with a total of 800,000 core-hours. One core-h is the use of one physical processing unit for one hour.

A total of 23,017 runs has been calculated. Out of the total number, 21,025 runs are used for further analysis. Some of the remaining runs were calculated with too many permutations for a time discrepancy variant or with a temporal length that does not match the receptive field of the machine learning model used. Of these, 5,887 models are of complexity “deep”, 7,569 models of complexity “medium”, and 7,569 models of complexity “shallow”. As much relevant data as possible was collected during the large number of individual runs to avoid wasting core-h unnecessarily. Several models of each computed machine learning model were stored. Among these are the models with the best training epoch as well as some additional training epochs. These are stored in such a way that they can continue to be used after the run, these models can be trained further and tested again. When testing the models, the predicted values of the individual tests are saved for each model so that they are still available later. In addition, the accuracy and validation loss of the training epochs are also stored. After validation, the final accuracy, the edit distance, and the F1-scores are stored. In addition, pickle files are created for the most important variables used. These contain, among other things, the parameters and settings for the machine learning models and the time discrepancies used. Furthermore, the time discrepancy used in each case, the parts of the machine learning model to which it is applied, and the distribution of the signals of the individual sensors are also saved.

5.3 Results

The data collected through numerous machine learning model runs is evaluated and assessed through various analyses in order to answer the research questions. The analyses include representations in confusion matrices and parallel coordinate plots, but above all, evaluations of the models' accuracies and their F1-scores.

The final amount of results data collected, which is used for further evaluation, includes 21,025 result folders with a total of 147,025 files. These files consist of 21,025 evaluation result files, 21,025 training result files, and 105,025 testing prediction files. In addition, there are also 21,025 pickle files, which contain the most important variables for subsequent calculations or checks, and 21,025 model folders in which the calculated epochs of the machine learning models are stored.

The models were calculated using 800,000 core-h at the VSC, as well as a large number of additional hours on a computer in the institute's laboratory. In total, the models were trained in 517,916 epochs. This means that the models were fed with a total of 388,437,000 inputs, whereby each input has a different temporal length of 198 to 3,300 individual samples. On average, each model was trained for 20 epochs, which is a total of 15,000 individual inputs to train one single model.

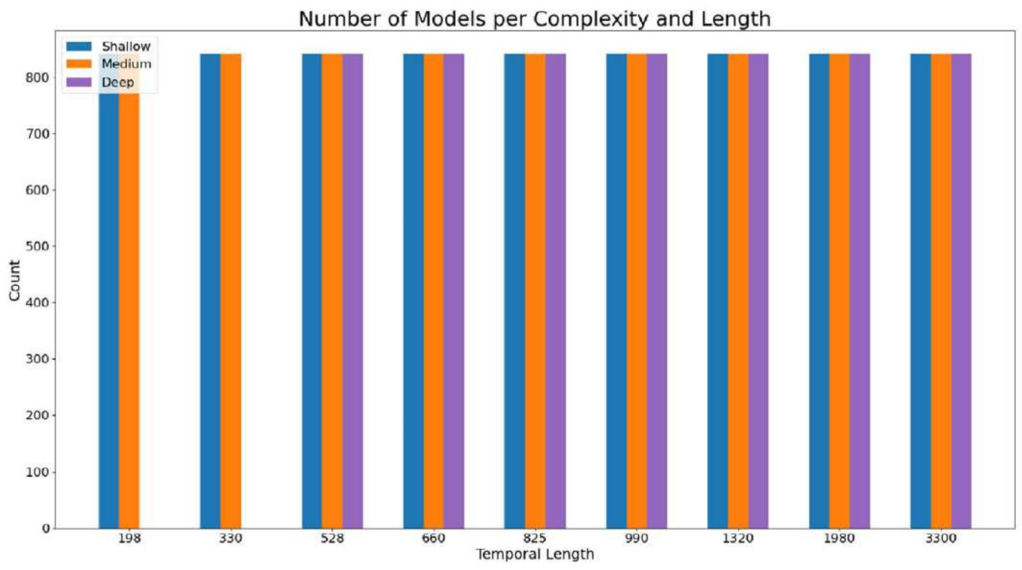
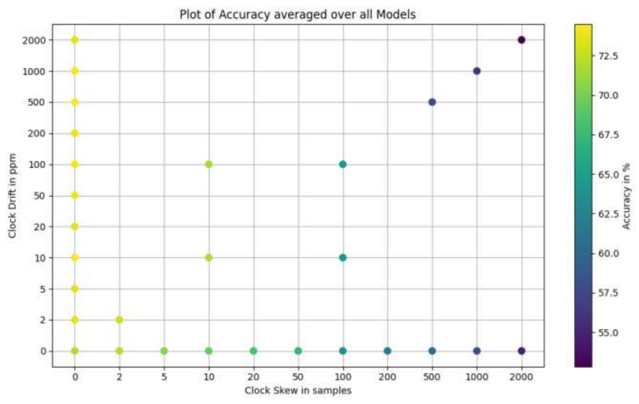
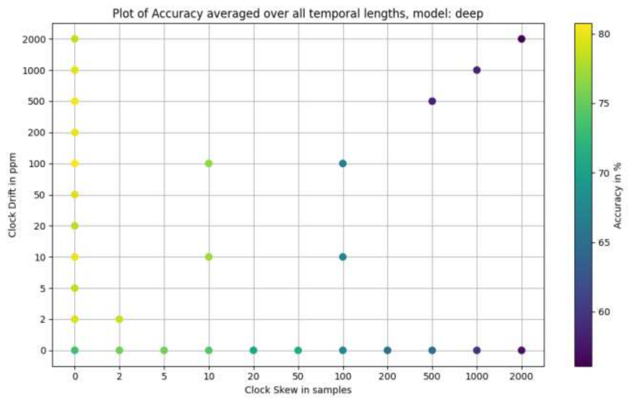


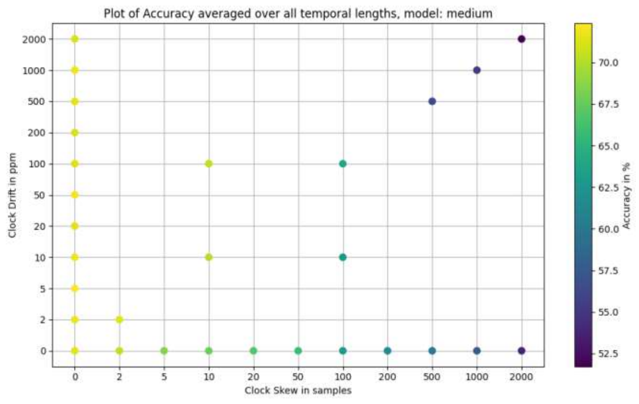
Figure 5.1: Number of calculated machine learning models per different complexity and temporal length.



(a) Accuracy for all time discrepancies averaged across all models.



(b) Accuracy for all time discrepancies averaged across all models with complexity deep.



(c) Accuracy for all time discrepancies averaged across all models with complexity medium.

Figure 5.2: Accuracies for all time discrepancies.

A total of 1245 individual SLURM jobs were started to calculate the models. Many of these were started simultaneously on several nodes using SLURM arrays. Some of these were merely test jobs for setting and finding the right parameters for the most efficient calculation. The models calculated in this way can be divided not only into different time discrepancies but also into different complexities and temporal lengths. The number of calculated models per complexity and length is shown in Figure 5.1.

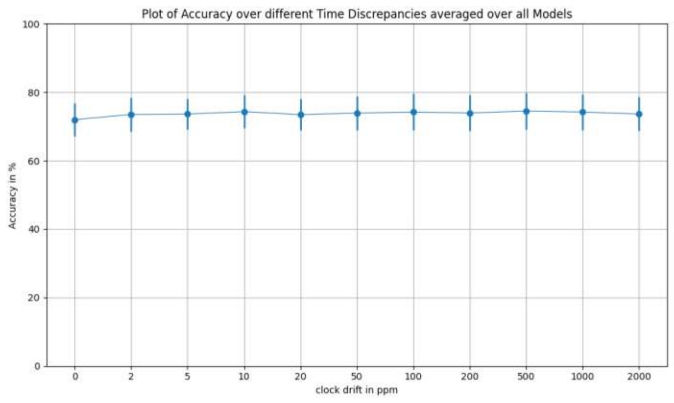
Figure 5.2a shows the accuracy for all applied time discrepancies. In this graph, the accuracy is averaged across all models. Similar results can be seen in Figure 5.2b and Figure 5.2c. However, here, the average is not formed over all models but rather only over all temporal lengths of the complexity deep and, respectively, the complexity medium.

Initial conclusions can be drawn from these graphs. Time discrepancy generally has an influence on the accuracy of machine learning models. For all three variants, the greater the time discrepancy, the lower the accuracy.
If there is only a clock drift, the accuracy is only slightly reduced. Especially with small clock drifts, the accuracy is very similar or even marginally higher than if there is no time discrepancy. But a slight reduction is still noticeable for the clock drift.

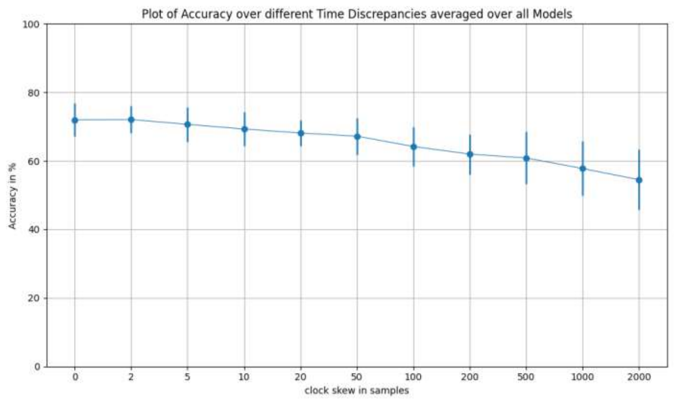
The clock skew has a much greater influence than the clock drift. The reduction is much more noticeable here. In the extreme case of 2,000 samples, the accuracy drops by around 15% on average across all calculated models. There is also a clear reduction in accuracy when both time discrepancies are applied simultaneously. Before reasons and explanations for the different accuracies are given, further graphs are analyzed.

The next figures take a closer look at the influence of the time discrepancy on the accuracy of the machine learning models. Figure 5.3a shows the accuracies averaged over the temporal lengths for the various clock drifts. In Figure 5.3b, the same means are applied across the different clock skews, and in Figure 5.3c, they are applied to combinations of the two time shifts. The different combinations are not directly related to each other. Therefore, they are of a sequence ordered by magnitude rather than a continuous progression on the x-axis. This is also why they are not connected by a line. Especially when applying clock skew, and the combination of both time discrepancies shows a drop in accuracy. Not only does the accuracy decrease, but the standard deviation also increases.

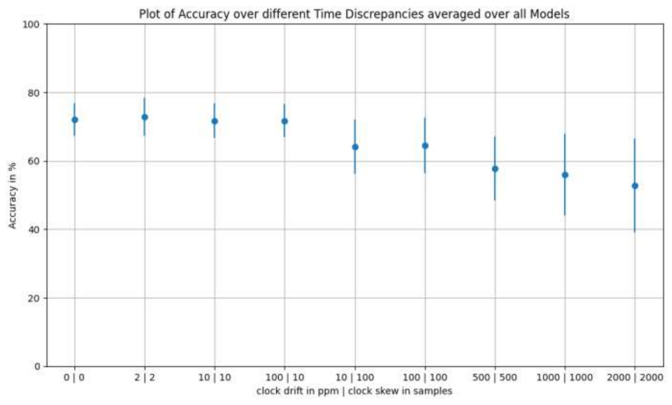
Figure 5.4a, Figure 5.4b, and Figure 5.4c show a breakdown of the various model complexities. Interestingly, in the model with the complexity deep, there is even a small increase in the accuracy for the smaller clock drifts. One reason can be that the applied time discrepancy causes activities to be recognized either later or earlier. However, because of the shift, samples are sometimes recognized correctly that would have been identified incorrectly without the applied time discrepancy. The lines are more or less constant across the various clock drifts. Only at 1,000 ppm and 2,000 ppm, a small reduction is recognizable. The deep models achieve an accuracy of 79.33% at 2 ppm and 78.51% at 2,000 ppm.



(a) Accuracy over clock drift in ppm averaged across all models.

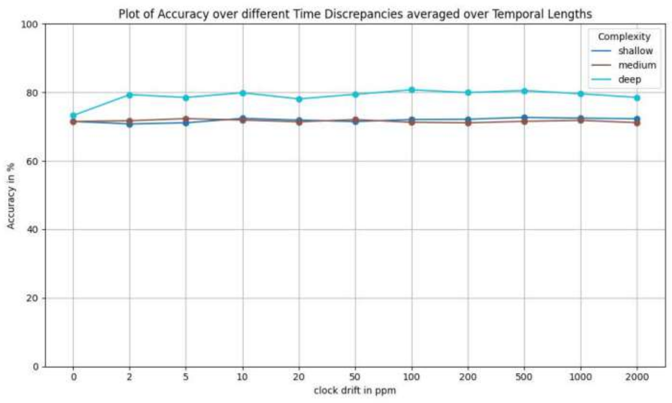


(b) Accuracy over clock skew in samples averaged across all models.

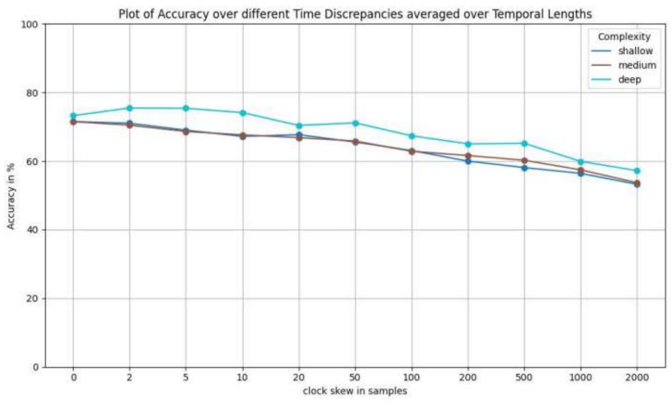


(c) Accuracy over combinations of clock drift in ppm and clock skew in samples.

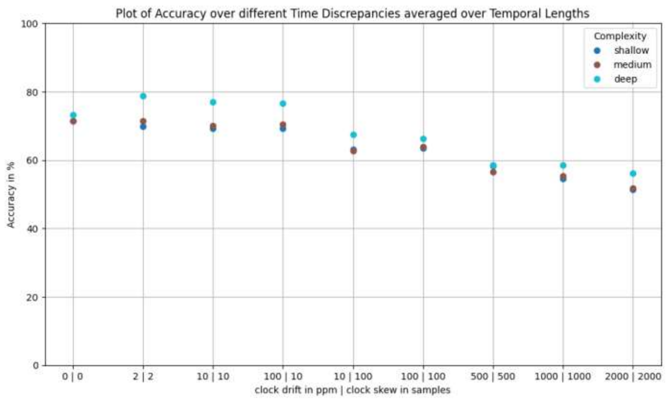
Figure 5.3: Accuracy over different combinations of clock drift in ppm and clock shift in samples averaged over all models.



(a) Accuracy over clock drift in ppm averaged across all models of the individual complexities.



(b) Accuracy over clock skew in samples averaged across all models of the individual complexities.



(c) Accuracy over combinations of clock drift in ppm and clock skew in samples averaged across all models of the individual complexities.

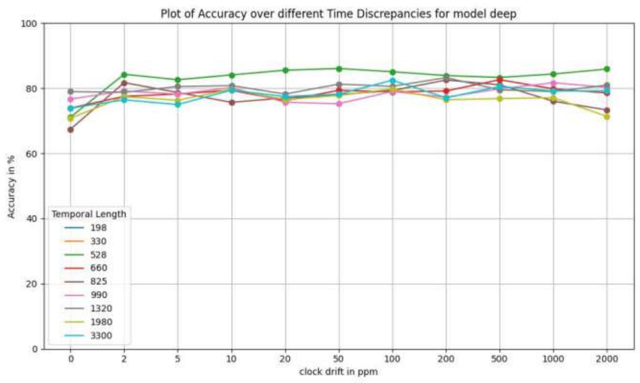
Figure 5.4: Accuracy over different combinations of clock drift in ppm and clock shift in samples averaged over all models of the individual complexities.

A rapid drop in accuracy can be seen much more clearly with shifts in the time domain. Already with a clock skew of 2 samples, a slight drop in accuracy occurs for the shallow and medium models. The accuracy is reduced more for each clock skew applied. The medium models achieve an accuracy of 71.51% at 0 samples. The accuracy is reduced to 70.47% at 2 samples and 68.64% at 5 samples.
With a clock skew of 1,000 samples, which is considered an unrealistic scenario, the accuracy even drops to 56.41% for the shallow models and 59.91% for the deep models. A sampling frequency of 33 Hz is used in the dataset. Accordingly, a temporal shift of 1,000 samples corresponds to 30.3 s. Especially in the area of HAR and detection of ADL, the temporal length is usually similar in size to this extreme time offset. Delays in receiving timestamps and sending data blocks, as well as different prioritization of task sequences, can theoretically lead to time delays in comparable dimensions to the temporal length of IMUs. However, in this visualization, even fewer samples of clock skew are sufficient to cause a clear reduction in accuracy.

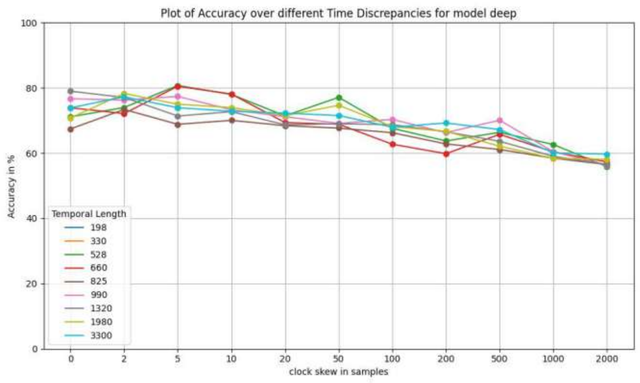
A drop in the accuracy of the machine learning models is particularly noticeable in the combinations of the two different time discrepancies. With the combination of a clock drift of 100 ppm and a clock skew of 10 samples, almost identical results were achieved for the model with complexity deep as with no time discrepancy. The accuracy for shallow models is reduced from 71.52% to 69.30%. For the reverse variant with a clock drift of 10 ppm and a clock skew of 100 samples, there is already a drop in accuracy to 63.13% for the models with the complexity shallow. A similar reduction is also present with 100 ppm. For the models with the complexity medium and deep, there is also a considerable reduction in the accuracy of the computed models. The accuracy with a clock drift and a clock skew is comparable for small time discrepancies when only a clock skew is applied. At 1,000 ppm and 1,000 samples, shallow models have an accuracy of 54.52%, and deep models achieve 58.46%.

The loss of accuracy can also be seen more in detail for the individual temporal lengths in Figure 5.5a, Figure 5.5b, Figure 5.5c. Here again the clock drift and the clock skew are shown separately, as well as their combinations.

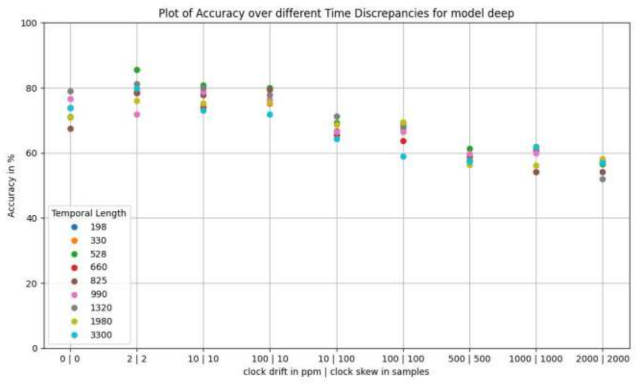
The accuracy is generally around 80% for all temporal lengths of the model deep for clock drifts of 500 ppm and below. For 2 ppm, the accuracy is lowest for the temporal length of 3,300 with 76.44% and highest for the temporal length of 528 with 84.29%. The accuracies are between 76.81% and 83.31% for 500ppm. When using no time discrepancy, the accuracy is between 67.40% and 78.98%. But there are also some lower values for some of the models with complexity medium and shallow, which are shown in the appendix.
The accuracies for clock drift are relatively constant across all temporal lengths. Only at 1,000 ppm and 2,000 ppm is a reduction recognizable for the temporal lengths 825 and 1980. The accuracy for the temporal length 825 drops from 81.06% at 500 ppm to 73.34% at 2,000 ppm. The accuracy for the temporal length 1980 drops from 76.81% at 500 ppm to 71.35% at 2,000 ppm. Some models are, therefore, already influenced by the clock drift of 2,000 ppm.



(a) Accuracy over clock drift in ppm for the different temporal lengths of the models with the complexity deep.



(b) Accuracy over clock skew in samples for the different temporal lengths of the models with the complexity deep.



(c) Accuracy over combinations of clock drift in ppm and clock skew in samples for the different temporal lengths of the models with the complexity deep.

Figure 5.5: Accuracy over different combinations of clock drift in ppm and clock shift in samples for the different temporal lengths of the models with the complexity deep.

While the accuracies for the various clock drifts are much more linear and consistent, the model curves for clock skew appear much more distributed. However, a general loss of performance is noticeable for each subsequent clock skew examined.
Similar results can still be achieved with a clock skew of 2 and 5 samples. But there is already a slight drop in accuracy with clock skews of 10 samples for some temporal lengths. The accuracy drops from 78.98% to 72.71% for models with the temporal length of 1320. With a shift of 20 samples, there is a reduction in accuracy to 68.68% for the temporal length of 1320.

The accuracies for the combinations of clock drift and clock skew are similarly widespread as the clock skews. However, a considerable reduction in performance can also be identified here across all temporal lengths. The behavior is again similar to that of clock skew only.

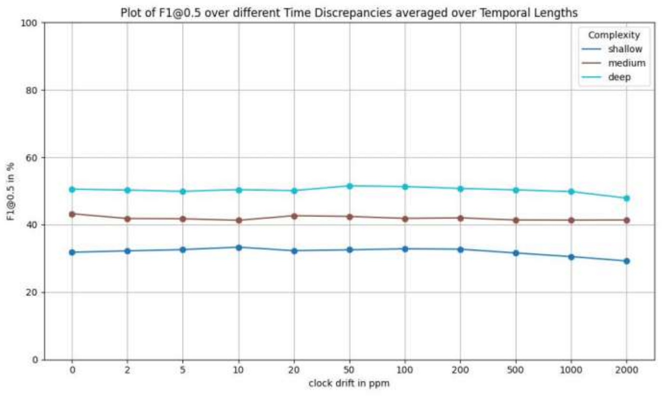
Similarly to the different accuracy figures, different graphs are now considered using F1-scores. As before, the different temporal lengths were averaged to obtain a general understanding of the influence of the different time discrepancies. Figure 5.6a, Figure 5.6b, and Figure 5.6c illustrate the F1@0.5-score for the different model complexities.

The threshold value k changes the F1@k-score. The F1@0.1-score is averaged to 55.21% for the various models without temporal discrepancies, 51.96% for the F1@0.25-score, and only 41.19% for the F1@0.5-score. Therefore, more positive values are found with the former. With the latter, the fewest positive values are found, but the predictions made for the positive values are more reliable. The F1-scores are again almost constant for the different clock drifts until they decrease slightly at 1,000 ppm, and more at 2,000 ppm. The F1@0.5-score decreases from 50.56% to 49.84% for 1,000 ppm and to 47.88% for 2,000 ppm.

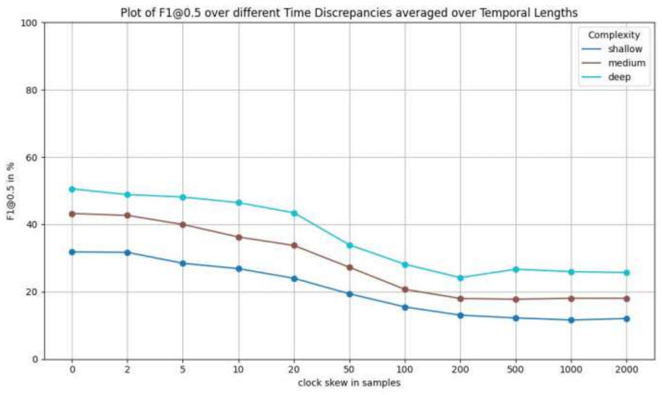
With the temporal shifts, there are also similarities with the accuracy figures. For 2 to 5 samples, only a slight reduction in all three F1-scores is observed. The F1@0.5-score for medium models decreases from 43.25% to 40.00% for 5 samples. It is 36.27% with 10 samples offset. Thereafter, the F1-score decreases clearly. At a clock skew of 100 samples, the F1@0.5-score is between 15.50% and 28.18% for the different complexities. A clock skew of 100 samples corresponds to a shift of 3.03 s at a sampling frequency of 33 Hz.

The combinations of the two time discrepancies are again more difficult to read, as each value is not directly related to the values next to it. However, there are also drastic drops in many models. With a clock drift of 2 ppm and a clock skew of 2 samples, the models still achieve nearly the same F1@0.5-scores. However, a clear loss of classification is already visible at 10 ppm and 10 samples for the deep model. The score lies between 25.39% and 43.02% for the different complexities. The models are already a lot less able to distinguish between the different classes. For a time discrepancy of 10 ppm and 100 samples, the F1@0.5-score drops to 21.35% for the complexity medium and 26.51% for the complexity deep. All three different F1-scores drop clearly.

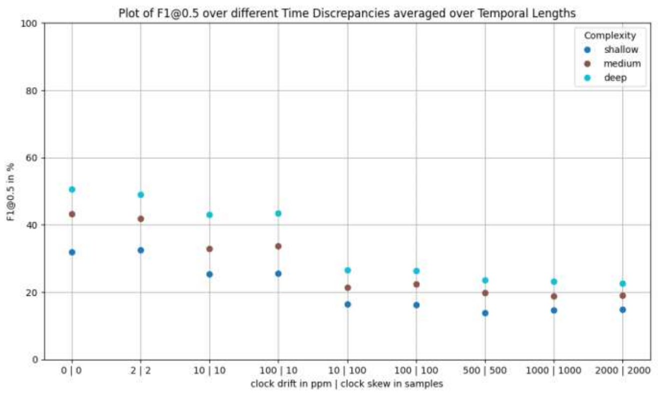
There are similarities in the behavior of the models with different time discrepancies



(a) F1@0.5-scores over clock drift in ppm averaged across all models of the individual complexities.



(b) F1@0.5-scores over clock skew in samples averaged across all models of the individual complexities.



(c) F1@0.5-scores over combinations of clock drift in ppm and clock skew in samples averaged across all models of the individual complexities.

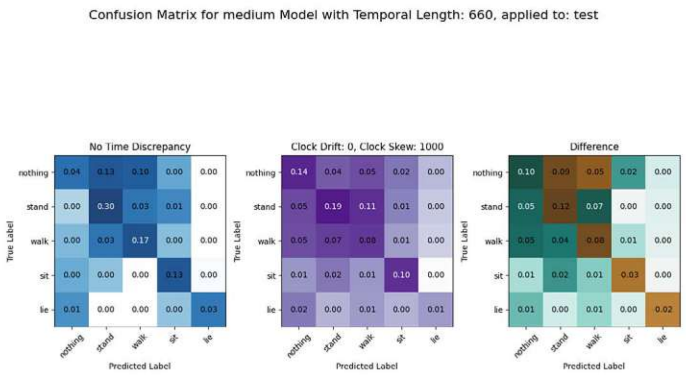
Figure 5.6: F1@0.5-scores over different combinations of clock drift in ppm and clock shift in samples averaged over all models of the individual complexities.

between the accuracy figures and the F1-score figures. However, the F1-score is more sensitive to time discrepancies than accuracy. The F1-score considers the distribution of data, making it particularly useful for datasets with unbalanced classes, such as the Opportunity dataset. This is an advantage of the F1-score over accuracy, which does not consider data distribution. Changes or actually only reductions are already visible with smaller discrepancies and, in particular, are also more easily recognizable than in the accuracy figures. For example, the F1-scores are already reduced with a clock skew of 5 samples. The situation is similar with the combinations of clock skew and clock drift.

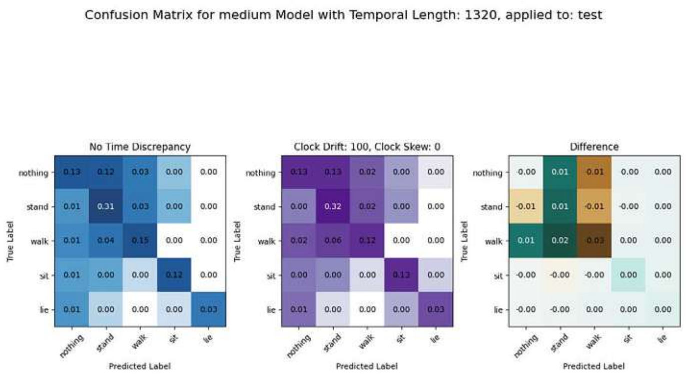
The first findings and conclusions can already be drawn from the figures and values analyzed so far. It is visible that time discrepancies have an impact on the accuracy of machine learning models. A time discrepancy unknown to the machine learning algorithm leads to a reduction in accuracy and the other evaluated parameters. Small deviations in the frequency spectrum have little to no influence on accuracy. Though even minor reductions in accuracy can be critical. This depends heavily on the application used. Larger discrepancies in the frequency range of 1,000 ppm and 2,000 ppm, however, already have a noticeable effect on the accuracy of such a system.
The deviations examined in the time spectrum have a greater influence on the accuracy, but these are also considerably more weighted. A direct comparison of the time discrepancies used in the frequency range and those in the time range is not possible. The investigated frequency changes are in the ppm range. For example, a change of 100 ppm at a sampling frequency of 33 Hz means a change in sampling frequency to 33.0033 Hz. The situation is completely different for the clock skews investigated, which are defined in samples. A change in time of 100 samples at the same sampling frequency corresponds to a shift in the signal of 3.03 seconds. Such a change is much more noticeable with small temporal lengths.

To support the previous findings, confusion matrices and parallel coordinate plots are analyzed below. The machine learning model with the complexity medium was used again. The confusion matrix for the respective model with no time discrepancy is shown on the left. In comparison, the same model with a time discrepancy is displayed on the right-hand side. Figure 5.7a shows a model with a temporal length of 660 and a clock skew of 1,000 samples. Figure 5.7b illustrates a model with a temporal length of 1,320 and a clock drift of 100 ppm. A combination of 100 ppm and 100 samples is visualized in Figure 5.7c with a temporal length of 3,300 samples. The confusion matrices are all given as a percentage of the total number of activities. The diagonal from the top left to the bottom right shows the true positive classifications. In an ideal network, all classified labels are on this diagonal. In each row, the remaining cells contain the false negative predicted labels. In each column, the false positive labels are located in the remaining cells.

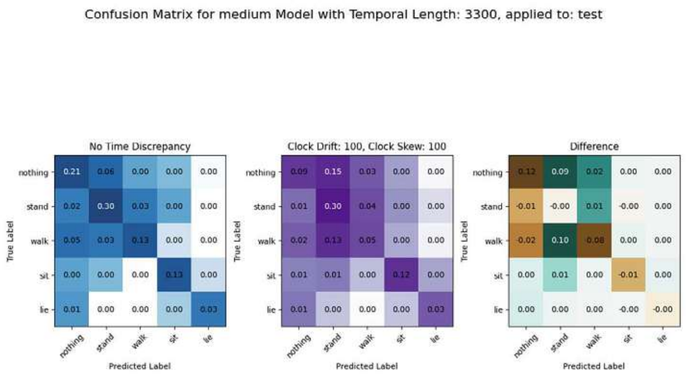
The research was never specifically aimed at outstanding classification performance, but instead at investigating the impact of time discrepancies. This results in some false positives and false negatives, even when there is no time discrepancy. When analyzing the matrix, it is noticeable that there are quite a lot of overlaps between the labels “stand”



(a) Confusion matrices for a model with complexity medium and temporal length of 660 with no time discrepancy and a clock skew of 1,000 samples.



(b) Confusion matrices for a model with complexity medium and temporal length of 1,320 with no time discrepancy and a clock drift of 100 ppm.



(c) Confusion matrices for a model with complexity medium and temporal length of 3,300 with no time discrepancy and a clock drift of 100 ppm and a clock skew of 100 samples.

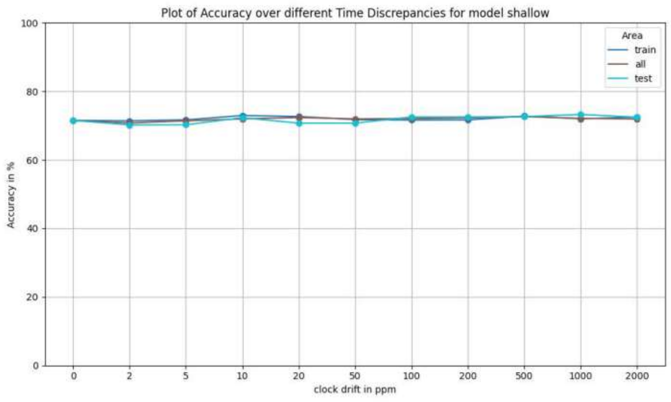
Figure 5.7: Different confusion matrices for a model with complexity medium with different temporal lengths and different time discrepancies.

and “walk”. These two activities, in particular, have many similarities, and it can be assumed that the activity “stand” precedes and follows each “walk” activity. Here it is difficult to determine when walking starts and how long it rather is still standing and vice versa. This is difficult both for the machine learning model and for the creators of the dataset, who labeled the data points manually. If the labeling is incorrect, but based on the sensor signals it is clear to the machine learning algorithm, an accuracy of 100% cannot be achieved because the machine learning model will always classify it differently than labeled.
Additionally, the label “nothing” is often incorrectly classified. This label was assigned if no other activity could be matched to it. Realistically, there can also be overlaps here, for example, if a walk was not labeled as a walk but as nothing by mistake. In the first confusion matrix, Figure 5.7a, with only the clock skew, this is the case. The number of true positives is lower for both the label “stand” and the label “walk”. In return, there are more false negative and false positive classifications of the respective other activity. In addition, there are also more false negative classifications of the two activities with the label “nothing”. However, there are even more true positive cases of “nothing” and fewer false negative cases of this. In the second confusion matrix, Figure 5.7b, there are only slight differences of up to 3%. As the previous results have already shown, a clock drift of 100 ppm has hardly any effect on accuracy or F1-scores. Accordingly, the two matrices agree to a large extent here. The largest deviations are again between walking and standing.

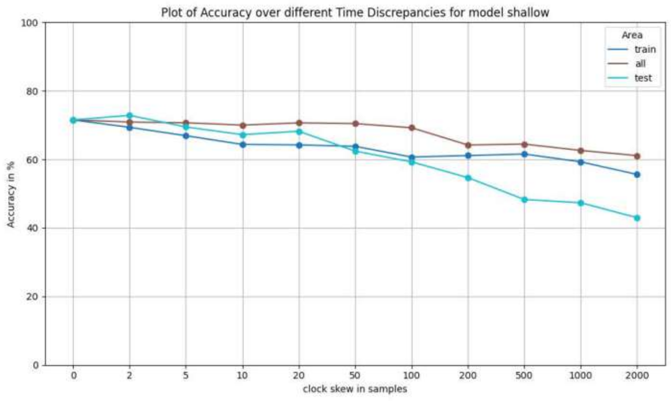
More differences can be identified in the last comparison of the confusion matrices. Here again, the clock skew of 100 samples has a clear negative influence. As in the previous figures, there are hardly any discrepancies in the labels “sit” and “lie”. These two labels are recognized equally well across all the matrices listed here. However, there is again a clear decrease in true positive classifications for the remaining activities. The activity “walk” is recognized 8% less correctly. There is also 12% less correct detection of the activity “nothing”.

The models with complexity deep achieve an accuracy of between 67.40% and 78.98% with no time discrepancy. This decreases up to 27% when time discrepancies are applied. In the models with complexity medium, the values for the temporal length of 3,300 samples fall the most, from 78.89% to 53.12% for a time discrepancy of 2,000 ppm and 2,000 samples. The F1@0.1-score halves for some temporal discrepancies. It decreases from 56.67% to 25.38% for a clock skew of 2,000 samples using medium models with a temporal length of 198. A reduction by half can also be observed for the F1@0.25-score. The F1@0.5-score drops even more, from 31.84% to only 12.04% for all models of complexity shallow.

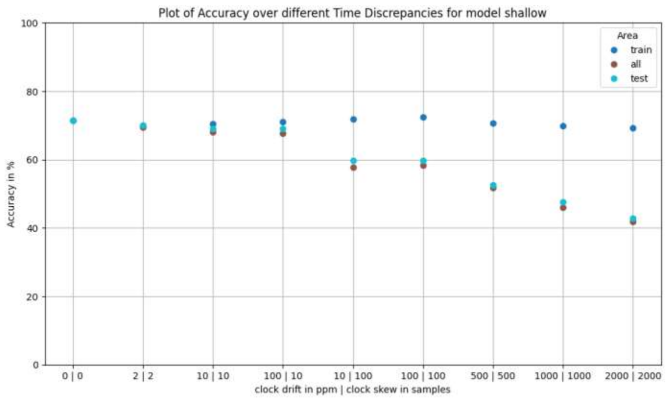
The accuracy of the different application areas has been averaged so far. However, the influence of time discrepancies during training, testing, and both should also be analyzed. Figure 5.8a, Figure 5.8b, and Figure 5.8c show models with the complexity shallow. They are averaged over the temporal length. The data is divided into the areas where the time discrepancy is applied. In the “train” domain, the time discrepancies are only applied during the training of the model, and in the “test” domain, only during the testing of



(a) Accuracy over clock drift in ppm divided into the different areas for models with the complexity shallow.



(b) Accuracy over clock skew in samples divided into the different areas for models with the complexity shallow.



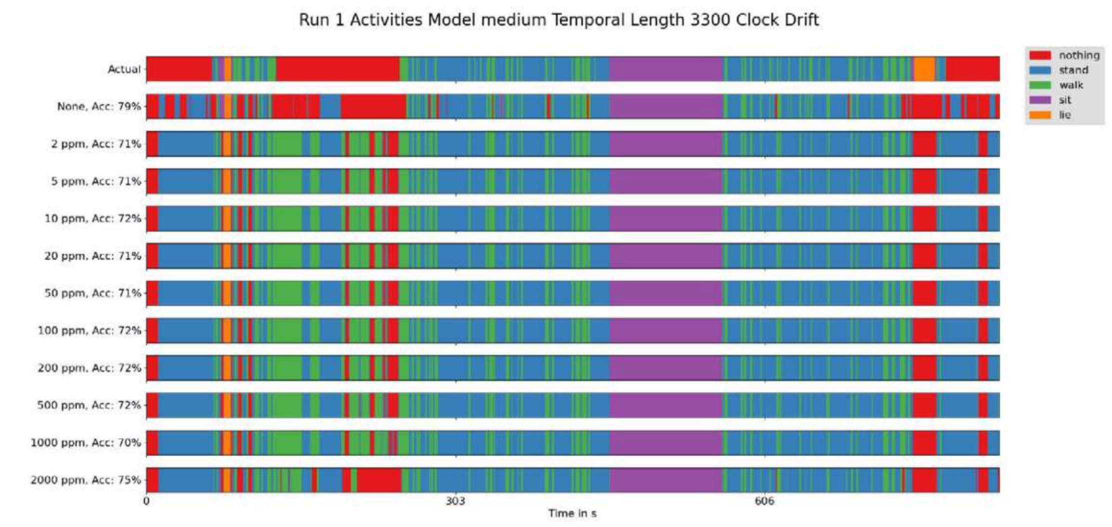
(c) Accuracy over combinations of clock drift in ppm and clock skew in samples divided into the different areas for models with the complexity shallow.

Figure 5.8: Accuracy over different combinations of clock drift in ppm and clock shift in samples divided into the different areas for models with the complexity shallow.

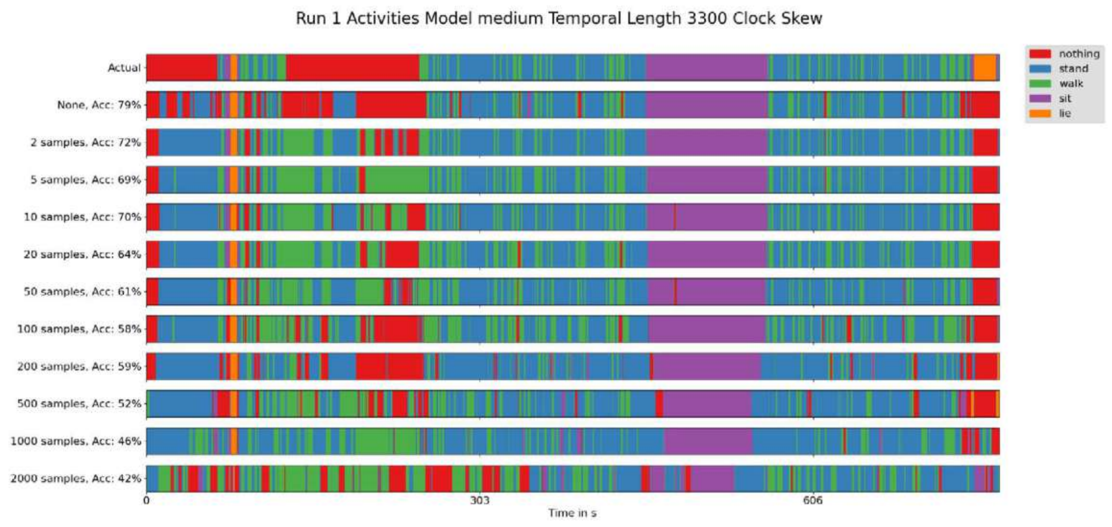
the model. In the “all” section, the time discrepancies are applied both during training and during testing of the model. When using only clock drift, there is no effect on the accuracy. The clock skew has an effect on the accuracy in all three areas. If the same time discrepancy is applied to both training and testing of a model, the accuracy remains constant at the beginning. However, from a skew of 100 samples, the accuracy starts to decrease. The accuracy is 64.20% for a clock skew of 200 samples. If the clock skew only occurs during training, the accuracy drops for each clock skew tested. The effects are most noticeable if the clock skew only occurs during testing. With 10 samples, the accuracy decreases from 71.52% to 67.23%, and with 500 samples, it is only 54.67%. The area “test” and “all” behave in the same way when using clock drift and clock skew as the area “test” with only clock skew. While the area test, therefore, behaves in the same way, the area “all’ worsens when both time discrepancies are applied. The area “train” remains almost constant when both time discrepancies are applied. It decreases from 71.52% to 69.30% at 2000 ppm and 2000 samples.

In order to discuss possible reasons why the accuracy does not drop even more with the larger time discrepancies, the activities performed from the first run of subject 4 are considered over time. The top bar is the ground truth and, thus, the activities actually performed. The second bar is the recognition of the machine learning model if no time discrepancy is applied. Below are the recognized activities for different time discrepancies. In Figure 5.9a different applied clock drifts and in Figure 5.9b different clock skews are applied. Activities that occur only for a short time are increasingly being confused with larger time discrepancies. This is particularly the case between the two activities “stand” and “walk”. However, the “sit” activity appears so distinct from the other activities that it can always be easily recognized. The long section of the “sit” activity in the second third of the run increases the overall accuracy. As this accounts for approx. 10% of the entire time signal. Only for the larger clock skews can it be seen that the beginning of this activity is always detected later and the end, respectively, earlier. This is due to the large time shift in the time signals. Another aspect that stands out in the analysis of the temporal sequence of the activities is the poor detection of the “nothing” activity at the beginning of the time signal. Here, the activities “stand” and “walk” are usually classified instead. This could be related to the fact that the category “nothing” is not really a locomotion activity at all. Rather, the creators of the dataset were unable to assign any of the other labels to the time periods here. However, there may be overlaps with the other activities for the time signals of the sensors.

Finally, two parallel coordinate plots are also considered to support the insights already gained. These differ in their last axis, which is the accuracy, see Figure 5.10a, and the F1@0.5-score, see Figure 5.10b. The time discrepancies are plotted on one axis to make the effects on the evaluation parameter clearer. The order on the axis was chosen so that models with a higher accuracy are placed above models with a lower accuracy. These plots lead to similar conclusions as those that could already be drawn from the plots with the time discrepancy. The accuracy is very high if only a clock drift is used, but again the accuracy decreases the larger the clock drift becomes.

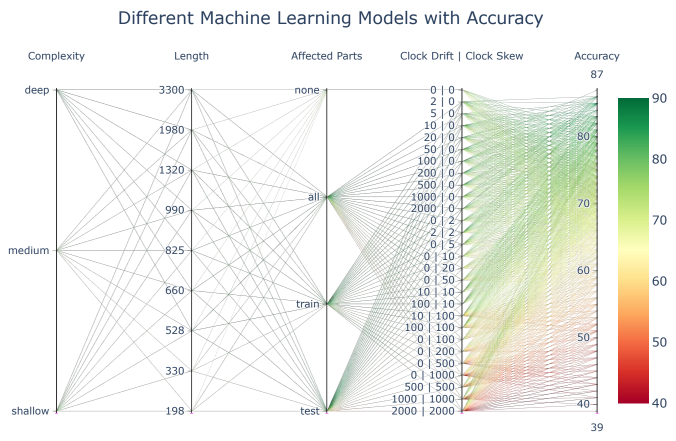


(a) Comparison of the activities actually performed with the predictions of models of complexity medium and temporal length 3300 at different clock drifts. The activities of the first run of the fourth subject are shown here. The first line shows the actual labels for the corresponding timestamps.

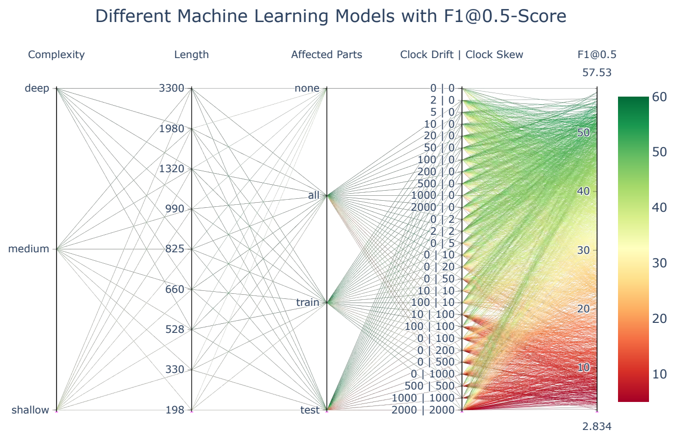


(b) Comparison of the activities actually performed with the predictions of models of complexity medium and temporal length 3300 at different clock skews. The activities of the first run of the fourth subject are shown here. The first line shows the actual labels for the corresponding timestamps.

Figure 5.9: Comparison of the actual activities of subject 4 run 1 with the predictions of models of complexity medium and temporal length 3300.



(a) A parallel coordinate plot is shown here. The first axis displays the different complexities of the machine learning models and the second axis shows the corresponding temporal lengths of the inputs for the respective models. The third axis illustrates which part of the machine learning was affected by temporal discrepancies. The fourth axis shows the various time discrepancies with clock drift and clock skew. The last axis represents the accuracy of the models. It can be seen that the accuracy decreases for larger time discrepancies across all models.



(b) A parallel coordinate plot is shown here. The first axis displays the different complexities of the machine learning models and the second axis shows the corresponding temporal lengths of the inputs for the respective models. The third axis illustrates which part of the machine learning was affected by temporal discrepancies. The fourth axis shows the various time discrepancies with clock drift and clock skew. The last axis represents the F1@0.5-score of the models. It can be seen that the F1@0.5-score decreases for larger time discrepancies across all models.

Figure 5.10: Different parallel coordinate plot for all calculated models.

Even small clock skews lead to considerably less accurate results. A deterioration in accuracy is noticeable, at least for some temporal lengths, at the small shift of 5 samples. As soon as the clock skew reaches the range of 100 samples, i.e. a few seconds, the accuracy drops by 25% for many models. The machine learning models also deliver poor accuracies for combinations of the two time discrepancies.

A similar picture can also be observed for the parallel coordinated plot with the F1-score. However, there is a considerably higher scatter for individual time discrepancies. This is because all models, regardless of their different complexities and temporal lengths, are presented together along the "Clock Drift | Clock Skew" axis. As already mentioned, there are already large differences in the F1-scores for the various models even when there is no clock skew.

The aim of this work was not to create a perfectly performing machine learning model. Even the models that were not affected by any time discrepancy only achieved an accuracy of 63.09% (medium models with temporal length of 198) to 79.78% (medium models with temporal length of 1980). This, however, only plays a secondary role, as the focus is much more on the relative difference and, in particular, the reduction in accuracy when applying various time discrepancies. The original MS-TCN implementation achieved an accuracy ranging from 65.1% to 80.7% across different video-based datasets [FG19]. Using cameras and IMUs an accuracy of 83% was achieved for capturing eating and drinking activities [KNN23]. A video-only solution achieved an accuracy of 90.3% using the 50salads dataset [vAKLS23]. The accuracies presented in this thesis fall within the range of those achieved by other papers. However, a less accurate performance of the machine learning models can be observed for the majority of the time discrepancies used. It can, therefore, be concluded that time discrepancies have a clear impact on machine learning models for HAR.

The experiments conducted show that a clock skew of just 5 samples is sufficient to cause a loss in the performance of a machine learning model. For some models, even a clock skew of 2 samples leads to a reduction in the accuracy and F1-score. This corresponds to a clock skew of 0.06 to 0.15 seconds. Models with complexity medium and a temporal length of 1980 have a decrease in accuracy from 79.78% to 71.97% for a clock skew of 2 samples. Models with complexity shallow and a temporal length of 330 have a decrease in accuracy from 77.75% to 68.18% for a clock skew of 5 samples. The F1@0.25-score decreases from 57.99% to 51.29% for a clock skew of 2 samples and to 44.20% for a clock skew of 5 samples using models with complexity medium and a temporal length of 990. When the clock skew is only applied during training, the decrease in accuracy is even more visible. The biggest absolute decreases for 5 samples are: 10.54% (shallow with temporal length of 330), 10.21% (medium with temporal length of 3300), 12.56% (deep with temporal length of 1320). Averaged over the different temporal lengths, medium models decrease the most for 5 samples, with an absolute decrease of 4.44%. However, this results in 93.79% relative to the accuracy with no time discrepancy.

Clock drifts below 1000 ppm have no effect on model performance. This is partly due to the small temporal input length in combination with the small clock drifts. With a temporal length of 3,300 samples, a clock drift of 100 ppm shrinks or stretches the signal by 0.33 samples. This results in only minimal changes to the inputs. A poorer performance is visible for some models with clock drifts of 1,000 ppm and 2,000 ppm. The much bigger issue with clock drifts is the associated clock skew. With a sampling frequency of 33 Hz and a clock drift of $\pm 100\,ppm$, there is a clock skew of 0.36 seconds after one hour if no other synchronization takes place. This clock skew corresponds to a shift of 11.9 samples in the conducted experiments. Such a shift leads to considerably less accurate model performance.

The experiments also show a clear difference depending on where the time discrepancy is applied. With clock skews of up to 100 samples, the accuracy is constant if the same time discrepancy is present during training and testing. The worst performing models are those that are trained with good training data but have a time discrepancy during testing.

CHAPTER 6

Results and Discussion

The previous chapter has provided an overview of the findings gathered in this diploma thesis and highlighted the important results. The following chapter compares the outcome with existing related work and discusses the findings and open issues. In addition, the research questions are explicitly answered and the results obtained are critically reflected upon.

6.1 Comparison with Related Work

Relevant literature that explicitly deals with the influence of time discrepancies on machine learning models for HAR hardly exists. Despite the importance, relevance, and influence of this topic shown in this diploma thesis, only a few research papers [XGMW17, XGM17, Ma18] deal with the existing problem. HAR is a relevant topic, especially in the context of the recognition of ADL. However, the important topic of synchronization often plays a minor role, despite being one of the principal challenges of multi sensor systems [XGM17, XGMW17, WSL^{+}19]. Nevertheless, as this thesis has shown, the existing research on synchronization between body-worn sensors should not be neglected.

Using event-based synchronization, a synchronization error of 0.3 seconds can be achieved between multiple devices. At the same time, the authors also highlight that good synchronization is the key factor for good performance and that it is not trivial in real-world applications. Nevertheless, the authors assume that an accuracy of less than one second is sufficient for the detection of simple activities. [BAL09] Another approach of clock synchronization between several independent sensors has been developed by another research group. Using another event-based approach, an error of 1.31 seconds could be achieved with this method. Another approach in which NTP was used, resulted in an error of 0.25 seconds. With a combination of both methods, the timing error could even be reduced to 0.02 seconds. [WSL^{+}19] Low synchronization errors can also be achieved using magnetometer-based methods. The signals of the individual sensors are aligned with magnetic pulses. This achieves an error of 0.036 seconds. However, constant distances to the pulse generators are required here, which is difficult to use in practice when detecting ADL. [GLD$^+$24] Therefore, there are certainly synchronization methods that can achieve high levels of accuracy.

This thesis was able to show that a maximum clock skew of 0.06 to 0.15 seconds between the signals of the different sensors worn on the body should not be exceeded for good machine learning model performance. Accuracy decreases clearly with higher time errors. For some temporal lengths, the accuracy drops by 12.56%, even at 0.15 seconds, if the time discrepancy occurs only during the testing of the machine learning model. Compared to the initial value without any time discrepancy, this is a reduction of 15.90%. The largest reduction averaged across all areas is 12.31%. Averaged across all models, the drop at 5 samples is not as severe but is already present. Accordingly, the statement that an accuracy of less than one second is sufficient for good recognition of HAR, especially with regard to ADL, is refined as follows. For successful and reliable detection of ADL, a clock skew between the time signals of the individual sensors of 0.06 to 0.15 seconds should not be exceeded. Synchronization is not only important in the time domain, but the synchronization of different, spatially separated clocks is just as crucial and necessary for reliable model performance. The accuracy of the timestamps of an IMU depends on the clock frequencies. Clocks are generated with oscillators, which have various crystals integrated. Manufacturers usually use quartz crystals with just enough accuracy to reduce costs. Therefore, the accuracy of an IMU depends on the purpose for which it was developed. Different technologies with different accuracies are used. A classification can be made as follows: CMOS: $\pm 500\,pm$, XO: $\pm 20\,pm$, TCXO: $\pm 5\,pm$. [TAA19] Oscillators with higher accuracy have higher energy consumption and costs [BDE13]. Therefore, developers of commercially available devices are often forced to make compromises and do not implement the most accurate clock technology, which increases the effects of drift [TAA19].
In this thesis, it could be shown that such production-related deviations and inaccuracies, in the range of up to 500 ppm, have little to no effect on the model performance. Nonetheless, additional measures must be taken for synchronization so that the error caused by the clock drift does not increase cumulatively. An IMU with a CMOS clock and a sampling frequency of 33 Hz has a clock skew of 1.8 seconds after one hour of operation without special synchronization. This corresponds to 59.4 samples. In other words, there is a loss of model performance after just one hour of operation.

6.2 Answers to Research Questions

In order to answer RQ1 “How do time discrepancies impact the performance of machine learning-based human activity recognition?”, a testbench was developed that is able to apply different types, sizes, and combinations of time discrepancies to a dataset that includes different ADL. The testbench utilized the Opportunity dataset. It is synchronized through post-processing. However, the IMUs used are connected by cable to a processing unit, which collects and transmits the data synchronously. This means that the IMUs used are synchronized with each other. This type of synchronization is sufficient for the experiments carried out. In addition, the results of the time discrepancies were always seen in relation to the results without time discrepancies. 21,025 machine learning models were trained using this modified input data. Afterward, the resulting changes in the accuracy and the F1-scores of these models were analyzed. In order to be able to make a meaningful statement, not only were a few predefined sensors subjected to a time discrepancy, but it was checked whether it is sufficient to examine a random assignment of sensors to a time discrepancy or whether all possible permutations of sensors to time discrepancies are necessary. Additionally, it was analyzed whether using a certain number of permutations in between is already sufficient. By analyzing an example model in which all possible permutations for a pure shift in time and a pure change in frequency were played through, the following statement can, therefore, be made. Just 10 possible sensor assignments are sufficient to make a well-founded statement about all theoretically possible sensor assignments.

In the implementation, the various time discrepancies were applied to the individual, small time sections, which were then used for training and testing, instead of subjecting the entire, long signal to a time discrepancy. This would be more in line with reality, as the clock inaccuracy of an IMU, for example, has an accumulative effect on the timestamps if the IMU is in operation for a longer period of time. However, this procedure not only involves a change in frequency but also indirectly induces a clock skew through the frequency change. Nevertheless, as both time discrepancies should also be tested separately, the individual time sections were changed first. If a signal of a sensor is affected by a clock drift, the frequency of the entire signal is changed. The signal is either squeezed or stretched. If a time section is removed from this distorted signal which does not start at 0, there is also a clock skew at the beginning of the section in addition to the clock drift within the time section, as the section is changed by the relative frequency change in time. With this context, the following could be determined. Time discrepancy influences the accuracy and all other evaluation parameters of machine learning models. This reduces the recognition of human activities. However, not only are activities less well recognized, but there are also more false recognitions of activities. Both false positive and false negative classifications occur more frequently. It is important to differentiate between the various time discrepancies examined. Clock skews are particularly critical, while clock drifts can be handled on their own. Nevertheless, all of the time discrepancies can cause a loss of both accuracy and F1-scores. It is not only the type and magnitude of time discrepancies that lead to poorer performance of machine learning models. It also depends on when and where the time discrepancies corrupt the data. If a model is trained with altered data, this model is biased from the beginning and can therefore also deal better with altered test data. The models performed best when the same time discrepancy was applied during testing and training. The accuracy is constant with smaller clock skews. However, performance also drops with clock skews of 100 samples or more. The worst models are those that are trained with correct time signals but are then exposed to a time discrepancy during testing. However, this type of time discrepancy is probably one of the most common in practice. This can occur when training consumer products takes place under optimal laboratory conditions, but time discrepancies arise during normal use, i.e., during testing. These time discrepancies during testing could, for example, be caused by cheaper hardware and clocks or by environmental influences.

For RQ2 "Which temporal accuracy is required for accurate human activity recognition using machine learning models?", it was investigated at what point certain time discrepancies have an effect on the evaluation parameters. With a clock skew between the signals of the sensors of 2 to 5 samples, effects can be seen. While the average accuracy only decreases by 1.34%, some models are reduced by up to 12.31%. When the clock skew is only applied during testing, it is even up to 12.56% reduced, the F1-scores are already up to 22.43% lower. This means that synchronization that allows a maximum time error of 0.06 to 0.15 seconds is required between the time signals of the individual sensors. Clock drifts alone are feasible, as the commonly used quartz oscillators are accurate enough to detect ADL. It is important that the time stamps are synchronized regularly, as otherwise, the clock drift just mentioned can lead to a problematic clock skew. If the existing frequency inaccuracies are not prevented from accumulating, problematic clock skews can quickly occur even with high-quality crystal oscillators.

6.3 Limitations

It was a challenge to find a dataset with many synchronous signal channels. The dataset had to contain data on ADL and needed to be labeled in order to be usable for machine learning. However, the point of synchronization alone posed a challenge. Since the time signals are manually changed in order to achieve targeted non-synchronization, ideally synchronous data is needed. The Opportunity dataset contains a large number of different sensors, even more individual features, and has very high-quality data labeling. The sensor data was synchronized using post-processing. However, there are five body-worn sensors that are not connected wirelessly to a computer but are first transmitted wired to another body-worn data unit. These five sensors are supposed to be synchronized. A limitation of the Opportunity dataset is the rather uneven distribution of the individual activities during the respective runs. The different runs are not only in drastically different lengths but some of the activities occur at similar times and with different frequencies. The activity "lie" never occurs more often than 8% of the total runs and is even less than 3% several times, which means five different activities or six activities if one counts the non-assignability of "nothing" to an activity. The "stand" activity is also very dominant, accounting for up to 44% of a run. In addition, the activity "sit", for example, is present in every run in the second half for a longer period of time. This is not an optimal distribution for training machine learning models. It can be assumed that the accuracy of the models under the influence of time discrepancy is only so high because the "sit" section is very long and unique. If an activity is almost always recognized correctly, this raises the accuracy. There are almost no incorrect interpretations of the models here. The beginning and the end are recognized incorrectly only in the case of large time discrepancies. It seems quite plausible that the most frequent incorrect assessment is a wrong classification of standing and walking. These activities are often very similar for sensors and very often occur alternately in short periods of time in the dataset. Due to time discrepancies, these short sections are shifted and thus incorrectly recognized. It is also difficult for both humans and a machine learning algorithm to determine the exact point at which a person stops standing and starts walking or vice versa. Additionally, all the sensors used are placed on the upper body or on the arms of the person. This makes it even more difficult to determine the beginning or end of a walking action. The recognition of the locomotion activities and their accuracy is also made more difficult by the dominance of the label “nothing” at the beginning of the time signals. The creators of the dataset were unable to assign a label here. However, according to the machine learning models, the time signals of the sensor recordings are mostly similar to “stand” and “walk”, and therefore there are many incorrect classifications in this area.

CHAPTER 7

Conclusions

In this diploma thesis, important insights into the effects of time discrepancies on the performance of HAR using machine learning are gathered. Initially, a broad understanding of the different time discrepancies that occur in real-world applications is established. The first relevant time discrepancy is the timestamp drift. This phenomenon is caused by a drift in the clock signal, which can either squeeze or stretch the time signal of a sensor. The second important time discrepancy is caused by timestamp absence. In this case, a device does not have its own absolute timestamp but receives one from a reference unit. Due to unknown and variable changes in the network latency and processing delays of all participating devices, this results in signal delays. These two phenomena are referred to in the paper as clock drift and clock skew and were initially considered independently of each other. Since both also occur simultaneously in real systems and are not completely independent of each other, they were also used in combination to analyze their combined effects.

A testbench was developed to test whether and how the just mentioned temporal effects impact a machine learning model. With the help of this testbench, various time discrepancies can be applied to the synchronous Opportunity dataset. By defining various parameters, different machine learning models and different time discrepancies were tested automatically. With many different variants, a broad generalizability for other comparable machine learning models and other ADL can be established. The stages, layers, and temporal lengths of the implemented MS-TCN model were varied. By applying each time discrepancy to several random sensor assignments and averaging the results obtained, specific anomalies in the results can be excluded. As the standard deviation is greatly reduced after just a few used permutations, only ten sensor assignments are sufficient for a well-founded statement. The other parameters for the various testbench runs are the different applied time discrepancies and the areas that are influenced by the discrepancies. The areas differ in training, testing, and both. The aim was to investigate how a synchronously trained machine learning model behaves when tested with data that has been subjected to time discrepancy. In this way, the phenomenon that occurs in the practice of a model being trained in perfect laboratory conditions but then being used on low quality sensors and conditions was investigated. In the second case, it was analyzed how a model already trained with data affected by time discrepancies behaves when it is subsequently trained with synchronous data. Finally, the effects of time discrepancies in both training and testing of a model were also observed. Due to the limited computing time, only variants of the first case could be analyzed for the most part.

In the end, a big challenge of this thesis was the available computing time. To achieve broad generalizability, many machine learning models with different parameters were defined. This led to a very large number of iterations to be carried out. As a result, the high-performance computers of the VSC were utilized to calculate the 21,025 individual machine learning models. In addition, the individual runs were noticeably optimized in their runtime efficiency. The clock drift, which is complex to calculate, was pre-calculated for the various values used. This means that more disk accesses and accesses to large dictionaries were necessary, but these were many times faster than the calculation of the clock drift during the training of the model.

Finally, recommendations can be given for the required time accuracy. This research provides insightful findings on the effects of time discrepancies in machine learning models. Clock skews between the signals of the individual sensors used should not exceed 0.06 to 0.15 seconds. Otherwise, reliable recognition of activities can no longer be ensured. Furthermore, a regular synchronization of the clocks must take place to avoid clock drifts. Clock inaccuracies of state-of-the-art quartz oscillators have no effect on the accuracy of detection. However, the clock skew caused by the clock drift can lead to performance losses even after short sensor runtimes.

7.1 Future Work

This diploma thesis provides many important insights into the influence of time discrepancies on machine learning models for HAR. Nevertheless, further aspects remain open for future research. Potential future work should focus on more different ADL. So that the statements made in this work can be confirmed with an even broader spectrum. In addition, the problem should be examined using several different synchronized datasets. Different variants of the MS-TCN model were examined. However, in order to rule out a general bias of this machine learning model, other models could also be investigated here. Furthermore, an even larger testbench can be developed to analyze even more real-world occurrences. Moreover, future work should not only focus on identifying the problems. Rather, once the problems have been successfully pinpointed, the focus should be on solving the issues. Synchronization is and remains a crucial factor in machine learning and activity recognition. Even though the synchronization of numerous, separated body-worn devices is complex, it plays a key role in successful accuracy and recognition. Especially, the regular synchronization of time stamps or, rather, the clock itself is important to avoid clock skew due to clock inaccuracies.

Overview of Generative AI Tools Used

DeepL - Translate[1], Grammarly[2]
These tools, which include partial AI support, were used to improve the quality of the written work in terms of the wordings used and the text flow in terms of grammatical improvements.

Codeium[3]
Codeium was used as an extension of the Visual Studio Code programming environment. Codeium is an AI autocompletion tool for various programming languages and was used in this sense to autocomplete single lines of programming code.

Google Gemini[4]
Google Gemini is an AI chatbot developed by Google. It was used to complete simple, straightforward parts of the analysis programs for evaluation, for example, to generate large subplots in Python faster.

Otherwise, no further support from any AI was used in this diploma thesis.

[1] https://www.deepl.com/de/translator, last accessed on 03/13/2025
[2] https://grammarly.com, last accessed on 03/13/2025
[3] https://codeium.com, last accessed on 01/08/2025
[4] https://gemini.google.com, last accessed on 02/14/2025

List of Figures

List of Tables

Bibliography

[20225a] CUDA Toolkit - Free Tools and Training | NVIDIA Developer, 2025.

[20225b] VSC: Vienna Scientific Cluster, 2025.

[AQAAC17] Sarah A. Al-Qaseemi, Hajer A. Almulhim, Maria F. Almulhim, and Saqib Rasool Chaudhry. IoT architecture challenges and issues: Lack of standardization. *FTC 2016 - Proceedings of Future Technologies Conference*, pages 731–738, 1 2017.

[ASZ17] Mark V. Albert, Ilona Shparii, and Xiaolu Zhao. The Applicability of Inertial Motion Sensors for Locomotion and Posture. *Locomotion and Posture in Older Adults: The Role of Aging and Movement Disorders*, pages 417–426, 2 2017.

[BAL09] David Bannach, Oliver Amft, and Paul Lukowicz. Automatic event-based synchronization of multimodal data streams from wearable and ambient sensors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5741 LNCS:135–148, 2009.

[BBS14] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3), 1 2014.

[BDE13] David Broman, Patricia Derler, and John C Eidson. Temporal Issues in Cyber-Physical Systems. *Journal of the Indian Institute of Science*, 93(3):389–402, 7 2013.

[Bel22] Jason Bell. What Is Machine Learning? *Machine Learning and the City*, pages 207–216, 5 2022.

[BHMVL21] Marius Bock, Alexander Hoelzemann, Michael Moeller, and Kristof Van Laerhoven. Tutorial on Deep Learning for Human Activity Recognition. 10 2021.

[BHV13] Gerald Bieber, Marian Haescher, and Matthias Vahl. Sensor Requirements for Activity Recognition on Smart Watches. *ACM International Conference Proceeding Series*, 2013.

[BI04] Ling Bao and Stephen S. Intille. Activity Recognition from User-Annotated Acceleration Data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3001:1–17, 2004.

[BJZP20] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan. Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, 3(3):196–207, 9 2020.

[BKV+97] Carlijn V.C. Bouten, Karel T.M. Koekkoek, Maarten Verduin, Rens Kodde, and Jan D. Janssen. A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity. *IEEE Transactions on Biomedical Engineering*, 44(3):136–147, 3 1997.

[BMVL23] Marius Bock, Michael Moeller, and Kristof Van Laerhoven. Temporal Action Localization for Inertial-based Human Activity Recognition. 1, 11 2023.

[BWG12] Andreas Bulling, Jamie A. Ward, and Hans Gellersen. Multimodal recognition of reading activity in transit using body-worn sensors. *ACM Transactions on Applied Perception (TAP)*, 9(1), 3 2012.

[CKC+05] Jay Chen, Karric Kwong, Dennis Chang, Jerry Luk, and Ruzena Bajcsy. Wearable sensors for reliable fall detection. *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, 7 VOLS:3551–3554, 2005.

[DSL19] Steven Díaz, Jeannie B. Stephenson, and Miguel A. Labrador. Use of Wearable Sensor Technology in Gait, Balance, and Range of Motion Analysis. *Applied Sciences 2020, Vol. 10, Page 234*, 10(1):234, 12 2019.

[DSY24] Guodong Ding, Fadime Sener, and Angela Yao. Temporal Action Segmentation: An Analysis of Modern Techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2):1011–1030, 2 2024.

[DX17] Li Ding and Chenliang Xu. TricorNet: A Hybrid Temporal Convolutional and Recurrent Network for Video Action Segmentation. 5 2017.

[ENM15] Issam El Naqa and Martin J. Murphy. What Is Machine Learning? *Machine Learning in Radiation Oncology*, pages 3–11, 2015.

[FG19] Yazan Abu Farha and Juergen Gall. MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentation. 3 2019.

[FMMN21] Anna Ferrari, Daniela Micucci, Marco Mobilio, and Paolo Napoletano. Trends in human activity recognition using smartphones. *Journal of Reliable Intelligent Environments*, 7(3):189–213, 9 2021.

[FRR11] Alireza Fathi, Xiaofeng Ren, and James M. Rehg. Learning to recognize objects in egocentric activities. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3281–3288, 2011.

[GLD$^+$24] Thomas J. Gilbert, Zexiao Lin, Sally Day, Antonia F. de C. Hamilton, and Jamie A. Ward. A magnetometer-based method for in-situ syncing of wearable inertial measurement units. *Frontiers in Computer Science*, 6:1385392, 4 2024.

[Hal20] Yann Hallouard. Time Series: The problem with resampling | by Yann Hallouard | TotalEnergies Digital Factory | Medium, 2020.

[Har20] Mark Harris. What Is An Oscillator? Everything You Need to Know | Blog | Altium, 2020.

[HEKA21] Yuman He, Kim Eun-Kyeong, and Hoda Allahbakhshi. Comparison of Feature Engineering Methods and Classifiers for Recognizing Physical Activity Types in Older Adults Using Real-Life IMU and GPS Data. 2021.

[Jos22] V. Roshan Joseph. Optimal ratio for data splitting. *Statistical Analysis and Data Mining*, 15(4):531–538, 8 2022.

[JWL$^+$24] Geonwoo Ji, Jiyoung Woo, Geon Lee, Constantino Msigwa, Denis Bernard, and Jaeseok Yun. AIoT-Based Smart Healthcare in Everyday Lives: Data Collection and Standardization From Smartphones and Smartwatches. *IEEE Internet of Things Journal*, 11(16):27597–27619, 2024.

[KAS14] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 780–787, 9 2014.

[KGH20] Seong Hoon Kim, Zong Woo Geem, and Gi Tae Han. Hyperparameter optimization method based on harmony search algorithm to improve performance of 1D CNN human respiration pattern recognition system. *Sensors (Switzerland)*, 20(13):1–20, 7 2020.

[KGS15] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*, 9 2015.

[KHA$^+$22] Hamza Khan, Sanjay Haresh, Awais Ahmed, Shakeeb Siddiqui, Andrey Konin, M. Zeeshan Zia, and Quoc Huy Tran. Timestamp-Supervised Action Segmentation with Graph Convolutional Networks. *IEEE International Conference on Intelligent Robots and Systems*, 2022-October:10619–10626, 2022.

[Kim17] Phil Kim. Convolutional Neural Network. *MATLAB Deep Learning*, pages 121–147, 2017.

[KK66] James F. Kaiser and Franklin F. Kuo. System analysis by digital computer. *New York: Wiley*, 1966.

[KM21] Ryan Kearns and Barr Moses. Data Observability in Practice: Data Monitoring at Scale with SQL and Machine Learning, 2021.

[KNN23] Amit Karmakar, Masafumi Nishida, and Masafumi Nishimura. Eating and Drinking Behavior Recognition Using Multimodal Fusion. *GCCE 2023 - 2023 IEEE 12th Global Conference on Consumer Electronics*, pages 210–213, 2023.

[KSB14] Svebor Karaman, Lorenzo Seidenari, and Alberto Del Bimbo. Fast saliency based pooling of Fisher encoded dense trajectories. 2014.

[Lau07] Hong-yin Ajax Lau. Classification of walking conditions of persons after stroke using accelerometers and gyroscopes. 2007.

[LFV$^+$16] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal Convolutional Networks for Action Segmentation and Detection. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:1003–1012, 11 2016.

[LPB$^+$10] P. Lukowicz, G. Pirkl, D. Bannach, F. Wagner, A. Calatroni, K. Foerster, T. Holleczek, M. Rossi, D. Roggen, G. Troester, J. Doppler, C. Holzmann, A. Riener, A. Ferscha, and R. Chavarriaga. Recording a Complex, Multi Modal Activity Data Set for Context Recognition | VDE Conference Publication | IEEE Xplore. In *23th International Conference on Architecture of Computing Systems 2010*, pages 1–6, Hannover, Germany, 2010.

[LRVH16] Colin Lea, Austin Reiter, René Vidal, and Gregory D. Hager. Segmental Spatiotemporal CNNs for Fine-grained Action Segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9907 LNCS:36–52, 2 2016.

[LT18] Peng Lei and Sinisa Todorovic. Temporal Deformable Residual Networks for Action Segmentation in Videos. 2018.

[Lyo11] Richard G Lyons. Understanding Digital Signal Processing Third Edition. 2011.

[Ma18] Jianhua Ma. Wear-I: A multi-wearable organic system for smarter individual services. *Proceedings - 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations*, pages 1994–1996, 12 2018.

[May22] Scott Mayo. Practical Understanding of Key Accelerometer Specifications. 2022.

[MH15] Jianhua Ma and Runhe Huang. Wear-I: A new paradigm in wearable computing. *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015*, pages 1063–1068, 12 2015.

[Nat23] National Instruments. Rational Resampling (Digital Filter Design Toolkit) - NI, 2023.

[NTAgA18] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-garadi, and Uzoma Rita Alo. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261, 9 2018.

[Num25] NumPy. numpy.lib.format — NumPy v2.3.dev0 Manual, 2025.

[ON15] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks. *International Journal for Research in Applied Science and Engineering Technology*, 10(12):943–947, 11 2015.

[OS10] Alan V.. Oppenheim and Ronald W.. Schafer. *Discrete-time signal processing*. Pearson, third edition, 2010.

[OSK+21] Aleksandr Ometov, Viktoriia Shubina, Lucie Klus, Justyna Skibińska, Salwa Saafi, Pavel Pascacio, Laura Flueratoru, Darwin Quezada Gaibor, Nadezhda Chukhno, Olga Chukhno, Asad Ali, Asma Channa, Ekaterina Svertoka, Waleed Bin Qaim, Raúl Casanova-Marqués, Sylvia Holcer, Joaquín Torres-Sospedra, Sven Casteleyn, Giuseppe Ruggeri, Giuseppe Araniti, Radim Burget, Jiri Hosek, and Elena Simona Lohan. A Survey on Wearable Technology: History, State-of-the-Art and Current Challenges. *Computer Networks*, 193:108074, 7 2021.

[OVS14] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. The LEAR submission at Thumos 2014. 2014.

[PKI+20] T. Poongodi, Rajalakshmi Krishnamurthi, R. Indrakumari, P. Suresh, and Balamurugan Balusamy. Wearable devices and IoT. *Intelligent Systems Reference Library*, 165:245–273, 2020.

[RAAS12] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1194–1201, 2012.

[RCND+10] Daniel Roggen, Alberto Calatroni, Long-Van Nguyen-Dinh, Ricardo Chavarriaga, and Hesam Sagha. OPPORTUNITY Activity Recognition - UCI Machine Learning Repository, 2010.

[RCR+10] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Hesam Sagha, Hamidreza Bayati, Marco Creatura, and Jose Del R. Millàn. Collecting complex activity datasets in highly rich networked sensor environments. *INSS 2010 - 7th International Conference on Networked Sensing Systems*, pages 233–240, 2010.

[RKG17] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly Supervised Action Learning with RNN based Fine-to-coarse Modeling. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:1273–1282, 3 2017.

[Sar24] Matt Sarrel. CPU vs GPU: What's best for Machine Learning? | Aerospike, 2024.

[Sci25] SciPy. resample_poly — SciPy v1.14.1 Manual, 2025.

[SDBV+23] Meng Shang, Camilla De Bleecker, Jos Vanrenterghem, Roel De Ridder, Sabine Verschueren, Carolina Varon, Walter De Raedt, and Bart Vanrumste. A Multi-Stage Temporal Convolutional Network for Volleyball Jumps Classification Using a Waist-Mounted IMU. 10 2023.

[SM13] Sebastian Stein and Stephen J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. *UbiComp 2013 - Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 729–738, 2013.

[SRO+08] Thomas Stiefmeier, Daniel Roggen, Georg Ogris, Paul Lukowicz, and Gerhard Tröster. Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing*, 7(2):42–50, 4 2008.

[SW11] Claude Sammut and Webb Geoffrey I. Encyclopedia of Machine Learning. *Encyclopedia of Machine Learning*, 1, 2011.

[SWBT20] Tobias Steinmetzer, Sandro Wilberg, Ingrid Bönninger, and Carlos M. Travieso. Analyzing gait symmetry with automatically synchronized wearable sensors in daily life. *Microprocessors and Microsystems*, 77:103118, 9 2020.

[SYN+23] Yuto Shimizu, Yoshitaka Yamamoto, Masafumi Nishimura, Yuki Shiono, Reiki Shirasawa, Takayuki Nakano, and Takahiro Aoki. Understanding Human Work Behavior by Action Segmentation Model with Feature Selection. *Proceedings of the 2023 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, CSDE 2023*, 2023.

[TAA19] Francisco Tirado-Andrés and Alvaro Araujo. Performance of clock sources and their influence on time synchronization in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 15(9), 9 2019.

[The25] The HDF Group. The HDF5 Library & File Format, 2025.

[vAKLS23] Beatrice van Amsterdam, Abdolrahim Kadkhodamohammadi, Imanol Luengo, and Danail Stoyanov. ASPnet: Action Segmentation With Shared-Private Representation of Multiple Data Sources, 2023.

[Vig92] John R. Vig. Introduction to quartz frequency standards. *STIN*, 1992.

[WKDR+22] Chunzhuo Wang, T. Sunil Kumar, Walter De Raedt, Guido Camps, Hans Hallez, and Bart Vanrumste. Drinking Gesture Detection Using Wrist-Worn IMU Sensors with Multi-Stage Temporal Convolutional Network in Free-Living Environments. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2022-July:1778–1782, 2022.

[WSL+19] Chaofan Wang, Zhanna Sarsenbayeva, Chu Luo, Jorge Goncalves, and Vassilis Kostakos. Improving wearable sensor data quality using context markers. *UbiComp/ISWC 2019- - Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pages 598–601, 9 2019.

[XGM17] Tongtong Xu, Ao Guo, and Jianhua Ma. Analysis of temporal features in data streams from multiple wearable devices. *2017 3rd IEEE International Conference on Cybernetics, CYBCONF 2017 - Proceedings*, 7 2017.

[XGMW17] Tongtong Xu, Ao Guo, Jianhua Ma, and Kevin I.Kai Wang. Feature-Based Temporal Statistical Modeling of Data Streams from Multiple Wearable Devices. *Proceedings - 2017 IEEE 15th International Conference on Dependable, Autonomic and Secure Computing*, 2018-January:119–126, 7 2017.

[XZSS12] Wenyao Xu, Mi Zhang, Alexander A. Sawchuk, and Majid Sarrafzadeh. Co-recognition of human activity and sensor location via compressed sensing in wearable body sensor networks. *Proceedings - BSN 2012: 9th International Workshop on Wearable and Implantable Body Sensor Networks*, pages 124–129, 2012.

[YFG$^+$24a] Po Kai Yang, Benjamin Filtjens, Pieter Ginis, Maaike Goris, Alice Nieuwboer, Moran Gilat, Peter Slaets, and Bart Vanrumste. Automatic Detection and Assessment of Freezing of Gait Manifestations. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 32:2699–2708, 2024.

[YFG$^+$24b] Po Kai Yang, Benjamin Filtjens, Pieter Ginis, Maaike Goris, Alice Nieuwboer, Moran Gilat, Peter Slaets, and Bart Vanrumste. Freezing of gait assessment with inertial measurement units and deep learning: effect of tasks, medication states, and stops. *Journal of NeuroEngineering and Rehabilitation*, 21(1):1–22, 12 2024.

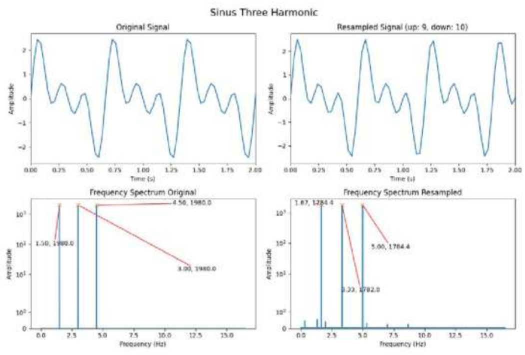
[YSMS14] Won Jae Yi, Oishee Sarkar, Sivisa Mathavan, and Jafar Saniie. Wearable sensor data fusion for remote health assessment and fall detection. *IEEE International Conference on Electro Information Technology*, pages 303–307, 2014.

[ZVVC22] Yiyuan Zhang, Bart Vanrumste, Sabine Verschueren, and Wei Chen. *Daily Living Activities Recognition and Segmentation in Older Adults with Sensor Technology and Machine Learning Algorithms.* PhD thesis, KULeuven, 5 2022.

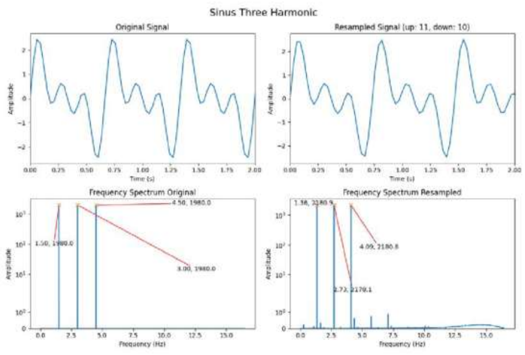
APPENDIX A

Appendices

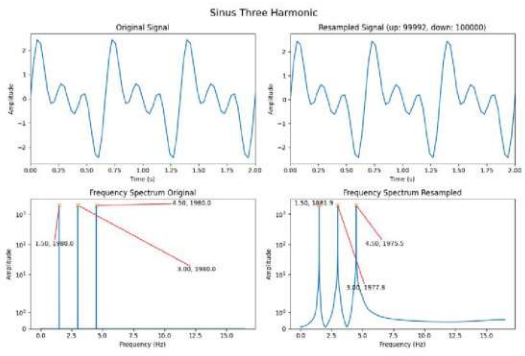
A.1 Additional Functionality Verification of the Rational Resampling Function



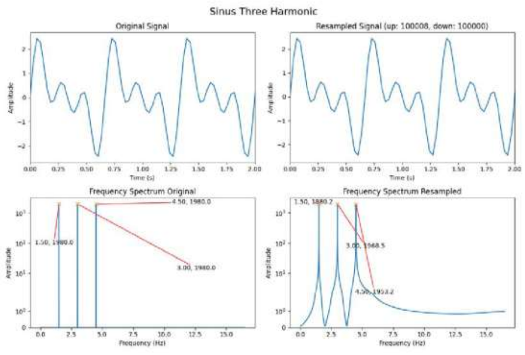
(a) Three harmonic sinuses with upsample factor = 9 and downsample factor = 10



(b) Three harmonic sinuses with upsample factor = 11 and downsample factor = 10



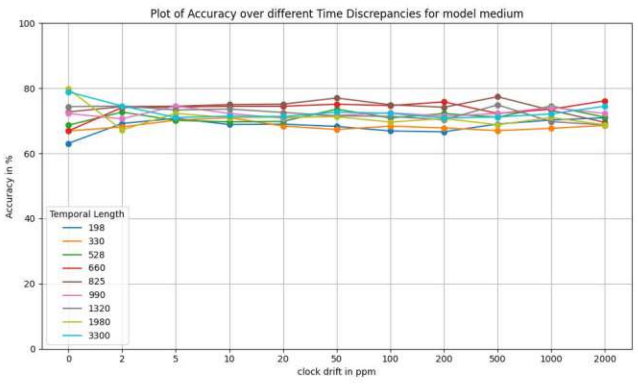
(c) Three harmonic sinuses with upsample factor = 99,992 and downsample factor = 100,000



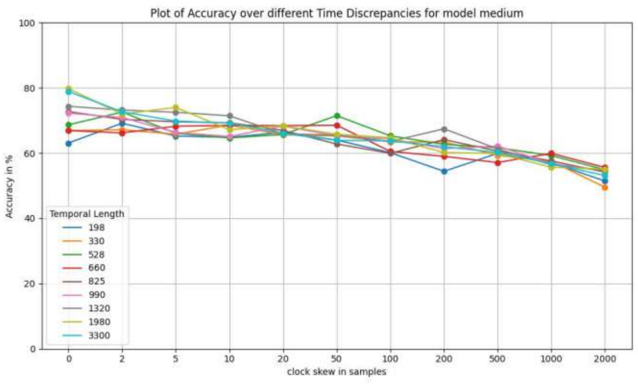
(d) Three harmonic sinuses with upsample factor = 100,008 and downsample factor = 100,000

Figure A.1: Time domain and frequency domain visualization of clock drift using rational resampling on three harmonic sinuses with different upsample and downsample factors. In addition to the three dominant frequencies of the actual signal, the top two figures also show additional small frequency components. These may be caused by slight noise during the resampling process. The successful resampling and, thus, change in the frequency of the peaks is visible. The lower two figures show wider main lobes caused by the use of the Kaiser window. But no additional frequency components appear.

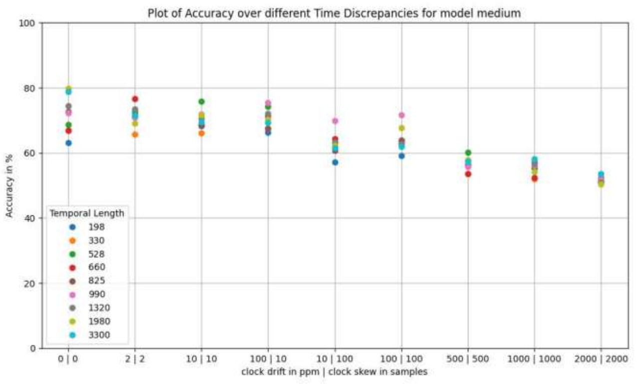
A.2 Additional Results



(a) Accuracy over clock drift in ppm for the different temporal lengths of the models with the complexity medium.

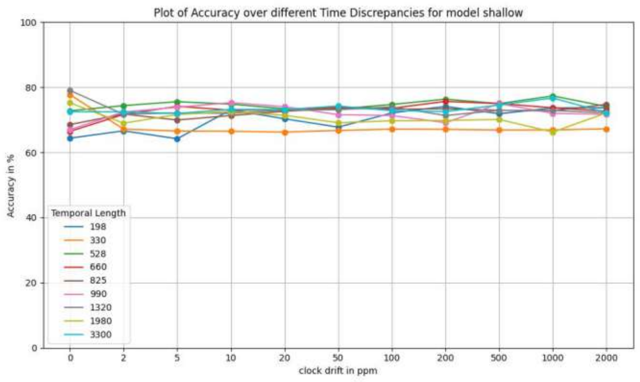


(b) Accuracy over clock skew in samples for the different temporal lengths of the models with the complexity medium.

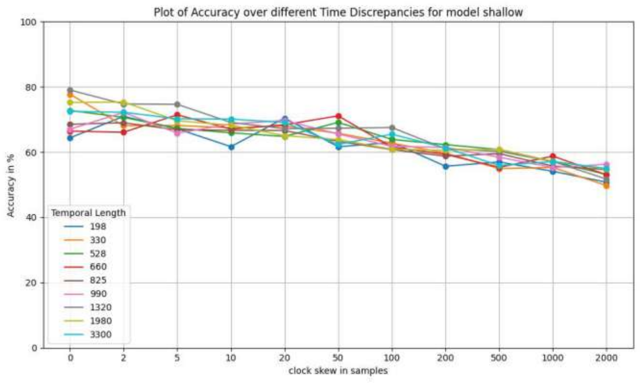


(c) Accuracy over combinations of clock drift in ppm and clock skew in samples for the different temporal lengths of the models with the complexity medium.

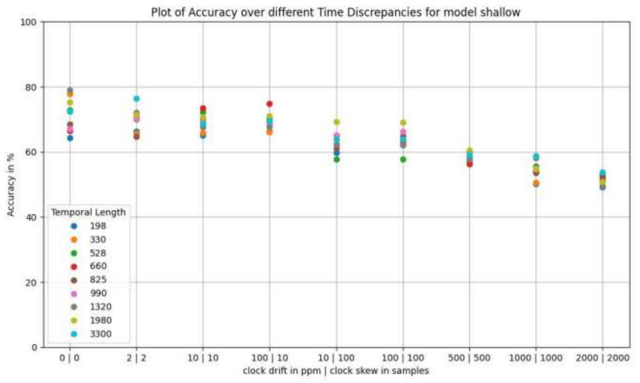
Figure A.2: Accuracy over different combinations of clock drift in ppm and clock shift in samples for the different temporal lengths of the models with the complexity medium.



(a) Accuracy over clock drift in ppm for the different temporal lengths of the models with the complexity shallow.

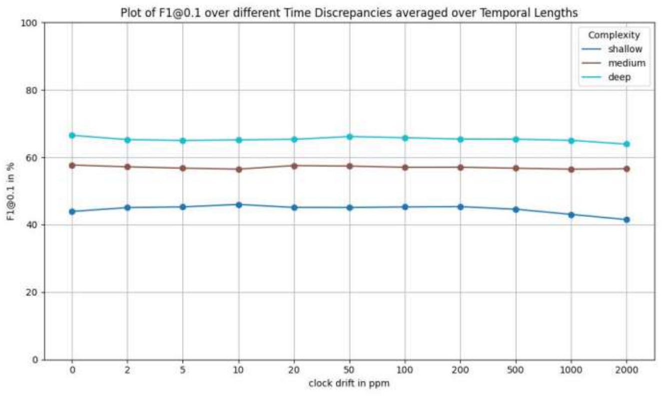


(b) Accuracy over clock skew in samples for the different temporal lengths of the models with the complexity shallow.

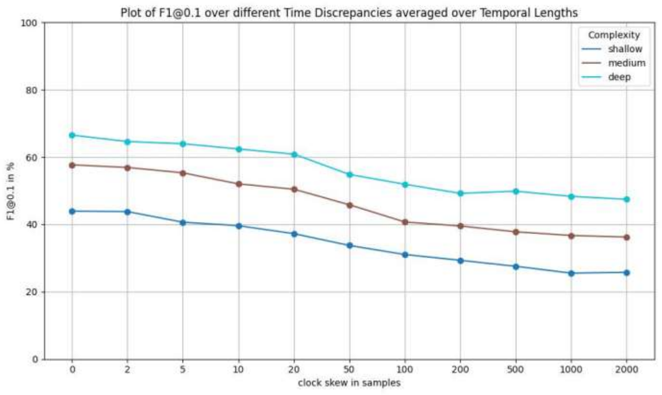


(c) Accuracy over combinations of clock drift in ppm and clock skew in samples for the different temporal lengths of the models with the complexity shallow.

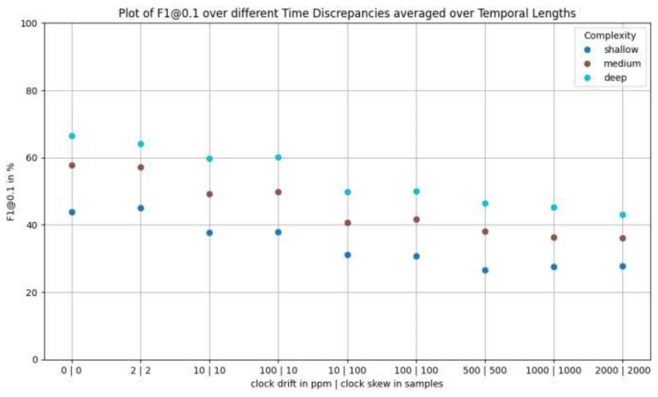
Figure A.3: Accuracy over different combinations of clock drift in ppm and clock shift in samples for the different temporal lengths of the models with the complexity shallow.



(a) F1@0.1-scores over clock drift in ppm averaged across all models of the individual complexities.

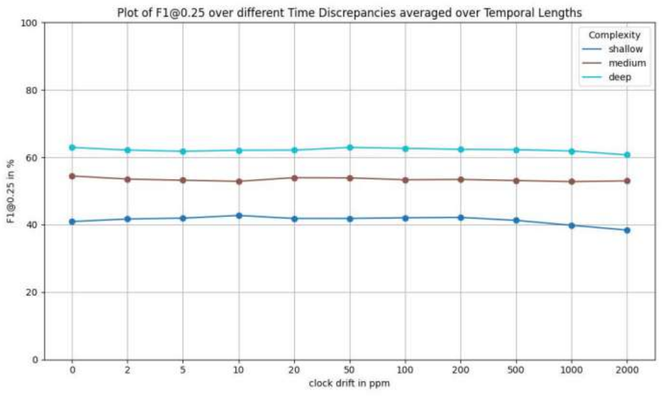


(b) F1@0.1-scores over clock skew in samples averaged across all models of the individual complexities.

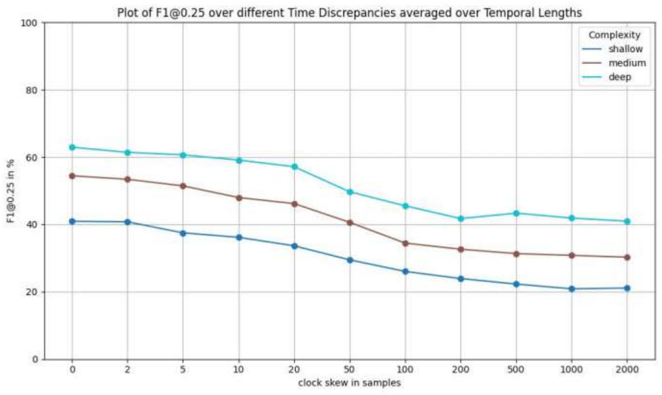


(c) F1@0.1-scores over combinations of clock drift in ppm and clock skew in samples averaged across all models of the individual complexities.

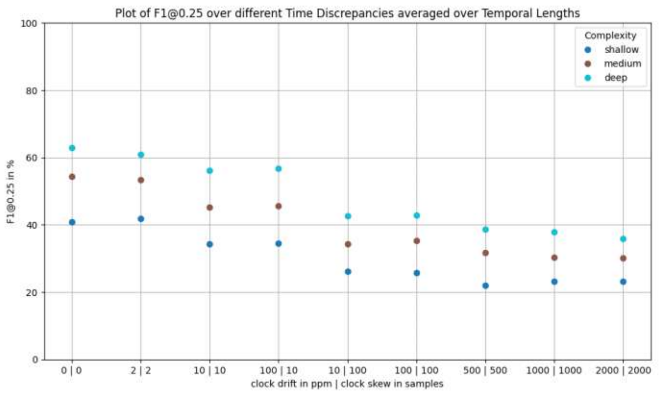
Figure A.4: F1@0.1-scores over different combinations of clock drift in ppm and clock shift in samples averaged over all models of the individual complexities.



(a) F1@0.25-scores over clock drift in ppm averaged across all models of the individual complexities.

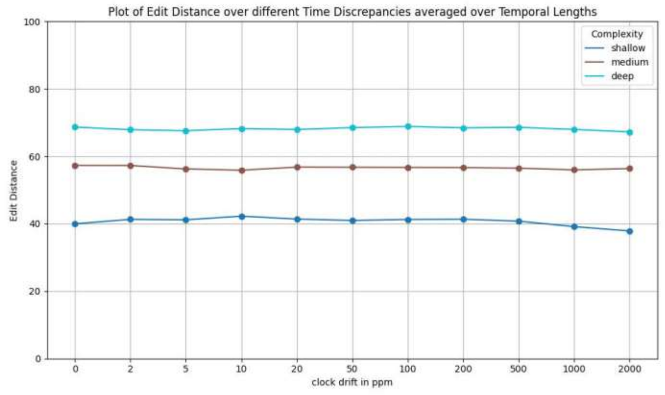


(b) F1@0.25-scores over clock skew in samples averaged across all models of the individual complexities.

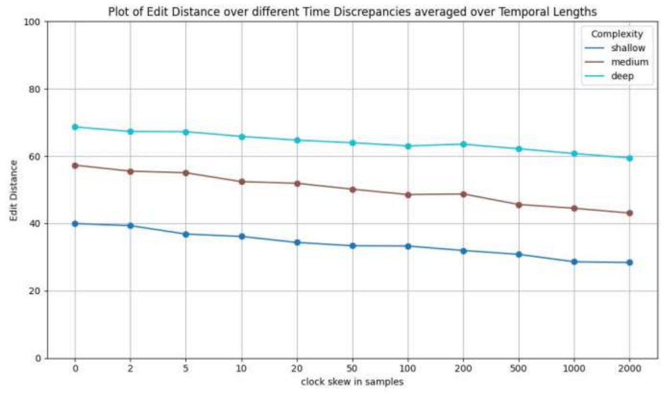


(c) F1@0.25-scores over combinations of clock drift in ppm and clock skew in samples averaged across all models of the individual complexities.

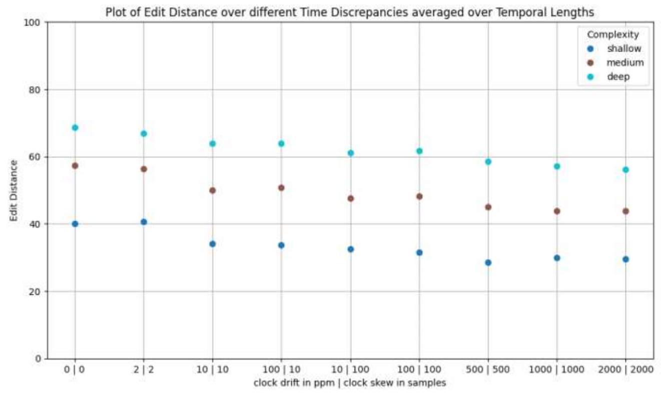
Figure A.5: F1@0.25-scores over different combinations of clock drift in ppm and clock shift in samples averaged over all models of the individual complexities.



(a) Edit distance over clock drift in ppm averaged across all models of the individual complexities.

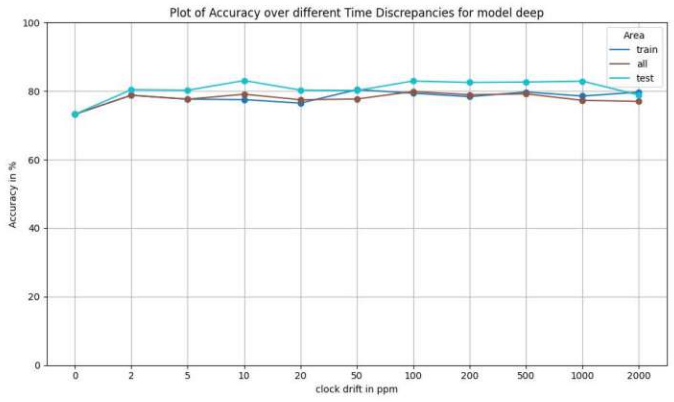


(b) Edit distance over clock skew in samples averaged across all models of the individual complexities.

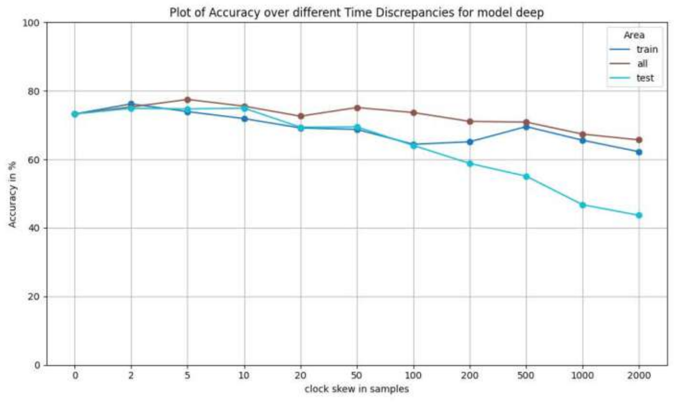


(c) Edit distance over combinations of clock drift in ppm and clock skew in samples averaged across all models of the individual complexities.

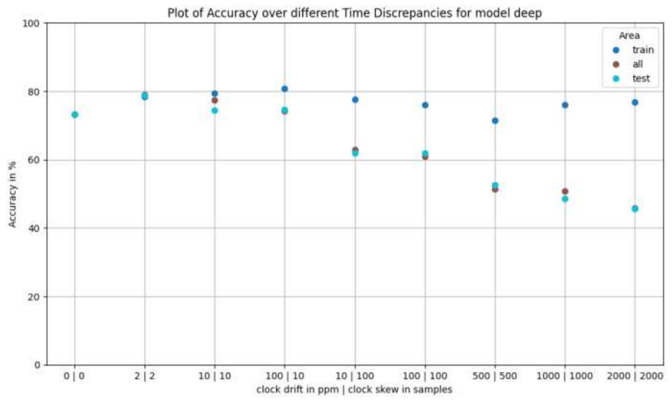
Figure A.6: Edit distance over different combinations of clock drift in ppm and clock shift in samples averaged over all models of the individual complexities.



(a) Accuracy over clock drift in ppm divided into the different areas for models with the complexity deep.

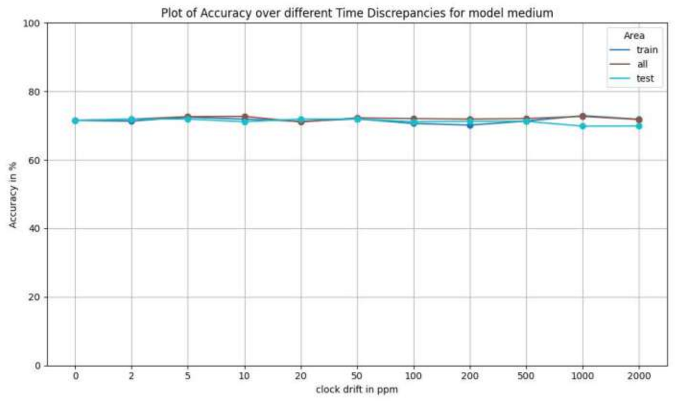


(b) Accuracy over clock skew in samples divided into the different areas for models with the complexity deep.

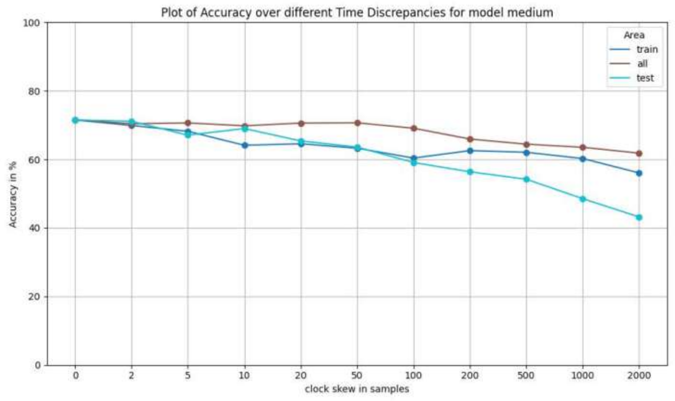


(c) Accuracy over combinations of clock drift in ppm and clock skew in samples divided into the different areas for models with the complexity deep.

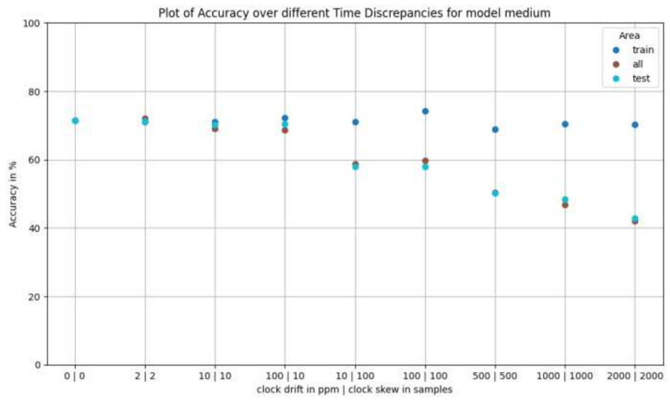
Figure A.7: Accuracy over different combinations of clock drift in ppm and clock shift in samples divided into the different areas for models with the complexity deep.



(a) Accuracy over clock drift in ppm divided into the different areas for models with the complexity medium.

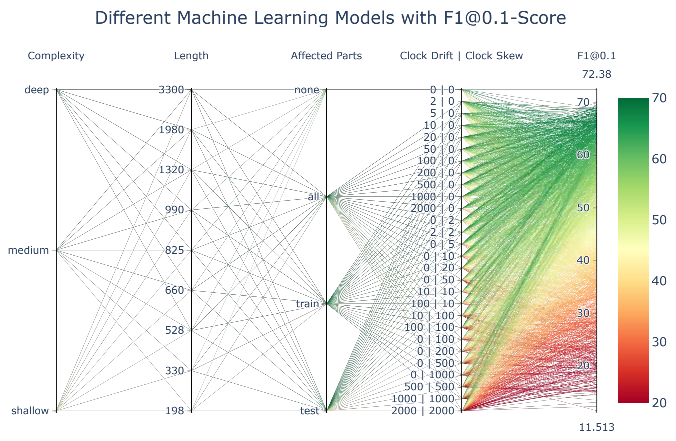


(b) Accuracy over clock skew in samples divided into the different areas for models with the complexity medium.

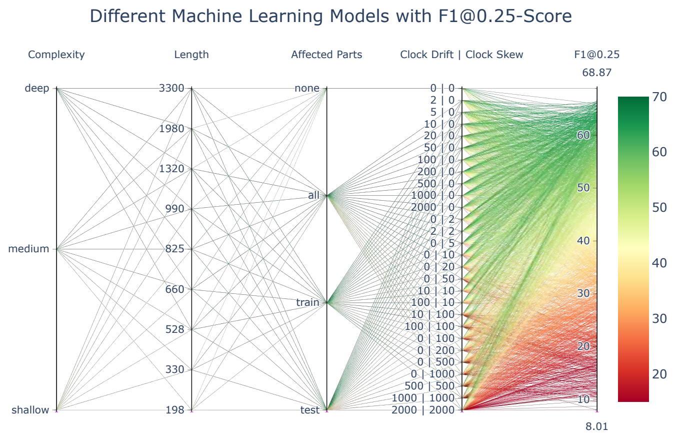


(c) Accuracy over combinations of clock drift in ppm and clock skew in samples divided into the different areas for models with the complexity medium.

Figure A.8: Accuracy over different combinations of clock drift in ppm and clock shift in samples divided into the different areas for models with the complexity medium.



(a) A parallel coordinate plot is shown here. The first axis displays the different complexities of the machine learning models and the second axis shows the corresponding temporal lengths of the inputs for the respective models. The third axis illustrates which part of the machine learning was affected by temporal discrepancies. The fourth axis shows the various time discrepancies with clock drift and clock skew. The last axis represents the F1@0.1-score of the models. It can be seen that the F1@0.1-score decreases for larger time discrepancies across all models.



(b) A parallel coordinate plot is shown here. The first axis displays the different complexities of the machine learning models and the second axis shows the corresponding temporal lengths of the inputs for the respective models. The third axis illustrates which part of the machine learning was affected by temporal discrepancies. The fourth axis shows the various time discrepancies with clock drift and clock skew. The last axis represents the F1@0.25-score of the models. It can be seen that the F1@0.25-score decreases for larger time discrepancies across all models.

Figure A.9: Different parallel coordinate plot for all calculated models.