Applications of Statistical Theory to Sensor Data Analysis

Ciszewski, M.G.

DOI
10.4233/uuid:628be99a-d5f1-4a60-a6dc-261e3b02cef2

Publication date
2024

Document Version
Final published version

Citation (APA)
Ciszewski, M. G. (2024). *Applications of Statistical Theory to Sensor Data Analysis*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:628be99a-d5f1-4a60-a6dc-261e3b02cef2

Applications of statistical theory to sensor data analysis

Applications of statistical theory to sensor data analysis

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates,
to be defended publicly on
Monday 14 October 2024 at 12.30 o'clock

by

Michał Grzegorz CISZEWSKI

Master of Science in Applied Mathematics,
AGH University of Krakow, Poland,
born in Zawiercie, Poland.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. G. Jongbloed,	Delft University of Technology, promotor
Dr. J. Söhl,	Delft University of Technology, copromotor
Independent members:	
Prof. dr. A.J. Cabo,	Delft University of Technology
Prof. dr. ir. K.M.B. Jansen,	Delft University of Technology
Prof. dr. ir. F.H. van der Meulen,	Vrije Universiteit Amsterdam
Prof. dr. H.E.J. Veeger,	Delft University of Technology
Dr. K. Proksch,	University of Twente
Prof. dr. H.M. Schuttelaars,	Delft University of Technology, reserve member

Keywords: data analysis, statistics, sensor data, activity recognition, modelling

Printed by: Ipskamp Printing

Cover: Pezo Kazadi

Style: TU Delft House Style, with modifications by Moritz Beller
`https://github.com/Inventitech/phd-thesis-template`

The author set this thesis in LaTeX using the Libertinus and Inconsolata fonts.

ISBN 978-94-6473-581-9

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Contents

Summary

Technological progress irreversibly changes the nature of sports. The relevance of technology in sports can be seen with relative ease to most spectators in tennis, football and many other elite sports. Some technologies have changed the sport in a way that many spectators might not be aware of. Behind any professional sport, there are countless hours of training and preparation. Athletes are pushing their own limits in achieving perfection. Coaches are trying to make sure that the training the athletes go through results in improvement of their performance, but without straining themselves too much which can lead to an injury. The technology of today helps with this training process and coaches need to be able to use it to provide good feedback to their athletes.

This thesis is written in the context of the Citius Altius Sanius (CAS) project aimed at injury prevention and performance improvement in sports. The CAS project combines the expertise of data scientists, industrial designers and biomechanical engineers together with the resources of sports associations and sports equipment designers among others. The goal of the CAS project is to initiate collaboration between various universities and departments to develop sensor technology, provide analysis based on the sensor data and provide a clear guideline of feedback to the athlete.

The primary goal of this thesis is to extract meaningful insights from sensor data through statistical modeling. Two sources of sensor data are used within the thesis: data from prototype sensor trousers worn by football players during training and data from a sensor sleeve worn by tennis players during serve practice. The research employs supervised learning algorithms within the framework of machine learning and deep learning models for capturing intricate patterns in the data as well as functional data analysis techniques such as functional principal components analysis and functional regression models applied for imputation purposes and dimension reduction.

We used neural network architecture, which mixes both convolutional and recurrent layers, consistently throughout this thesis. The main application of this network lies in recognizing football-related activities using sensor data. The neural network achieves good accuracy and is easily adaptable to other human activity recognition problems. We also considered various other models for this task, however none could match the computational speed and accuracy of the neural network. Nonetheless, given a plethora of methods that were tested and dissatisfaction with the accuracy measures used to assess the goodness-of-fit of the tested methods, a novel quality measure was introduced for activity recognition problems, to leverage the domain knowledge for the purpose of determining accuracy of an activity recognition method. In the case of our application, one of the constraints is the length of activities that are predicted. This measure accounts for the fact that activities such as jumping or passing a ball realistically have a minimum duration. Instances where a prediction model outputs an activity shorter than physically plausible incur harsh penalties.

We also propose a novel post-processing procedure tailored specifically to human activity recognition problems, ensuring that predictive models adhere to physical constraints,

such as the minimum duration of activities. This post-processing method aims to increase the accuracy of prediction models which violate these constraints and as a result, to narrow the gap in accuracy between different prediction methods.

In the context of tennis, we encountered difficulties in predicting the serve performance metrics using sensor data. While predicting the ball speed can be easily achieved, accurately predicting the velocity-accuracy index (VA index), which combines ball speed with serve accuracy, proved more complex. To assess the effectiveness of our model in distinguishing true predictions from noise, we applied a permutation test. Notably, the main contribution of this research lies in the rigorous formulation of the null hypothesis for this test, linking it to established permutation test theory.

This research contributes to the fields of sports science and data analysis by offering insights into activity recognition and performance prediction using sensor data. The methodologies developed here have potential applications across various other sports as well as activities unrelated to sports. While data provided for purposes of this research comes from wearable sensors, it is possible to also apply these models and procedures in other types of sensor data or even beyond.

Samenvatting

Technologische vooruitgang verandert de aard van sport onomkeerbaar. De relevantie van technologie in de sport is voor de meeste toeschouwers van tennis, voetbal en vele andere topsporten relatief gemakkelijk te zien. Sommige technologieën hebben de sport veranderd op een manier waarvan veel toeschouwers zich misschien niet bewust zijn. Achter elke professionele sport gaan ontelbare uren training en voorbereiding schuil. Sporters verleggen hun eigen grenzen om perfectie te benaderen. Coaches proberen ervoor te zorgen dat de training van de atleten resulteert in een verbetering van hun prestaties, maar zonder zichzelf te veel te belasten, wat kan leiden tot blessures. De technologie van tegenwoordig helpt bij dit trainingsproces en coaches moeten deze technologie kunnen gebruiken om goede feedback te geven aan hun atleten.

Deze scriptie is geschreven in het kader van het Citius Altius Sanius (CAS) project gericht op blessurepreventie en prestatieverbetering in de sport. Het CAS-project combineert de expertise van datawetenschappers, industrieel ontwerpers en biomechanisch ingenieurs met de middelen van onder andere sportbonden en ontwerpers van sportuitrusting. Het doel van het CAS-project is om een samenwerking tussen verschillende universiteiten en afdelingen op gang te brengen om sensortechnologie te ontwikkelen, analyses te maken op basis van de sensordata en duidelijke feedback aan de sporter te geven.

Het primaire doel van dit proefschrift is om zinvolle inzichten uit sensordata te halen door middel van statistische modellering. In dit proefschrift worden twee bronnen van sensorgegevens gebruikt: gegevens van een prototype van een sensorbroek die door voetbalspelers tijdens de training wordt gedragen en gegevens van een sensorhoes die door tennisspelers tijdens de serveertraining wordt gedragen. Het onderzoek maakt gebruik van supervised learning algorithmen binnen het kader van machine learning en deep learning modellen voor het vastleggen van ingewikkelde patronen in de gegevens, evenals functionele dataanalysetechnieken zoals functionele principale componentenanalyse en functionele regressiemodellen die worden toegepast voor imputatiedoeleinden en dimensiereductie.

We hebben in dit proefschrift consequent gebruik gemaakt van een neurale netwerkarchitectuur, die een mix is van convolutionele en terugkerende lagen. De belangrijkste toepassing van dit netwerk ligt in het herkennen van voetbalgerelateerde activiteiten met behulp van sensordata. Het neurale netwerk bereikt een goede nauwkeurigheid en is gemakkelijk aan te passen aan andere menselijke activiteitsherkenningsproblemen. We hebben ook verschillende andere modellen overwogen voor deze taak, maar geen daarvan kon de berekeningssnelheid en nauwkeurigheid van het neurale netwerk evenaren. Gezien de overvloed aan geteste methoden en de ontevredenheid over de nauwkeurigheidsmaten die gebruikt werden om de goodness-of-fit van de geteste methoden te beoordelen, hebben we een nieuwe kwaliteitsmaat geïntroduceerd voor activiteitenherkenningsproblemen, om gebruik te maken van de domeinkennis om de nauwkeurigheid van een activiteitsherken-

ningmethode te bepalen. In het geval van onze toepassing is een van de beperkingen de lengte van de activiteiten die worden voorspeld. Deze maat houdt rekening met het feit dat activiteiten zoals springen of het passen van een bal realistisch gezien een minimale duur hebben. Als een voorspellingsmodel een activiteit voorspelt die korter is dan fysiek plausibel, wordt dit zwaar bestraft.

We stellen ook een nieuwe post-processing procedure voor die specifiek is afgestemd op menselijke activiteitsherkenningsproblemen en die ervoor zorgt dat voorspellende modellen zich houden aan fysieke beperkingen, zoals de minimale duur van activiteiten. Deze post-processing methode heeft als doel om de nauwkeurigheid van voorspellingsmodellen die deze beperkingen schenden te verhogen en daardoor de kloof in nauwkeurigheid tussen verschillende voorspellingsmethoden te verkleinen.

In de context van tennis ondervonden we moeilijkheden bij het voorspellen van de prestatiemetingen van de opslag met behulp van sensorgegevens. Het voorspellen van de balsnelheid is eenvoudig, maar het nauwkeurig voorspellen van de velocity-accuracy index (VA index), die de balsnelheid combineert met de serveernauwkeurigheid, bleek ingewikkelder. Om te beoordelen hoe goed ons model werkt om ware voorspellingen te onderscheiden van ruis, hebben we een permutatietest toegepast. De belangrijkste bijdrage van dit onderzoek ligt in de strikte formulering van de nulhypothese voor deze test, die gekoppeld is aan de gevestigde theorie van permutatietests.

Dit onderzoek draagt bij aan de sportwetenschappen en data-analyse door inzicht te bieden in activiteitsherkenning en prestatievoorspelling met behulp van sensordata. De hier ontwikkelde methodologieën hebben potentiële toepassingen in verschillende andere sporten en activiteiten die niet gerelateerd zijn aan sport. Hoewel de gegevens voor dit onderzoek afkomstig zijn van draagbare sensoren, is het mogelijk om deze modellen en procedures ook toe te passen op andere soorten sensordata of zelfs nog meer.

Acknowledgments

I never imagined that I would successfully complete my PhD. This entire journey was challenging and tumultuous, but it became the most rewarding five years of my life. What began during the pandemic, initially a difficult time, proved to be a blessing in disguise. Those early months of isolation allowed me to lay the groundwork for the paper published three years later, now represented in Chapter 3. It served as proof that I could contribute meaningfully, and deserved to finish this PhD. Ultimately, this thesis would not have been possible without the support of many people, whom I wish to thank in this section.

Jakob, thank you for your patience with me and the time you have spent to help me. Thank you for pushing me to be better at communicating and collaborating. Thank you for instilling confidence in me and helping me to be more assertive.

Geurt, thank you for your expertise and many unconventional suggestions to lead me to the solutions of my problems. Thank you for supporting me through the years and making sure I was passionate about the direction this project went in. I would be remiss, if I did not thank you for your humor, the witty remarks and attempts at spelling Polish words.

I would like to thank all my collaborators. **Kaspar**, thank you for teaching me how to be a better supervisor. I cherish the memories of our joint supervision of students. **Bart**, **Ton**, **Annemarijn** and **Erik**, thank you for data collection, sensor design and your invaluable feedback regarding my research. **Rafael** and **Ricardo**, thank you for your contributions in creation and improvement of the activity recognition model. I would also like to thank **NWO** for funding this PhD project and making it all possible.

Thank you to all my friends and colleagues. **Vicente**, **Ivet** and **Frederick**, thank you for your friendship, camaraderie and our board game nights. **Ardjen**, **Jan-Tino**, **Serena**, **Francesco**, **Marc**, **Andrea**, thank you for being part of my PhD journey. **Bart**, thank you for your suggestion to the paper on which Chapter 3 was eventually based on.

Dziękuję również mojej rodzinie za pomoc, motywację, cierpliwość, wyrozumiałość i wsparcie w trakcie tego doktoratu: moim **rodzicom**, siostrze **Julii**, dziadkowi **Jurkowi**, ś.p. dziadkowi **Piotrkowi**, babci **Ani** i babci **Wenii**.

Also thank you to my family in-law, who supported me throughout this process: **parents in-law**, **Kanyabu**, **Sakwasa** and **Mibenge**.

Last, but most certainly not least, **Pezo**, thank you for your love, support and patience. Your belief in me has been the cornerstone of this journey, and this PhD would simply have been impossible without you by my side.

Inspired by my supervisors, supported by the love of my wife, and encouraged by the comfort of friends and family, I have reached the final destination of this incredible journey. I can finally say I am truly proud of what I have achieved.

Michał
Rotterdam, August 2024

1

Introduction

Since the advent of professional sports, new technologies have been enhancing training methods, influencing the equipment design, and providing data-driven insights for strategic decision-making for coaches. In recent years we have seen many examples of technology changing crucial parts of certain sports. One such example is the Hawkeye system introduced in early 2000s to professional tennis matches. Initally, it served as a supporting mechanism to the work of the line umpires. Now, Grand Slams do not even employ line umpires and in 2025 the Association of Tennis Professionals (ATP) announced that all ATP tournaments will use electronic line umpires. Another example is the goal-line technology in football. Throughout the history of football, many human-made mistakes led to the unfair allowing or disallowing of goals, most famously in the recent history during the 2010 World Cup, in which England was not given a goal despite the ball crossing the goal line.

These examples show that technology has permeated sports and redefined the way in which rules of the game are applied. More than that, technology can refine the performance of athletes through the use of camera system or wearable sensors and providing direct feedback. Tools and insights offered to the elite athletes are immense and can simulate coaching, analyze the performance and potentially reduce injury risks.

1.1 Citius Altius Sanius

This thesis represents one of the outcomes of the Citius Altius Sanius (CAS) program. The project's name, a play on the Olympic motto, translates from Latin to mean "quicker, higher, healthier". Initiated in 2017 and led by TU Delft and Vrije Universiteit Amsterdam, CAS is a research program with a primary focus on injury prevention and performance improvement in sports.

The primary motivation for the project stems from the numerous diseases, including coronary heart disease, obesity and type II diabetes, primarily caused by prevailing sedentary lifestyle and unhealthy diet. In many ways, *regular* physical activity acts as a counterbalance to this unhealthy lifestyle. On the other hand in elite sports, a rigorous training regimen coupled with limited rest can lead to an increased risk of muscle injury.

The CAS program is dedicated to encouraging people to engage in physical activities more frequently. To achieve this goal, the program leverages data science and sensor

1

technology, aiming to provide feedback to individuals during exercise regarding their performance and exertion.

The program comprises three fundamental projects and six applied projects, see fig. 1.1. The fundamental projects focus on sensor technology (P1), data science (P2) and feedback (P3). This structured approach follows a pipeline: sensors are developed in subproject P1, then data produced are analyzed and distilled in subproject P2. Finally results from these analyses develop into motivational feedback for the athlete in subproject P3. Six applications were selected to cover most sport injuries in the Netherlands: fitness and strength training (P4), running (P5), football and field hockey (P6), tennis and baseball (P7), heat stroke (P8) and cycling (P9), see fig. 1.1.

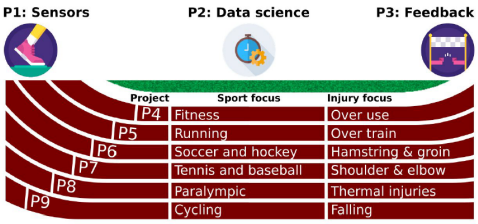


Figure 1.1: The overview of different subprojects within the CAS program.

Figure 1.2: Sensor trousers prototype with fully embedded sensors developed within subproject P1 of the CAS program. Taken from [1].

Especially relevant to this thesis are subprojects P2, P6 and P7. This dissertation provides models for the sensor data as well as its analysis, which contributes to subproject P2. Subprojects P6 and P7 specifically focus on football and tennis, both supported by national associations (KNVB for football and KNLTB for tennis). Data gathered in P6 and P7 are the essential part of the application of our models. The primary goal of the CAS program is injury prevention and performance enhancement and in the process of addressing those topics it is important to automatically recognize activities performed by an athlete. Quite often, sensor data are partly missing and data imputation techniques can potentially be employed to be able to recognize activities in spite of this. This thesis covers the topics of activity recognition and data imputation.

1.2 Data

Wearable sensors have evolved rapidly in recent years and have found widespread applications, also in the realm of sports. In football, players routinely wear vests equipped with GPS trackers, offering detailed insights into their location, distance traveled, speed, power, intensity and heart rate. One of the achievements of the CAS program is the development of the trousers with embedded sensors (fig. 1.2). The unobtrusive nature of these sensors, combined with the low production cost, makes them a highly appealing alternative to a camera system, which is much more expensive and unavailable to most amateur athletes.

The sensor trousers were specifically developed for use in football for subproject P6. The football sensor data used in this thesis originate from experiments utilizing these trousers. For gathering tennis data in subproject P7 sensors were attached directly to the body. In both cases, Inertial Measurement Units (IMUs) were employed to measure characteristics of specific body parts. Each IMU contains a tri-axial accelerometer, gyroscope and magnetometer. The accelerometer measures acceleration, the gyroscope measures angular velocity and the magnetometer measures the magnetic field. Magnetometer data were not used in this thesis. These were only used during the data formatting process to correctly orient the data.

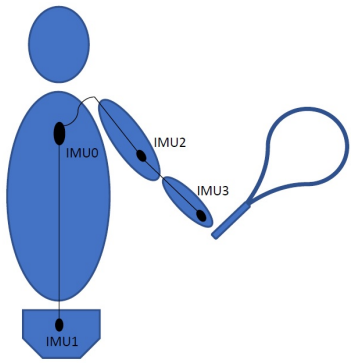


Figure 1.3: Segment model of right-handed player and racquet (back view, frontal plane) demonstrating sensor placement. Taken from [2].

The placement of sensors naturally depends on the activities that are performed. In football, the sensors were strategically placed on five body parts: pelvis, thighs and shanks, each crucial to football-specific activities such as kicking, running and jumping. The sensor trousers are equipped with sensors corresponding to each of these body parts. For tennis-specific activities, sensors were positioned on four key body parts: pelvis, trunk, upper arm and lower arm of the dominant side (fig. 1.3). The tennis data used in this thesis specifically pertains to tennis serves; sensors on other body parts, such as the legs, were not attached.

1.3 Modelling and data analysis

In the pursuit of extracting meaningful insights from the wealth of sensor data, the focus shifts to modelling. This section dives into the intricate process of using data to predict and recognize various physical activities or performance metrics related to physical activity. The main goal of this research project is to understand the patterns within the sensor data, allowing us to address issues of activity recognition, missing data and prediction. The aim is to apply existing techniques to the sensor data whenever available and to contribute to the broader knowledge by introducing our own methods when necessary. We will now explore the relevant topics regarding modelling in this thesis.

Machine learning

Machine learning is a broad field within mathematics and computer science, focused on deriving patterns from training data and generalizing to unseen data. *Generalization*, a fundamental concept in machine learning, emphasizes a model's capacity to use the training data to make accurate predictions on new, unseen data. As the name suggests, the idea behind machine learning is to imitate the process of learning. The field is typically categorized by the type of input. Supervised learning involves machine learning algorithms that learn based on patterns present in the training data; patterns supplied into the training data in the form of labels. On the other hand, unsupervised learning refers to algorithms that

find patterns in unlabeled data. This approach can help separate data into subsets, revealing similarities between the data in each subset. This thesis will exclusively employ supervised learning algorithms thanks to the data available to us and thanks to the research goals. Classic machine learning techniques include linear regression, decision trees, K-nearest neighbor algorithm and support vector machines.

Deep learning models

Deep learning models, characterized by their multi-layered neural architectures, have demonstrated an ability to capture intricate patterns and representations such as images or videos. The shift from traditional machine learning techniques towards deep learning allows us to make use of complex data structures, while effectively handling large volumes of data. They also capture non-linear relationships in data, which can be much harder to achieve for traditional machine learning techniques usually requiring human input. However, deep learning models are inherently harder to interpret and require significant computational resources to train. Different neural network architectures are considered in this thesis and are explained in more detail in later chapters.

Functional data analysis

Representation of data is an important topic to consider when modelling. Sensor data consists of multivariate time series. However, sometimes it is better to represent time series data as a set of smooth functions. Functional data introduces the concept of treating functions as data points, where each function becomes an element of the dataset. This representation better handles irregularly sampled data and reduces the dimensions of the dataset. It also allows to use advanced statistical techniques tailored for functional data analysis, which leverage the structure of the data to extract patterns. We use functional data analysis for the purposes of data imputation, exploring the idea of what would happen if we remove one or two IMUs from the analysis and replace them with imputed data learned from all the other sensors and how that could impact the analysis.

Quality measures

The last step of modelling is finding an appropriate quality measure. Optimally, a quality measure answers a question of how well does a particular model match the data? When choosing the quality measure, we need to consider what we find to be the most important feature of quality and then translate it into a mathematical concept. Consider a model that assigns one of two possible labels. Given such a binary outcome, many different quality measures can be used, such as the misclassification rate, the false positive rate and the F score. There is no definitive right choice; it all depends on our application and how one wants to use it. In this thesis, we introduce a novel quality measure for activity recognition. To define the measure, multiple parameters can be selected based on the domain knowledge, e.g. the minimum length of activities we recognize, which allows greater level of adaptability to a specific application.

1.4 Outline

This thesis centers around two separate topics. The first is the activity recognition of football-specific activities. Chapter 2 describes neural network architectures, which are designed for prediction of football activities. Multiple architectures are considered and compared with regards to their performance on the football sensor data. The models proposed in this chapter can be applied in other human activity recognition problems as well. Chapter 3 addresses the problem of football activity recognition and proposes a new post-processing scheme exploiting domain knowledge. This post-processing method allows to eliminate unrealistic activity predictions made by the model and as an effect, increases the accuracy of the model. It also presents a novel quality measure for activity recognition problems. This measure accounts for the fact that activities such as jumping or passing a ball realistically have a minimum duration. Chapter 4 revisits the framework of Chapter 2 and explores the impact of removing sensors from the dataset, replacing the associated data with imputed features predicted using all the other sensors. This is a relevant problem as some sensors can be removed for the next redesign of the sensor trousers and we would like to use the already developed model on a smaller dataset.

The second topic considered in this thesis is the prediction of ball speed and the so-called VA index of tennis serves, which combines speed with the accuracy of the shot. Chapter 5 examines the capabilities of the chosen model to capture a relationship between explanatory variables X and the response variable Y. A permutation test is considered with a novel formulation of the null hypothesis. One of the advantages of this test is that it does not rely on performance comparison between different models. Only one model is considered and the test either rejects the null hypothesis, which means that the model can capture some relationship between X and Y or does not reject it, in which case the model might not be able to do so. The test does not require sample splitting which is especially important when the sample size is small. The method is later applied to the tennis serve data and the prediction of the ball speed and the VA index. Two models were considered, a linear regression model and a deep learning model with a specific architecture. The test rejected the null hypothesis for both models in the case of the prediction of the ball speed, but it did not reject the null hypothesis for either of them in the case of the VA index prediction. In this case, the test gave evidence that a seemingly well-fitting model is not necessarily trustworthy.

2

Football activity recognition

2.1 Introduction

The world of sports has seen a continuous and rapid increase in the usage of technology in both competition and training during the last decades. Specifically in football (soccer), it is nowadays common to see players training and even playing competitive matches wearing vests with GPS trackers below their shirts. These vests can track the players during the whole training or match and give information about their location, distance traveled, speed, power, intensity, and heart rate, among others. These values can be processed to give the players, trainers, and journalists a very detailed analysis of each player's performance. During the last decade, the scientific community has shown important advancements in Human Activity Recognition (HAR) and, since technology and sports have developed a mutually beneficial relationship, there is a higher demand for systems capable of recognizing specific football-related activities.

If a coach, team, or player has information about which activities the player performs during a match or training, it enables a more detailed analysis of the player's performance. A more complete assessment of the player's movements and loads gives the teams the possibility of better training planning, a personalized follow-up to each player, and even a potential way to prevent, treat, and understand injuries. Nowadays, activity classification is done either manually or with the aid of cameras. To this end, it is required to have a large number of high-quality cameras equipped with artificial vision technologies or a considerable number of human labelers. Both of these two options are very expensive and can only be afforded by elite teams. This problem calls for the usage of a low-cost activity recognition system that is also affordable for smaller teams. Sensors are continuously being developed to be smaller, cheaper, and highly accurate (for example, [4]), so their use is a clear solution to this problem. They can be incorporated into the player's sportswear and provide reliable, real-time measurements of different body parts.

This chapter is based on *R. Cuperman, K.M.B. Jansen, M. Ciszewski. An end-to-end deep learning pipeline for football activity recognition based on wearable acceleration sensors, Sensors, 2022* [3].

This chapter studies the usage of deep learning-based models for football (soccer) activity recognition based on acceleration and angular velocity signals obtained from Inertial Measurement Unit sensors (IMUs). This is done in contrast to traditional machine learning approaches, in which a non-neural network-based model, such as k-nearest neighbors (kNN), decision tree, or support vector machine (SVM), is used to recognize an activity. Traditional methods make simpler and linear connections between the explanatory variables and the outcome, while the deep learning models are able to detect complex patterns (often non-linear). Furthermore, with this work, it is intended to use the raw signals from the IMUs and evaluate how robust the deep models are with respect to the signals acquired on different players. This is why little or no pre-processing will be applied to the sensor outputs, trying to recreate real-life scenarios. Different deep architectures will be proposed for the models and their performance and evaluation time will be examined. Furthermore, a complete training and evaluation pipeline will be designed, in which also the preparation of the training dataset and the strategy for the evaluation phase via a sliding window approach will be taken into account.

The way this chapter is structured is as follows. A literature review of the state of the art of methodologies based on sensors in the field of Human Activity Recognition is presented in Section 2. In it, both traditional machine learning and deep learning approaches are shown with their comparison of results, best practices, and challenges. Section 3.1 presents the dataset that was used for training and validating the models. An activity detection algorithm is presented, which is needed for the training phase. In Section 3.2, the training of several deep learning models is thoroughly explained and discussed; and in Section 3.3, the proposed evaluation pipeline is presented, with which the activities present in a given recording can be effectively recognized. The results of the training scenarios and the complete pipeline are shown in Section 4, which are then discussed in Section 5 with the conclusive remarks and future research recommendations.

2.2 Related work

2.2.1 Machine learning for football activity recognition

Identifying and recognizing human activities using signals obtained from Inertial Measurement Sensors (IMUs) is an area of machine learning and signal processing that has recently been studied by several authors. Recognition of daily activities, such as walking, climbing stairs, or sitting is especially popular amongst researchers, due to the availability of public domain datasets composed of these types of movements, and the possibility to easily compare the results with previous works [5–12]. On the other hand, studies on recognition of sport-specific activities (such as football, tennis, table tennis, or golf) are less frequent because building these types of datasets is difficult due to the costs involved in resources and time [13–19]. However, since the nature of signals is in many cases the same, it is possible to build upon the works of authors who have studied Human Daily Activity Recognition to build accurate and efficient methods designed for an application in a specific sport. The focus of our study is the application in football (soccer).

2.2.2 Traditional machine learning approaches

Human Activity Recognition applications can be developed based on two types of algorithms. The first one will be referred to as traditional machine learning approaches, in which the input features from the signals are manually defined and extracted. This process is not only heavily manual and subjective but is also extremely time-consuming [6]. Common choices of those features are the mean, standard deviation, maximum, minimum, kurtosis, and coefficients of the Fast Fourier Transform [5]. After that, classification is performed using traditional (non-neural network based) algorithms, such as Support Vector Machines (SVM), Decision Trees (DT), k-Nearest Neighbors (kNN), among others.

Several research papers have been written in which Human Activity Recognition is made with these approaches. In all of them, a manually selected set of features is extracted from the signals, usually containing a combination of time- and frequency-domain metrics. The majority of those works focus on daily human activities [6–12]. Others consider specific sports, such as table tennis [13], tennis [14], skateboarding [15], or volleyball [16]. Among the reviewed articles, ref. [17] studied Human Activity Recognition with application on football. They reported an accuracy of 88.6% on their dataset when using linear SVM. Some years later, ref. [20] developed other algorithms to further explore Football Activity Recognition. In that work, they developed a hierarchical architecture to recognize fullinstep kicks, side-foot kicks, or null activities (other movements different from kicks, such as dribbling or running). Their method includes an initial filtering of the signals followed by a peak detection algorithm. The detected peaks are then isolated, a set of several manually selected features are extracted for each peak, and finally, the movement is classified as either a kick or not. They achieved an accuracy of 94% using a Naive Bayes classifier.

With traditional machine learning algorithms, performing additional pre-processing is often necessary. Various pre-processing techniques are used: filtering [7, 13, 16–18], normalization, standardization [13, 16, 18], and the usage of norms [7, 13, 16, 17] are common practices. Stroke detection is also frequently used in sports-related applications [13, 14, 16, 18].

2.2.3 Deep learning approaches

"The traditional feature engineering methods are becoming more and more incapable" [21]. The second approach for Human Activity Recognition is based on neural networks. Deep learning approaches are able to extract complex and non-linear patterns from multidimensional data. This is one of the reasons why recent researchers in areas such as Human Activity Recognition have abandoned traditional approaches in favor of deep learning architectures [22]. An additional important consideration is the general lack of a pre-processing phase of the sensor signals prior to their input into the deep networks. When working with deep learning architectures, the raw signals from the sensors can be used directly.

Moreover, recent works have shown that the use of deep learning approaches for Human Activity Recognition is not only beneficial in terms of feature extraction, but also in achieving high accuracy. The study performed in [5] does an extensive review of different approaches for Human Activity Recognition, and concludes that, on average, traditional machine learning algorithms obtain an accuracy of 83.3%, while systems based on deep

learning achieve a much better 94.9%. They also expressed that there are more studies around traditional machine learning algorithms in comparison to deep learning ones, which shows that the use of the latter in Human Activity Recognition tasks is promising, but not yet fully explored.

No relevant scientific publications related to the use of deep learning approaches with sensor data for football activity recognition were found, whereas a large amount of works where these types of algorithms are used for Human Daily Activity Recognition is available. Since the nature of the signals and the ultimate goal of those studies are very similar to the objective of this chapter, their methodology and results were considered. Convolutional Neural Networks (CNNs) are very popular because "CNN-based models are able to extract and leverage latent feature representations in time series with high tolerance of time translation; thus, results outperform methods based on hand-crafted features" [19]. In that paper, many different deep architectures were reviewed: from CNN composed of consecutive convolutional layers to more complex and modern possibilities, such as inception CNN and residual CNN. However, " (...) CNN lacks the capability to capture temporal dependency in time-series sensory data. RNNs (Recurrent Neural Networks) are designed to model time series data, and are suitable for discovering relationships in temporal dimension" [23]. This is the reason why the usage of Recurrent Neural Networks was also evaluated by other authors, mainly using Long Short-Term Memory (LSTM) units. A very interesting approach is the combination of CNNs and RNNs to build a larger and, according to the authors of such papers, better performing network. Examples of such architectures are the ones proposed by [21–24], where usually an RNN (mainly composed of LSTM units) extracts temporal relations of the signals following a feature extraction process made by a CNN.

2.2.4 Background on deep learning

Unlike traditional machine learning algorithms, deep learning refers to the use of models based on stacked layers of artificial neural networks. By using multiple layers, it is possible to train the model to progressively extract and learn complex features from the inputs. This has a huge advantage over traditional machine learning algorithms since deep learning models perform both feature extraction and the specific machine learning task. There are many different types of neural networks based on variations of the basic structure of artificial neurons. In particular, this work focuses on two of those types: Convolutional Neural Networks and Recurrent Neural Networks (specifically LSTMs).

Convolutional neural

Convolutional Neural Networks (CNNs) are a type of deep learning models that were originally designed to tackle artificial vision and image processing problems. Studies on vision and perception of shapes in the human brain showed that the neurons responsible for those tasks have receptive fields, which means that each cell responds to a specific pattern. The combination of these simple patterns generates more complex ones that are furthermore combined so that the brain can finally interpret and understand the image. Convolutional neural networks try to replicate this behavior and have shown impressive results in a huge variety of machine learning applications. Although originally designed

for recognizing shapes and figures in images, CNNs can also be used to extract patterns from signals. The general idea of CNNs is based on two types of operations [25]:

- **Convolution**. In a convolutional layer, the input data are convolved with a certain number of kernels or filters. A filter k is traversed through all the points of the input image (or signal) and on each location, the convolution between the filter and the overlapping area of the data is calculated. The usage of each filter results in a new convolved image (or signal) called a feature map. The combination of all the k filters with different weights then generates a set of k feature maps.

- **Pooling**. It is a common practice to include pooling layers after convolutional ones. These types of layers condense information from spatially or temporally close points to reduce the data size. In neurology, a receptive field of a neuron is defined as the region in which the presence of a stimulus triggers the response of that particular neuron [26]. Pooling layers introduce the concept of receptive fields into CNNs because they condense information of neighboring points of the convolved data into a single value. Many types of pooling layers can be used, but one of the most common is the max pooling layer. This operation defines a small block (also called a filter) and runs it on top of the input data of the layer. At each point, the maximum of the values contained on the overlapping area of the data and the filter is extracted and the output image (or signal) is built with those maximum values. Pooling layers have the additional property that they reduce the size of the feature maps, which translates into fewer weights to be learned by the model. The outputs of these layers are smaller feature maps, but each element of those maps has information about its neighbors from the previous layer.

Recurrent neural networks and LSTM

Recurrent Neural Networks (RNN) are a type of deep learning models that are especially designed to work with data that have an underlying temporal sequence. Because of that, they are typically used for Natural Language Processing and Signal Understanding. With this type of data, it is important to take into account past information since the information is treated as a sequence and what has happened in the past has influence on what will happen in the future. Based on this idea, RNNs are able to "remember" prior inputs when generating the output, which are based not only on the current input vectors, but also on the so-called hidden state vectors that carry information about prior data [25, 27].

A major drawback is found when training RNNs due to a problem called "vanishing gradients". Basic RNNs are unable to remember long-term dependencies because the gradients that are used to train the model tend to disappear as the input sequence grows in length [28]. When training deep learning models, small gradients are undesirable as the training takes longer and, as a result, becomes less effective.

When working with sequence data such as signals produced by a set of sensors, it is important to have a model able to handle long-term dependencies. This is the reason why more complex RNN cells were developed. Long Short Term Memory (LSTM) cells are one of the most common RNN networks that are used to overcome that problem [27]. They are built upon the basic RNN cells in which prior information is captured in hidden cells and used to generate outputs.

In order to understand how LSTMs work, it is easier to analyze them by parts. One of the most important properties of LSTMs is their capacity to easily propagate information from one cell to another. This is implemented by the cell state c_t, which can run through the cell with only minor linear modifications. The cell state is the heart of the LSTM and is what carries past information of the input sequence. LSTMs can add or remove information from the cell state by using structures called gates, which are themselves regular feed-forward neural networks operating mainly with an input tensor x and a hidden tensor h. There are three gates in an LSTM unit:

- Forget gate This gate is responsible for deciding what information must be forgotten (removed) from the cell state. To do so, it concatenates the hidden state at time $t-1$ (h_{t-1}) and the current input x_t and calculates a value between 0 (forget) and 1 (keep) for each element of the cell state c_{t-1}.
- Input gate This gate is responsible for deciding what new information will be stored in the cell state and where. It is composed of two parts. The first part calculates candidate values to potentially update the cell state, and the second part decides which parts of the cell state must be updated with those candidate values. This completes the update of c_{t-1} into c_t determined by the forget and input gates.
- Output gate This gate is responsible for deciding which elements of the cell state will be given as the output of the LSTM unit. Only the desired parts of the cell state are output as the new hidden state values h_t.

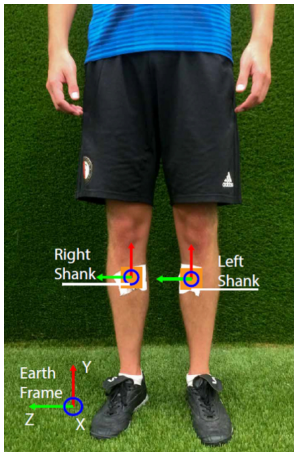
2.3 Materials and Methods

2.3.1 Data and preparation

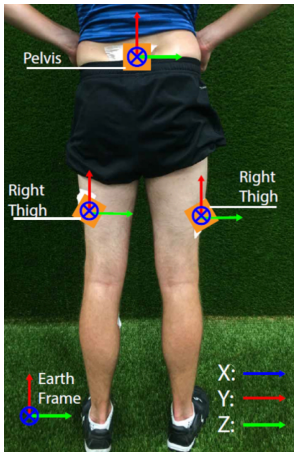
Data collection procedure

This work used data acquired by [29] for the training phase and an initial evaluation. For more information and details on the data collection procedures, see [29]. The experiments there conducted included 11 male soccer players with 5 IMUs (Ivensense MPU-9150) attached to their bodies in the following locations: pelvis, right thigh, left thigh, right shank, and left shank, as shown in fig. 2.1a and fig. 2.1b. Each one of the IMUs had a tri-axial accelerometer, gyroscope, and magnetometer. The range for the accelerometers was set to ±16 g and for the gyroscopes to ±2000°/s (the magnetometers' range was not specified. However, they were not used in this work). The sampling frequency of the signals was set to 500 Hz. The sensitivity (with 16 bits) of the accelerometers was set to 2048 LSB/g and of the gyroscopes to 16.4 LSB/(°/s). Each one of the participants executed a set of well-specified football-related activities, including passes, shots, jumps, sprints, among others. The experiments were designed in order to simulate an actual football match to have reliable and real movements. The most common football activities were used as the different classes to be recognized: pass, shoot, jump, sprint, and jog.

The use of those 5 IMUs allowed the measurement of the tri-axial accelerations and velocities of the 5 respective body parts related to each one of the activities performed by the subjects. This resulted in several signals representing the movements, each one of them with their manually annotated category (activity). It is important to note that in each experiment, before and after performing an activity, the subject walked or stood still for



(a) Placement of IMUs on lower legs in experiments. Taken from [30].



(b) Placement of IMUs on upper legs and pelvis in experiments. Taken from [30].

(c) New trousers prototype with fully embedded sensors. Taken from [1].

Figure 2.1: Placement of the sensors in the sensor trousers and the prototype.

some time. In order to more effectively train the models, a process of activity isolation was performed in which those low activity intervals were removed prior to training.

Activity isolation

As explained before, the recordings that were to be used to train the model included activities that were not isolated from surrounding noise and, in order to build a more reliable model, the signals were cleaned so that only the desired activities were present. This follows the logic that the deep learning model learns to extract features by itself. If a lot of irrelevant information would be present in the training phase, it could be possible that the model would learn some features from the low activity patterns and not from the actual activities. Therefore, to make sure that the model learns to recognize accurately the football-related movements, a procedure called activity isolation was developed. It prepared the recordings for the training phase by isolating the important activities from the aforementioned low-activity intervals.

A low-activity measurement is characterized, as its name suggests, by signals with low magnitude and variance which lies in contrast to the behavior of the signal during a high activity movement. This is especially true when we focus on the acceleration values. When a football player moves from being still or relaxed, the lower limbs accelerate quickly. This is the reason why only the accelerometer signals (and not gyroscope and magnetometer) were used for the activity isolation phase. On the other hand, this big change in acceleration between a low-activity interval and a high-activity one can happen in any of the measured body parts. A standing player can start to run with the right leg while another player can move the left leg first. This is also true with the axes (X, Y, and Z): when jumping the movement is primarily vertical, but when passing we expect the longitudinal component to be more present. In other words, the transition between a low- and a high-activity interval can be detected with any of the body parts and on any axis. This is the reason why the norm of X, Y, and Z axis of each sensor location is used. Each sensor is treated independently and, at the end of the process, they are combined for the final result.

In order to identify when a high activity happens, a baseline value for each signal is obtained: the mean value. When the player is still or walking, the norm of the signal is usually smaller than its mean. However, when the player performs a more intense activity, the norm of the signal presents peaks larger than its mean value. So, the beginning of a high-activity measurement can be found by identifying the moment when the norm of the signal exceeds a threshold based on the mean value. To avoid small meaningless peaks, the algorithm looks for the window of fixed size (based on domain knowledge, we chose 50 timesteps = 0.1 s) in which the norm of the signal exceeds the threshold for each of the timepoints of such window. Similarly, the end of the high-activity interval is identified by placing the window in the opposite direction.

We found that activities such as sprints and jogs required the mean of the signal to be the aforementioned threshold, while movements, such as shots, jumps, and passes, were better isolated when using 1.5 times the mean as the threshold. By understanding the nature of these two groups of movements, the former group was called periodic activities, in which the activity is performed in a periodic manner; and the latter explosive activities, in which the activity is performed only once without repetition. Since explosive activities tend to be shorter and without repetitive patterns, the Interquartile Range (IQR) was proposed as the

metric to use to discriminate between both groups of movements. To classify an activity as periodic or explosive, the Euclidean norm of all the accelerometer signals of the recording was taken, then this resulting signal was normalized between 0 and 1, and finally, the IQR was calculated. In fig. 2.2, the distribution of the IQR values for periodic and explosive activities can be seen. This plot shows that the IQR is a good metric to distinguish between both types of movements if a threshold is chosen. It was defined that if the IQR exceeded 0.12, the recording was considered as a periodic movement or, otherwise, as an explosive movement. This IQR-based classifier showed to have an accuracy of 99.42%, as shown in the confusion matrix on the top right of fig. 2.2.

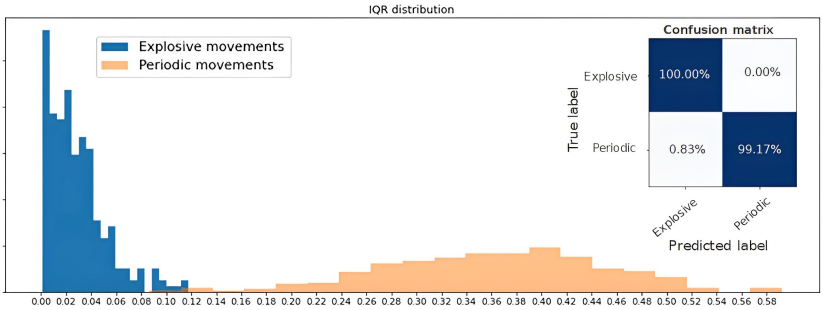


Figure 2.2: IQR distribution of explosive and periodic movements. On the top right, the confusion matrix of the periodic- vs. explosive-activity classifier based on a threshold of 0.12 on the IQR.

Algorithm 1 presents the full procedure of activity isolation. Its application showed to be very effective in isolating high activities from low-activity periods. Some examples of results obtained with this algorithm are shown in fig. 2.3. In these images, all the signals are superimposed for visualization purposes. The isolated high activities are shown with white background, while the low-activity periods are grayed out.

Algorithm 1: Threshold-based activity isolation

Result: Initial and final times of a detected activity isolated from low intensity periods.
Take one recording with an activity present;
Take only the accelerometer data of the 5 locations: pelvis, right shank, right thigh, left shank, and left thigh;
Take the Euclidean norm of the 5 signals. Name it s_5;
Calculate the IQR of s_5;
if $IQR \leq h_{IQR}$ **then**
  Mark the recording as an explosive movement;
else
  Mark the recording as a periodic movement;
end
Take the Euclidean norm of the X, Y, and Z axis of each one of the 5 locations.;
Name those 5 signals s_p, s_{rs}, s_{rt}, s_{ls}, s_{lt};
for $s \in \{s_p, s_{rs}, s_{rt}, s_{ls}, s_{lt}\}$ **do**
  Calculate the mean value μ of the signal;
  if *The respective* s_5 *was classified as a periodic movement* **then**
    Set $thr = \mu$;
  else
    Set $thr = 1.5\mu$;
  end
  Find the first timestep where the signal and the following 50 timesteps are larger than the previously defined threshold. Take that timestep as the start of the activity for that bodypart s;
  Find the last timestep where the signal and the previous 50 timesteps are larger than the previously defined threshold. Take that timestep as the end of the activity for that bodypart s;
end
Take the minimum among the starts of activity from the previous step. Subtract 250 timesteps (if possible). This is the overall start of the activity. Call it t_0;
Take the maximum among the ends of activity from the previous step. Add 250 timesteps (if possible). This is the overall end of the activity. Call it t_f;
return t_0, t_f

2.3.2 Neural network architecture

After the high activities were effectively isolated from low-activity periods, the training and validation datasets were built. The five most common activities in football practice were selected as the classes to be recognized by the deep learning models: shot, pass, jump, jog, and sprint. Since the data were recorded for activities following a well-defined script (e.g., 10 jogs each followed by a shot), the movements were manually labeled. All the examples of those activities were isolated from the original dataset and a window segmentation process was applied to them to extract the training and validation examples. This process consisted of a one-second-long window traversing through the recordings, extracting, at

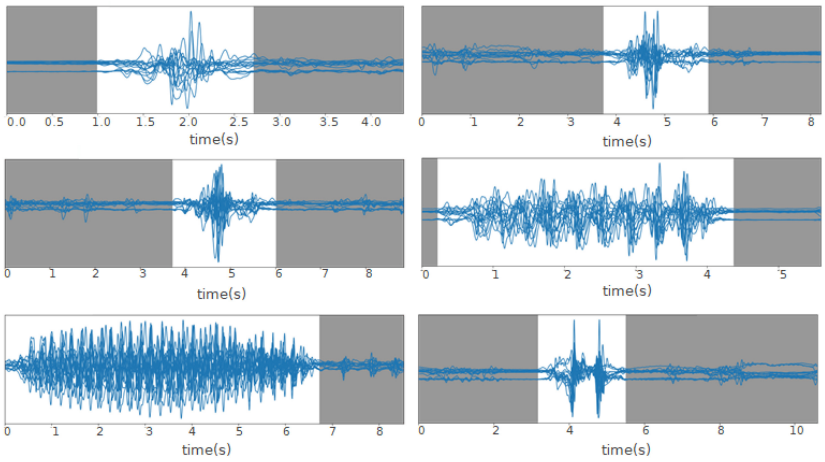


Figure 2.3: Examples of periods of high activity (white background) obtained after applying Algorithm 1. The examples show accelerometer data related to a shot, pass, pass, jog, sprint, and jump.

each time, the respective interval from the original signal. In order to capture temporal dependencies, an overlap of 75% was used when extracting the one-second-long windows, meaning that every 250ms of a recording, a new interval of 1s was extracted.

This set of one-second-long windows was divided in two via the technique of random subsampling cross-validation: a train dataset, composed of a random 70% set of the samples; and a test dataset, with the remaining 30%. The former was used to train the models and the latter to evaluate them with unseen samples. This train-test split was made by randomly sampling among the recordings of all the subjects. In order to obtain robust results, this procedure was repeated five times (which resulted in five different randomly selected train and test datasets), so that at each time each train and test dataset contained samples from different subjects. The accuracy metrics presented in this chapter are the averages of the five repetitions. In other words, the train-test split was performed using a 5-fold random subsampling validation [31]. This approach is similar in effectiveness to 5-fold cross validation, but slightly faster. Additionally, only accelerometer and gyroscope data were used. Magnetometer data were ignored to reduce the dimensionality of the problem. The datasets were balanced via undersampling of the most frequent classes.

As explained before, HAR tasks could benefit from the usage of Convolutional Neural Networks and Recurrent Neural Networks. The former would be responsible for extracting relevant patterns of features from signals and the latter would use those features and give them temporal meaning by understanding the signals as a time series. A combination of both types of layers could be, in theory, very powerful.

The networks proposed follow the same general architectures shown in fig. 2.4. This chapter explored models based solely on CNNs (fig. 2.4a), RNNs (fig. 2.4b), and on a combination of CNNs and RNNs (fig. 2.4c). In fig. 2.4, the connections referred as CNN sub-network and RNN sub-network are shown in fig. 2.5 and they consist of specific types of convolutional and/or recurrent layers that will be explained below.

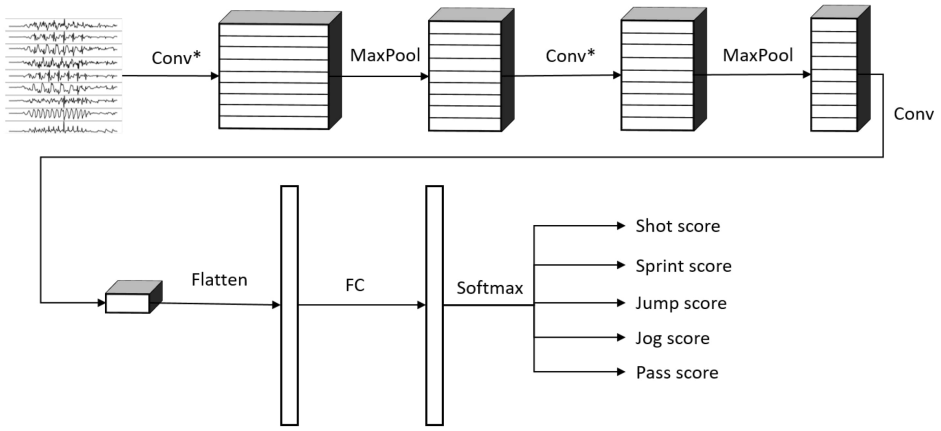
One of the most important elements of this chapter is the evaluation of different variations of convolutional layers. In fig. 2.4 and 2.5, the asterisks in the convolutional parts represent the implementation of the different types of convolutional layers proposed. It is important to note that, even if all the convolutions are theoretically one-dimensional (because the convolution only happens across the temporal dimension), some of them are referred to as one-dimensional and some others as two-dimensional to distinguish among them. By one-dimensional convolution, we refer to convolutions in which the spatial dimension of the filter is 1, so that each signal is processed independently and the filters do not process more than one signal at the same time. By two-dimensional convolutions, we refer to convolutions in which the spatial dimension of the filter is more than 1, so that several signals are convolved simultaneously. The following variations of convolutions were built:

- 1DCNN weight sharing:
  One-dimensional convolutions with the same filters for all the signals. In this type of convolution, filters of size $1 \times m$ are used, where m is a hyperparameter that determines the timesteps used in the convolution. The 1 implies that each signal is convolved alone. Additionally, weight sharing means that the same filters are used for all the signals. fig. 2.6 explains this logic. Note that each sensor is composed of three signals (X, Y, and Z axis of the sensor). The convolutions are made for each signal using the same set of k filters (represented in red).

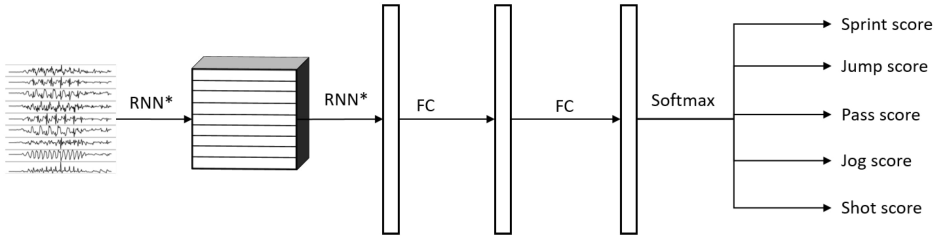
- 1DCNN per sensor:
  One-dimensional convolutions with the same filters for all the signals of the same sensor, but different filters for each sensor. In this type of convolution, filters of size $1 \times m$ are used, where m is a hyperparameter that determines the timesteps used in the convolution. The 1 implies that each signal is convolved alone. However, each sensor has its own set of k filters, meaning that the filters are not shared among the sensors. fig. 2.7 shows this logic. The convolutions are made for each sensor using, for each one of them, a different set of k filters (but the same set of filters are used for the three axis of the same sensor). The different sets of filters are represented with different colors in the figure.

- 1DCNN combined:
  Combination of 1DCNN weight sharing and 1DCNN per sensor. Both types of convolutions are performed and their results (feature maps) are concatenated one on top of the other. Figure 2.8 shows this logic.

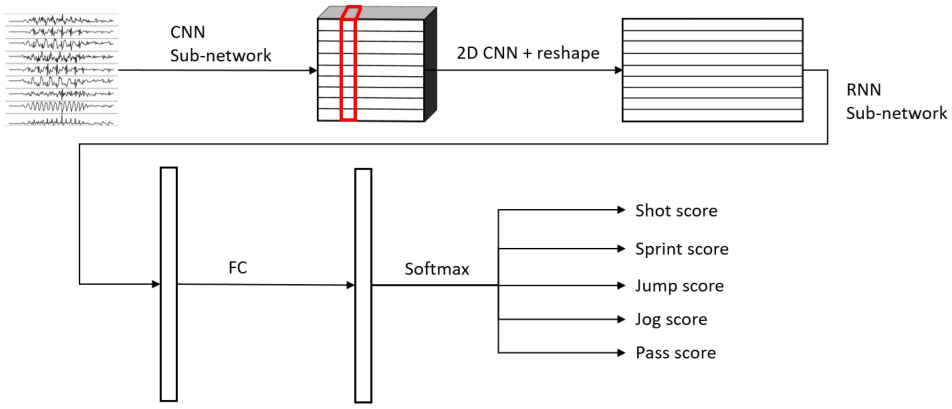
- 2DCNN weight sharing:
  Two-dimensional convolutions with the same filters for all the sensors. In this type of convolution, filters of size $3 \times m$ are used, where m is a hyperparameter that determines the timesteps used in the convolution. The 3 means that the 3 axes of the same sensor are used together in the convolution. In order to convolve each sensor by itself so that specific patterns can be extracted by combinations of the X, Y, and Z signals of the same sensor, a spatial stride of 3 is used. Additionally, weight sharing means that the same filters are used for all the sensors. fig. 2.9 explains this logic.



(a) General architecture for models based on CNNs

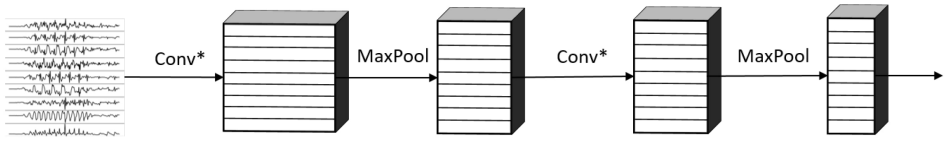


(b) General architecture for models based on RNNs

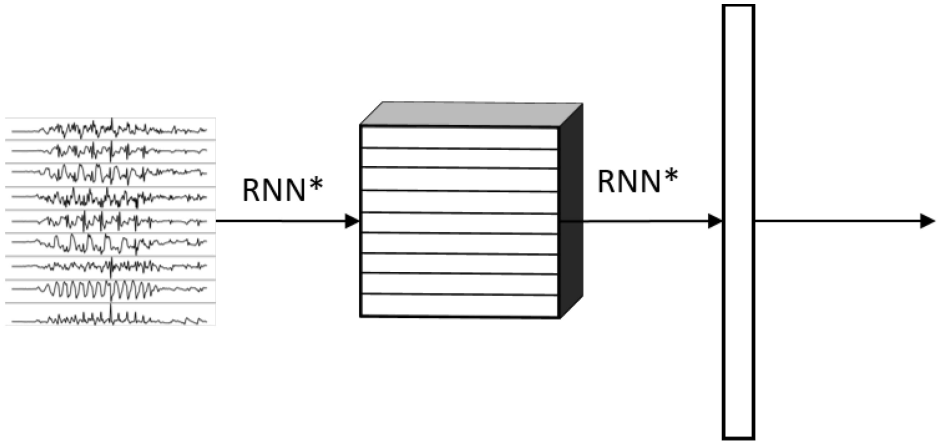


(c) General architecture for models based on combination of CNNs followed by RNNs

Figure 2.4: General architectures of the different types of models built. Three types of models were evaluated: using only CNN, only RNN, or a combination of both. The asterisks on the Conv layers mean the usage of a variation of a convolutional layer. The asterisks on the RNN layers represent either LSTMs or bidirectional LSTMs. (FC = Fully Connected Feed Forward Neural Network).



(a) General CNN sub-network for the models that combine CNNs and RNNs.



(b) General RNN sub-network for the models that combine CNNs and RNNs.

Figure 2.5: General CNN and RNN sub-networks for the models that combine CNNs and RNNs. The asterisks on the Conv layers mean the usage of different variations of CNNs. The asterisks on the RNN layers represent either LSTMs or bidirectional LSTMs. (FC = Fully Connected Feed Forward Neural Network.)

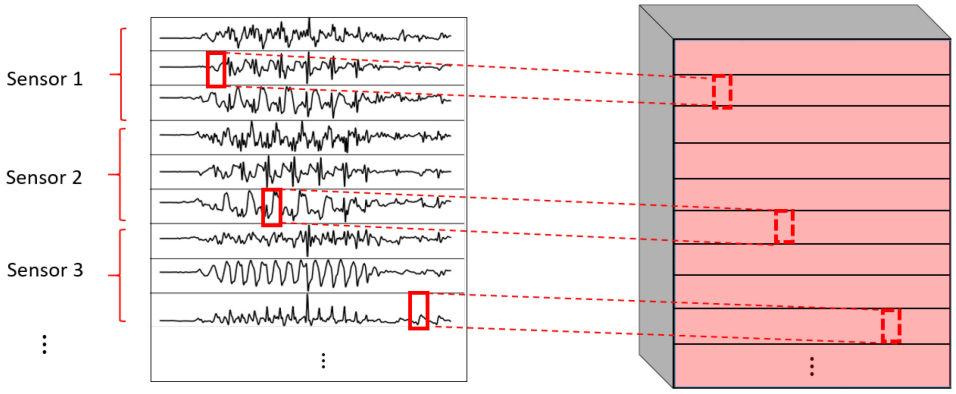


Figure 2.6: 1DCNN weight sharing convolution logic.

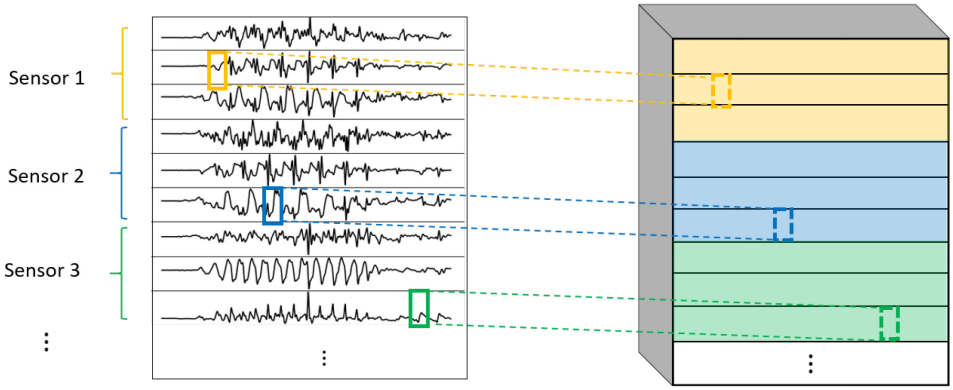


Figure 2.7: 1DCNN per sensor convolution logic.

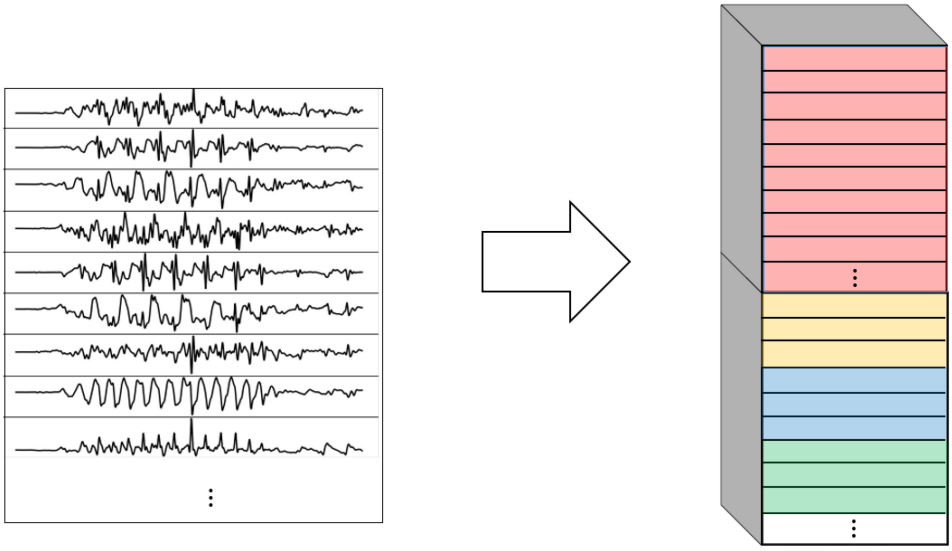


Figure 2.8: 1DCNN combined convolution logic.

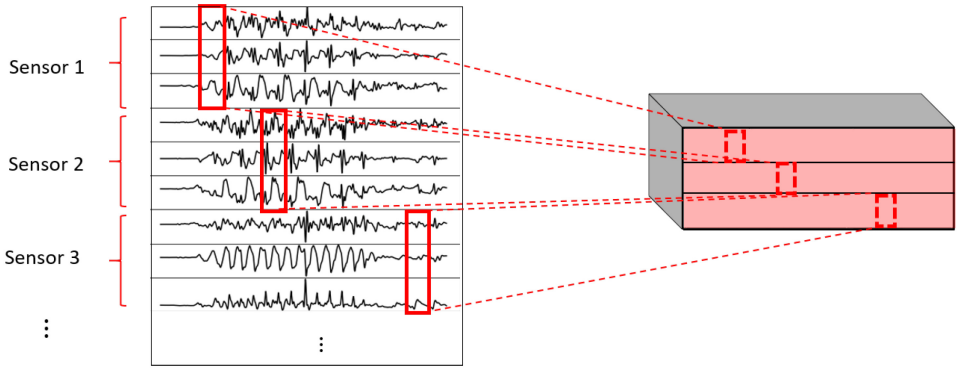


Figure 2.9: 2DCNN weight sharing logic.

- 2DCNN per sensor:
  Two-dimensional convolutions with different filters for each sensor. In this type of convolution, filters of size $3 \times m$ are used, where m is a hyperparameter that determines the timesteps used in the convolution. The 3 means that the 3 axes of the same sensor are used together in the convolution. In order to convolve each sensor by itself so that specific patterns can be extracted by combinations of the X, Y, and Z signals of the same sensor, a spatial stride of 3 is used. fig. 2.10 explains this logic. The convolutions are made for each sensor using, for each one of them, a different set of k filters. The different sets of filters are represented with different colors in the figure.

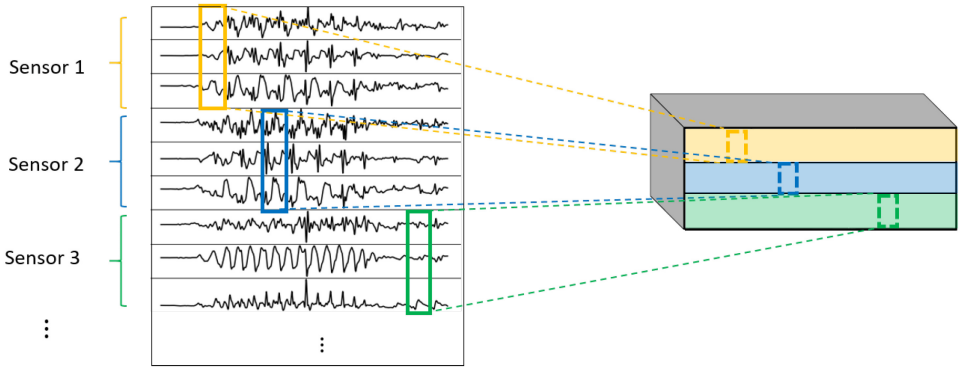


Figure 2.10: 2DCNN per sensor logic.

- 2DCNN all sensors:
  Two-dimensional convolutions made across all the sensors (thus also signals) at once. In this type of convolution, filters of size NumSensor $\times m$ are used, where m is a

hyperparameter that determines the timesteps used in the convolution. Figure 2.11 explains this logic.

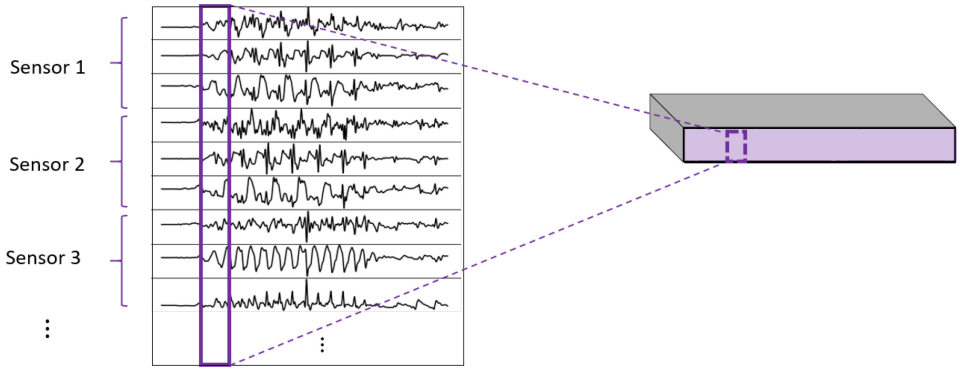


Figure 2.11: 2DCNN all sensors logic.

- 2DCNN combined:
  Combination of 2DCNN weight sharing, 2DCNN per sensor, and 2DCNN all sensors. The three types of convolutions are performed and the resulting feature maps are concatenated one on top of the other. Figure 2.12 shows this logic.

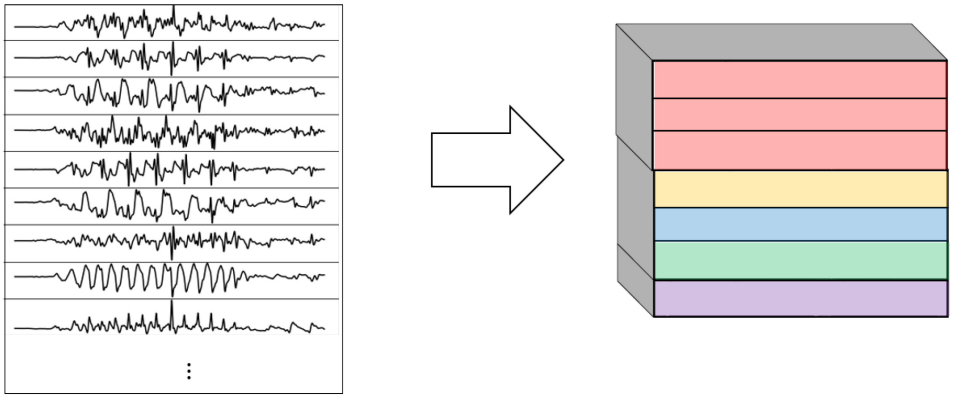


Figure 2.12: 2DCNN combined logic.

Specific details of the convolutions used on the different CNN sub-networks are shown in Table 2.1. There, the following convention is used:

- Conv(f,(m,n),s): Convolutional layer with f filters of size (m,n) with s strides in the vertical (spatial direction) followed with *act* activation function. In all the cases, a ReLU activation function ($f(x) = \max(0, x)$) is used.

- MaxP(m,n): Max Pooling layer with filters of size (m,n), as explained in Section 2.2.4.
- d: Number of spatial dimensions of the output tensor of the previous layer.

CNN Sub-Network	Layer 1	Layer 2	Layer 3	Layer 4
1DCNN weight sharing	Conv$(16,(1,5),1)$	MaxP$(1,4)$	Conv$(32,(1,5),1)$	MaxP$(1,4)$
1DCNN per sensor	Conv$(16,(1,5),1)$	MaxP$(1,4)$	Conv$(32,(1,5),1)$	MaxP$(1,4)$
1DCNN combined	Concatenation of 1DCNN variations			
2DCNN weight sharing	Conv$(32,(3,5),3)$	MaxP$(1,4)$	Conv$(64,(1,5),1)$	MaxP$(1,4)$
2DCNN per sensor	Conv$(32,(3,5),1)$	MaxP$(1,4)$	Conv$(64,(1,5),1)$	MaxP$(1,4)$
2DCNN all sensors	Conv$(32,(d,5),1)$	MaxP$(1,4)$	Conv$(64,(1,5),1)$	MaxP$(1,4)$
2DCNN combined	Concatenation of 2DCNN variations			

Table 2.1: Details of layers used on different CNN sub-networks. Recall that the convolutions named *weight sharing* use the same kernels for all the sensors and signals, while the convolutions named *per sensor* use independent kernels for each sensor.

The last convolutional layer (which occurs just before the RNN sub-network) is defined as Conv$(128,(d,1),1)$. The LSTM and bidirectional LSTM (bLSTM) layers are, in all cases, composed of 128 units. The fully connected layers after the sub-networks are also built with 128 units with ReLU activation function except for the final fully connected layer, which has 5 units (one per each class) and uses a softmax activation function instead. Additionally, to reduce possible overfitting, dropout layers (not shown in the figures) are applied before each fully connected layer.

All of the different architectures were trained using both accelerometer and gyroscope data or only accelerometers to evaluate the influence of gyroscopes on the classification of movements. To train the models, the ADAM optimization algorithm [32] was used to optimize a categorical cross-entropy loss function. In all cases, the chosen parameters were $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$. The learning rate ($\alpha$) was set to decay during the learning phase using a learning rate scheduler. By doing this, the model was able to take large steps towards the optimum during the initial phases of the training, and, as the model approached this point, the steps taken were smaller. This resulted in a better and faster training scheme. The learning rate schedulers for the models were defined as:

- For all the models without any LSTM or bLSTM component, the learning rate was initialized at 10^{-3}. After every 10 epochs of training, it was reduced to its 75%.

- For all the models with only LSTM or bLSTM (no convolutional part), the learning rate was initialized at 10^{-4}. After every 10 epochs of training, it was reduced to its 50%.
- For all the models that combined a CNN part with a LSTM or bLSTM part, the learning rate was initialized at $5 \cdot 10^{-5}$. After every 10 epochs of training, it was reduced to its 75%.

The training was made with batches of 32 samples for at most 200 epochs. Additionally, an early stopping criterion was defined in order to reduce overfitting and unnecessary training: the validation loss was monitored and if it did not improve (reduce) for 5 consecutive epochs, the training was halted. Each model was trained 5 times using different train-test partitions so that every case used a different subset of data for training and testing. The resulting accuracies of the trainings of each model were averaged to obtain their overall performance.

2.3.3 Post-processing and evaluation

Once the models were successfully trained, the next step consisted in using the model to actively recognize and classify the activities present in a recording. A sliding window approach was followed: a window of 1s (500 timesteps) is swept through the recording and, for each position of the window, a prediction of the activity is made. By doing this, several windows will have periods where no relevant activity is performed because the player is just standing or walking passively. Therefore, the model also needs to be able to recognize these low-activity periods and classify them as such. To allow the model to recognize these low-activity periods, a binary classifier on top of the already trained deep learning-based model was built. That binary classifier was responsible for recognizing whether a window captured a low activity period or not. The *low activity* class could also be included as another class in our models, however, we opted for detecting low activities using a binary classifier before applying the deep model. This is due to the significant differences between low and high activities, which can be exploited to accurately distinguish them.

The sliding window evaluation phase is shown in fig. 2.13. A window of the same length as the one used for the training phase (1s = 500 timesteps) is traversed through the recording, extracting at each time a window of such length. The extracted window is first classified by the binary classifier as either a high or low activity window. If the classification returns low activity, the window is understood as such. However, if the binary classifier predicts that the window corresponds to a high activity, then the window is passed through the deep learning model that further classifies the interval as one of the activities: shot, sprint, jump, jog, or pass. This means that, at the end of this process, the window is classified as either shot, sprint, jump, jog, pass, or low activity. This is what we call a prediction. The sliding window moves and a new window is extracted and then classified following the same process.

To build the binary low- vs. high-activity classifier, different metrics were extracted from low- and high-activity windows and evaluated for their discriminant power between both classes. The following values were evaluated for the Euclidean norm of the accelerometer signals:

- mean,

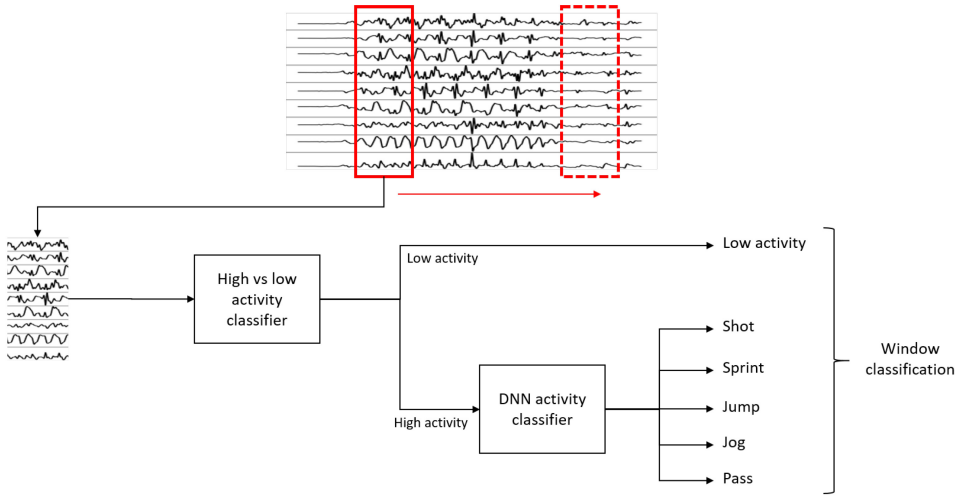


Figure 2.13: Sliding window evaluation diagram.

- standard deviation (std),
- coefficient of variation (CV),
- interquartile range (IQR),
- range.

The distributions of the metrics for those two groups were examined to identify which one of the statistics would have more discriminatory power between both categories. Additionally, a two-sample Kolmogorov–Smirnov (KS) test was used to evaluate the similarity of these high- and low-activity distributions. The larger the KS value, the more certain we are that both samples come from different distributions. The resultant KS values for the selected metrics are summarized in Table 2.2.

Metric	KS value
Mean	0.9185
Std	0.9302
CV	0.7997
IQR	0.9237
Range	0.9155

Table 2.2: Two-sided KS values for the metric distributions of high- and low-activity window.

The standard deviation was chosen as the metric to build the binary classifier since its KS value was the largest. Additionally, the distribution plots of the standard deviation of

both categories (high- and low-activity windows) could be separated using a threshold, as seen in fig. 2.14.

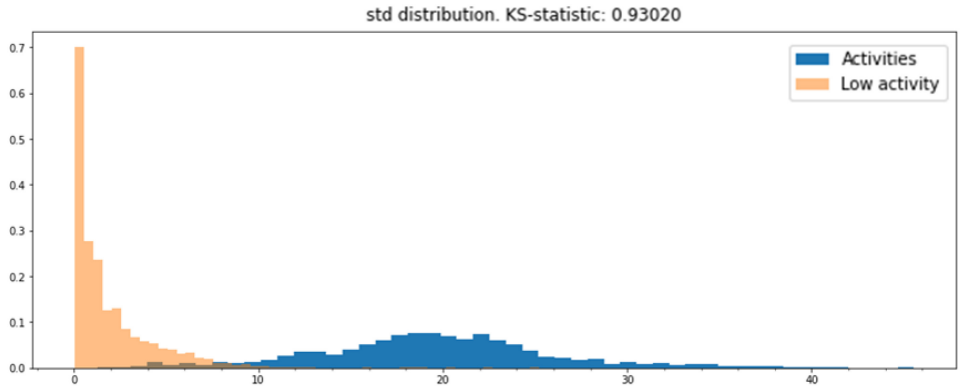


Figure 2.14: Standard deviation distribution of low- and high-activity windows. Both distributions can be separated using a threshold value.

To define the threshold that should be used for the standard deviation so that the separation between low and high activity was as clean as possible, different values were used to classify a set of new unseen windows with high or low activities: if the standard deviation of the Euclidean norm of the accelerometer signals was larger than a given threshold, the window would be classified as a high activity, and as a low activity otherwise. The F1 score was calculated for each one of the thresholds for the standard deviation and the value where the F1 score was the largest was taken as the optimal threshold. This experiment is depicted in fig. 2.15 and shows that a threshold of 8.5 gives the best F1 score of 96.67% (and accuracy of 96.56%), which is high enough to consider this binary classifier as good performing.

The complete sliding window evaluation diagram is shown in fig. 2.13. As it can be seen, the low- vs. high-activity binary classifier lies on top of the deep learning model. We recommend using sliding steps between $10ms$ and $100ms$ for the windows (windows of 99% and 90% overlap, respectively). It is important to note that, even if the training of the model was made with windows of 75% overlap, this value does not need to be the same as the one chosen for the evaluation phase. The length of the window ($1s = 500$ timesteps), on the other hand, must be exactly the same as the one used for the training.

The sliding window approach at the evaluation phase has the issue that there are fewer predictions than timesteps of the recording since the windows are not evaluated at each time point. Furthermore, since there is an overlap between the sliding windows, all the timesteps are evaluated (and therefore predicted) several times by different windows. To solve this problem, the proposed best-score post-processing method post-processes the predictions so that we can associate a unique activity to each moment of the recording. The proposed method initially assigns all the timesteps of window i to the prediction of window i. Then, the final prediction for each timestep is the prediction with the largest confidence among the ones of all the windows that contained that particular timestep.

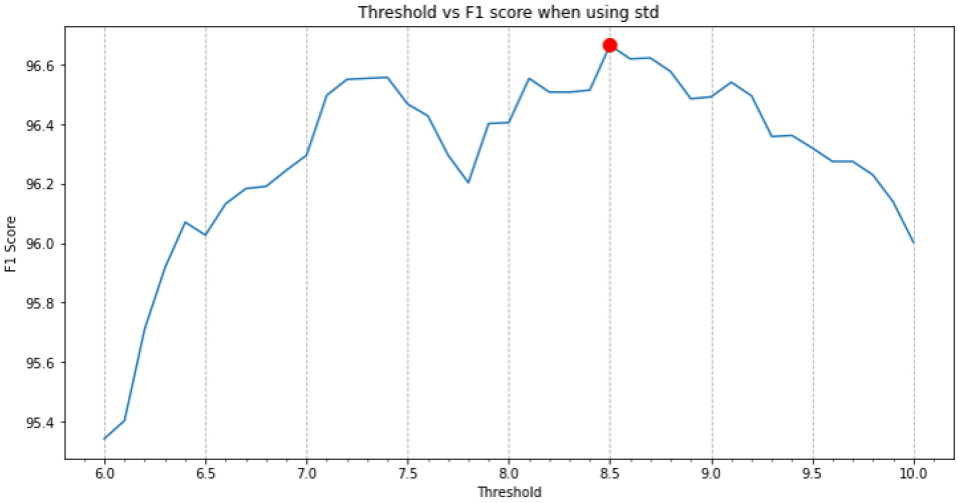


Figure 2.15: F1 scores for different standard deviation thresholds when classifying high- and low-activity windows. The red dot indicates the chosen threshold.

The softmax activation function at the last layer of the neural network was used to calculate the confidences of the predictions. Since the problem is a multiclass classification task, the last layer of the model includes a softmax function. The output of this activation function can be understood as the probability distribution over all the possible categories. The maximum among these probabilities can be taken as the confidence of the predicted class. Therefore, it was defined that a window was considered as "other high activity" if it had confidence lower than a certain threshold (in this work, it was defined as 95%), no matter the prediction that it initially had. With this addition, it was possible to identify movements different from the five predefined ones, such as turns.

Fig. 2.16 explains the previously explained process in a graphical, simplified way with a toy example. Suppose that the multicolored horizontal bar on the top of the image represents the recording. That recording is traversed with a sliding window and, for each position of the window, a prediction of the activity is made. Those windows are depicted in the figure as rectangles with thick borders and are named $W1, W2, W3, \ldots$ The prediction made for each window is represented by a color: blue, red, or green. Then, the horizontal bar on the top shows the predictions made by the sliding windows using that color code. That bar is the output of the evaluation pipeline explained in fig. 2.13. The best-score post-processing method assigns the prediction of the window to all the timesteps contained in that window, as it can be seen with the small horizontal colored bars in the middle of the figure. Each one of those predictions is composed of the recognized activity (red, blue, or green in the figure) and the confidence of the prediction (light to dark tone of the color). Then, as it was mentioned, for each timestep of the recording, the prediction with the largest confidence is taken as the final prediction of the respective timestep. The horizontal bar at the bottom of the figure shows the resulting predictions obtained with the best score

post-processing method. This post-processing option gives more importance to predictions with high confidence. It has the additional ability of "cleaning" the results of short, isolated, low-score predictions surrounded by predictions with larger confidence.

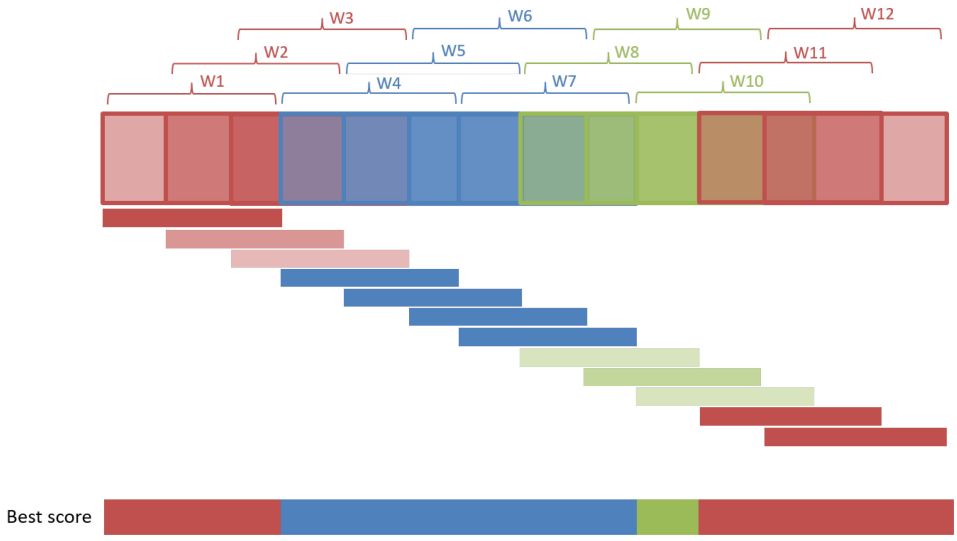


Figure 2.16: Best-score post-processing option. The darker the color, the larger the confidence of the prediction.

After post-processing, an outlier removal procedure is applied to the predictions. Even if the predictions are cleaned with the post-processing phase, there is still a chance that short peaks of isolated activities remain. An activity performed by a player cannot last less than a certain amount of time in real life. So, if a prediction of a movement lasts less than τms, the predicted activity of that interval is replaced with the predicted activity of the next interval that lasts at least τms. Good results were obtained when choosing τ to be between $100ms$ and $300ms$.

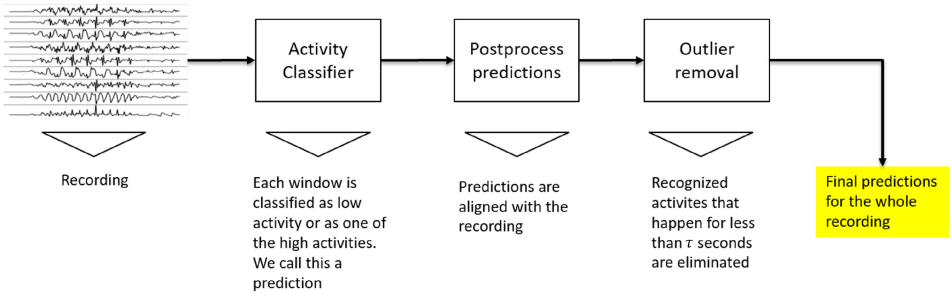


Figure 2.17: Complete sliding-window evaluation procedure.

The whole sliding-window evaluation process can be summarized with the diagram shown in fig. 2.17. Each position of the sliding window is classified, then the prediction is post-processed, and, finally, the outliers are removed. The block called *Activity Classifier* in the diagram is the activity classification process, depicted previously in fig. 2.13.

2.4 Results

2.4.1 Training results

As explained before, deep learning models composed of a combination of CNNs and RNNs were proposed, built, and trained to recognize shots, passes, jumps, jogs, and sprints (fig. 2.4). The training and validation datasets were balanced via undersampling, so that the accuracy could be used as the metric to evaluate the performance of the model. The prediction accuracy is defined as:

$$\text{Accuracy} = \frac{\text{Number of correctly classified samples}}{\text{Number of samples}}$$

When working with deep neural networks, it is a good practice to perform a normalization or standardization of the input data. This has mainly two benefits: the model can be trained faster, and the model can learn to better generalize from the input data [33]. Therefore, the models were trained with and without an initial normalization/scaling of the input signals. Since the signals have both positive and negative values and we wanted to capture information about their positiveness and negativeness, instead of using a min-max normalization, we used a max-abs scaler: we divided by the maximum absolute value of the signal. This process brings all the values to the range $[-1, 1]$. In particular, if the maximum absolute value is found in the positive range, then the values are transformed to the range $[-x, 1]$ where $x \leq 1$. Similarly, if the maximum absolute value is found in the negative range, the transformed range is $[-1, x]$. The resulting series has the same shape and structure as the original, but its values lie between -1 and 1 without losing the positive-negative relationship. The max-abs scaling does not shift the data nor destroys the sparsity between positive and negative values. This type of scaling was applied to each one of the signals and the models were trained with these new scaled windows. To have a fair comparison with the non-scaled models, all the models were trained with the same architectures, parameters, and algorithms as the ones previously built. Only the initial learning rate of the models had to be increased 10 times due to much smaller absolute values for the scaled signals. Note that in real-life applications, it would be necessary to perform this scaling with respect to a calibration recording: the player would perform a certain sequence of activities before the competition or training and that recording would be used to scale the subsequent data.

We trained several models based on the proposed convolutional operations and their combination with recurrent layers. The mean and standard deviation of the accuracies of the five trainings of each model can be seen in Tables 2.3 and 2.4. In those tables, a red-to-blue color code was used to visualize the accuracies on the test dataset, where red is worse and blue is better. No color code was given for the train dataset for visualization clarity.

The main goal of this research was not only to have a highly accurate model when recognizing football activities, but it was also desired that those classifications could be

	Acc + Gyro			
	Normalized		Unnormalized	
	Train	Test	Train	Test
1D CNN weight sharing	99.93%	97.53%	99.13%	94.36%
1D CNN per sensor	99.34%	97.26%	98.35%	93.92%
1D CNN combined	99.46%	97.21%	98.66%	93.97%
2D CNN weight sharing	99.29%	96.44%	98.80%	93.59%
2D CNN per sensor	99.44%	96.55%	99.32%	94.79%
2D CNN all signals	99.76%	97.81%	97.58%	92.88%
2D CNN combined	99.44%	98.08%	99.34%	95.34%
1D CNN weight sharing + LSTM	97.81%	96.11%	99.01%	96.05%
1D CNN per sensor + LSTM	99.20%	97.10%	98.31%	95.95%
1D CNN combined + LSTM	98.54%	97.04%	98.87%	96.55%
2D CNN weight sharing + LSTM	98.54%	96.71%	98.87%	96.27%
2D CNN per sensor + LSTM	98.31%	96.71%	99.32%	96.71%
2D CNN all signals + LSTM	98.87%	97.32%	98.94%	95.45%
2D CNN combined + LSTM	99.08%	97.53%	99.32%	96.71%
1D CNN weight sharing + bLSTM	99.39%	98.03%	99.08%	96.55%
1D CNN per sensor + bLSTM	99.51%	97.86%	98.66%	96.38%
1D CNN combined + bLSTM	99.44%	98.25%	99.22%	96.71%
2D CNN weight sharing + bLSTM	99.39%	97.48%	99.20%	96.22%
2D CNN per sensor + bLSTM	99.48%	97.04%	98.99%	96.11%
2D CNN all signals + bLSTM	99.13%	97.26%	98.45%	94.96%
2D CNN combined + bLSTM	99.46%	97.32%	99.29%	96.00%
LSTM	63.51%	60.44%	91.48%	77.48%
bLSTM	80.49%	76.71%	99.65%	87.07%

Table 2.3: Comparison of mean prediction accuracies between the original unnormalized models and the normalized ones. Five runs. A red-to-blue color code is used to facilitate the visualization of the values on the test dataset, where red is worse and blue is better.

	Acc + Gyro			
	Normalized		Unnormalized	
	Train	Test	Train	Test
1D CNN weight sharing	0.09%	0.87%	0.81%	1.33%
1D CNN per sensor	0.41%	0.74%	1.68%	1.78%
1D CNN combined	0.44%	0.70%	0.33%	1.48%
2D CNN weight sharing	0.20%	0.81%	0.49%	1.60%
2D CNN per sensor	0.29%	0.37%	0.46%	1.42%
2D CNN all signals	0.15%	0.67%	0.87%	1.38%
2D CNN combined	0.55%	0.35%	0.44%	1.42%
1D CNN weight sharing + LSTM	0.79%	0.56%	0.52%	1.21%
1D CNN per sensor + LSTM	0.58%	1.44%	1.22%	0.87%
1D CNN combined + LSTM	0.54%	0.56%	0.62%	1.60%
2D CNN weight sharing + LSTM	0.50%	0.65%	0.92%	1.00%
2D CNN per sensor + LSTM	0.76%	1.47%	0.39%	1.31%
2D CNN all signals + LSTM	0.75%	0.66%	0.58%	1.02%
2D CNN combined + LSTM	0.30%	0.83%	0.37%	1.05%
1D CNN weight sharing + bLSTM	0.44%	0.63%	0.34%	0.73%
1D CNN per sensor + bLSTM	0.53%	0.68%	0.60%	0.53%
1D CNN combined + bLSTM	0.27%	0.66%	0.53%	1.54%
2D CNN weight sharing + bLSTM	0.65%	0.97%	0.41%	0.96%
2D CNN per sensor + bLSTM	0.35%	1.17%	0.75%	1.10%
2D CNN all signals + bLSTM	0.55%	0.62%	0.96%	0.99%
2D CNN combined + bLSTM	0.92%	0.82%	0.36%	1.50%
LSTM	19.13%	18.19%	2.17%	3.36%
bLSTM	24.22%	20.14%	0.24%	2.25%

Table 2.4: Comparison of standard deviation of the prediction accuracies between the original unnormalized models and the normalized ones. Five runs. A red-to-blue color code is used to facilitate the visualization of the values on the test dataset, where red is worse and blue is better.

made in a short time. To demonstrate the computational efficiency of the proposed deep learning models, traditional machine learning models were trained, and the evaluation times for both types of models were computed. In particular, the following classifiers were built: k-Nearest Neighbors (kNN), Naïve Bayes (NB), Quadratic Discriminant Analysis (QDA), Decision Tree (DT), Random Forest (RF), SVM with linear kernel version ECOC (Error Correcting Output Codes) (SVM-l ECOC), SVM with Gaussian kernel version ECOC (SVM-rbf ECOC), SVM with linear kernel version One-vs-One (SVM-l OvO), SVM with Gaussian kernel version One-vs-One (SVM-rbf OvO), SVM with linear kernel version Onevs-Rest (SVM-l OvR), and SVM with Gaussian kernel version One-vs-Rest (SVM-rbf OvR). For all of them, the following manually selected features were extracted: mean, median, standard deviation, maximum, minimum, skewness, kurtosis, sum of real coefficients of Fast Fourier Transform, and maximum of real coefficients of Fast Fourier Transform. The evaluation times and accuracies of these models in comparison to a particular deep learning-based model (2DCNN per sensor + bLSTM, abbreviated as DNN in the graph) are presented in fig. 2.18. For the traditional methods, these times include the manual feature extraction process. In general, the deep learning models required about 0.15s to evaluate 365 samples, while this process took from 0.4 to even 0.8s with the traditional methods. In addition to this, the best performing deep learning model had the highest accuracy (98%), whereas traditional models showed accuracies between 40% and 90%. Deep learning models, thus, perform better on both evaluation time and accuracy.

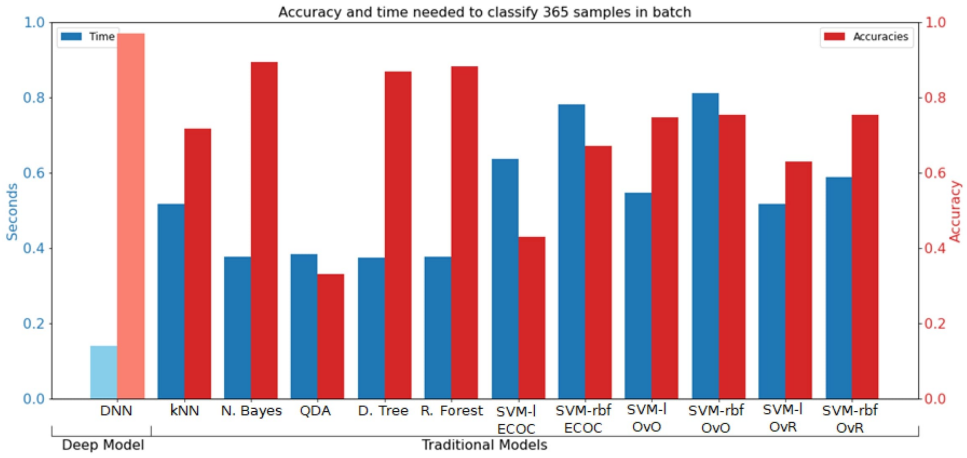


Figure 2.18: Prediction accuracy and evaluation time for traditional models in comparison to a deep learning-based model (DNN). The blue bars (left vertical axis) represent evaluation time and the red bars (right vertical axis) the prediction accuracy. The names of the models can be found in the main text.

2.4.2 Complete pipeline results

The complete activity recognition pipeline was presented in fig. 2.13, where a sliding window approach is used to first classify each window of the recording, then post-process those predictions, and, finally, remove the outliers. Here, some examples of results obtained

with the full pipeline are shown. These examples correspond to recordings that were not seen previously by the models, neither during the training nor the validation phase. The figures in this section are composed of four graphs. The one on the top is called the predictions and has the predictions obtained by the model (outputs of fig. 2.17). The second one is called the post-processed predictions and is the result of post-processing the predictions. The third graph is called the final predictions and contains the final predictions after the post-processed predictions are passed through the outlier removal process (outputs of fig. 2.17). Finally, the bottom graph is for reference and contains only three of all the sensor signals of the original recording. The horizontal axis (time) is shared among the four graphs. For the top three graphs, the vertical axis corresponds to the predictions made by the model and the orange-black color code represents the confidence of the predictions, where orange means low confidence and black high.

Fig. 2.19-2.21 present results of the complete pipeline on three different recordings. The true labels of the recordings can be seen in their respective captions.

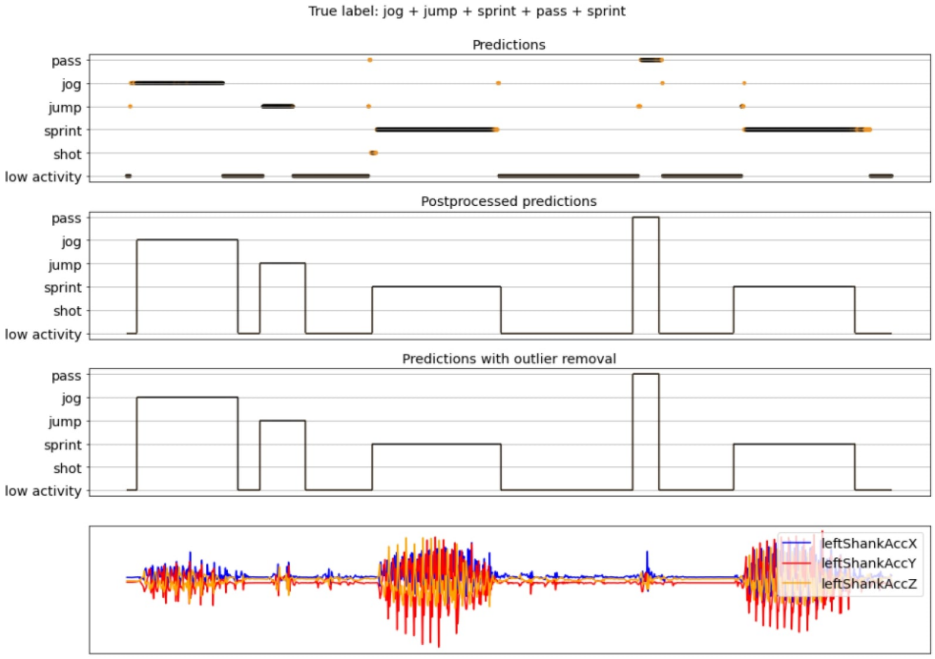


Figure 2.19: Example 1 of final results. True labels (in order): jog, jump, sprint, pass, and sprint with low activity periods in between each one of them. For the activity prediction plot, the orange-black color code of the dots and lines in the predictions represents the confidence of the predictions, where orange means low confidence and black high.

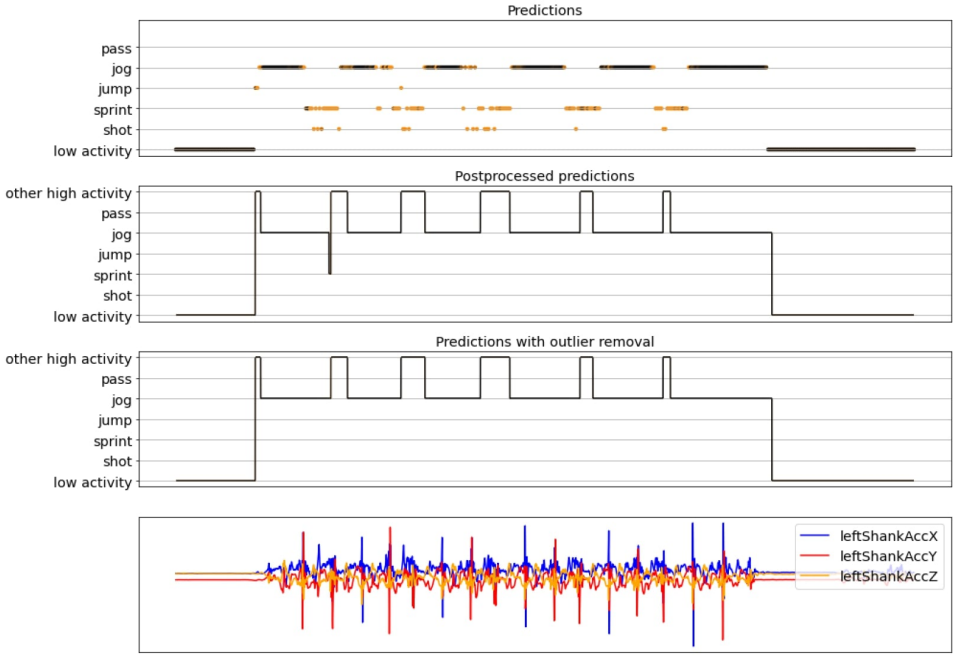


Figure 2.20: Example 2 of final results. True labels: jog and turn 5 times with a final jog.

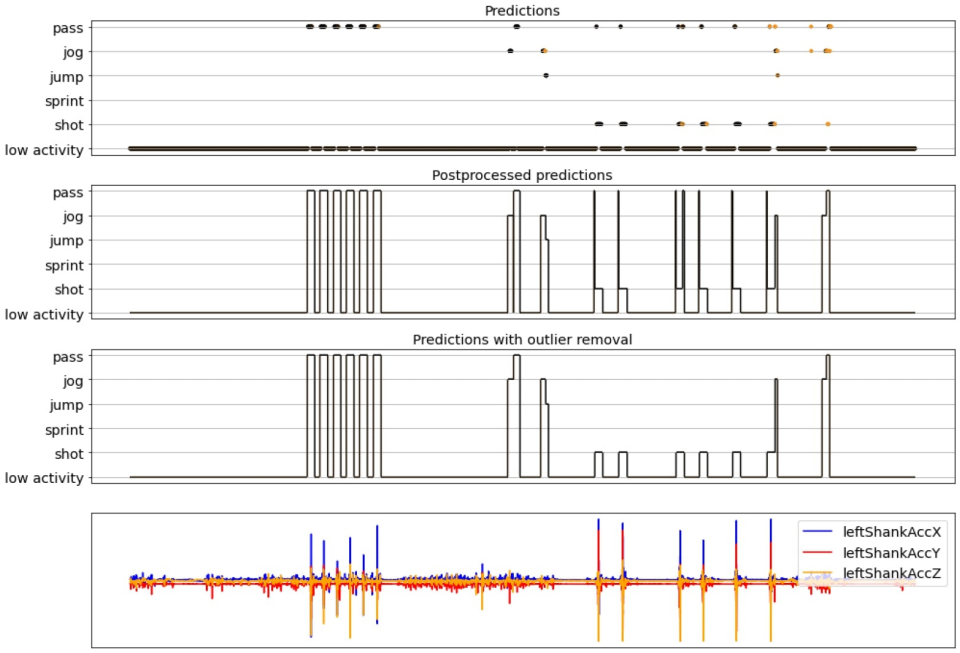


Figure 2.21: Example 3 of final results. True labels: 6 jumps-passes followed by 6 shots. Fast, unplanned movements are identified before and after the shots.

2.5 Discussion

This chapter explores the usage of deep learning models to accurately and rapidly recognize football activities based on IMU measurements from different body parts. The literature review shows that, for Human Activity Recognition, these types of techniques have been taking over the use of traditional machine learning algorithms, such as kNN, decision trees, and SVMs, where a manual process of feature extraction is required. The majority of reviewed articles focuses on deep learning models to recognize daily human activities, but this study includes explosive and repetitive activities typical of football practice. The possibility of building deep learning-based models on raw, not pre-processed IMU signals is one of the main goals.

A robust and end-to-end pipeline is proposed. It includes activity detection and isolation in order to prepare the required datasets, train the models, evaluate a recording via a sliding window approach, and post-process the results to obtain the final ones. Although the built deep models are trained to recognize the five most common football activities (sprints, passes, shots, jumps, and shots), the proposed methodology can be used to train models to consider additional activities, provided that there are enough training samples of those movements. Currently considered activities can also be split into sub-activities, if possible, e.g. sprints consist in multiple strides.

A dynamic activity detection algorithm is proposed to isolate activities in recordings from low activity periods. Internally, this algorithm uses a simple classifier that distinguishes between periodic (sprints and jogs) and explosive (shots, passes, and jumps) activities. Then, several deep learning architectures are built, trained, and evaluated in terms of prediction accuracy, overfitting, and evaluation time. Those architectures are based on novel variations of convolutional layers acting across signals, sensors, and/or a combination of them with and without weight sharing. Recurrent layers (LSTMs and bidirectional LSTMs) implemented after the convolutional layers are used to further give temporal meaning to the features previously extracted by the CNNs. The proposed models obtain high accuracies (up to 98.25% in the test set). Compared to traditional machine learning algorithms, deep learning models achieve better accuracies and faster evaluation times, showing that their use is recommended for HAR tasks. The combination of CNNs and bLSTMs is beneficial and achieves better results than the use of only CNNs. Furthermore, models with only bLSTMs do not generate good results, implying that convolutional operations are critical to extract relevant features. Further experimentation is encouraged regarding the training of the models, not only with respect to proposed architectures but also around fine-tuning of architecture and training variables, such as number of layers, number of convolutional filters, type of convolutional operations, learning rate, optimization algorithm, among others.

A sliding window approach for the evaluation phase is implemented following recommendations and best practices found in the literature. With this approach, a complete recording can be evaluated by the models. Post-processing of the predictions made by the model was necessary, thus the best-score post-processing method is proposed. This algorithm acts analogous to the non-max suppression algorithm for object detection on computer vision applications. It not only aligns the predictions to the original recording but also filters out short-lived, undesired activities. An outlier removal process is implemented at the end of the pipeline to further refine the final recognized activities from short-lived,

unnatural predictions.

Even if the obtained accuracies are high, it is recommended to acquire more data and retrain the models, especially with measurements taken from real matches. The more relevant data that a model can be trained with, the better the learning will be. The literature review shows that HAR is an active research area in the field of deep learning, with several open-source datasets present online ([34–36], to name a few). Although the majority of them are about daily human activities, such as walking, sitting, or standing, those large datasets can be used to build a base model, and then use transfer learning to fine-tune it with football data and allow it to recognize specific football activities. The use of transfer learning in deep learning applications has shown to be very effective when building models for tasks that do not have large training datasets and it is expensive or time-consuming to build them. Hence, it is highly recommended to explore this approach to potentially improve the proposed models.

Finally, it is encouraged to use this work as input to future research topics and use-case scenarios such as injury prevention, tracking of activity statistics during competition and training, monitoring physical load, and personalized training. This work is focused on football, but the proposed methodology and pipeline can also be applied to other sports. Recognizing the activities that a player does while playing a sport is just one of the steps that must be done in order to analyze their movements. The recognized activities can be combined with additional data sources such as video recordings and biomechanical analysis to study the prevalence and early detection of injuries. This would help obtain a better understanding of the players' performance and development.

3

Improving state estimation through projection post-processing for activity recognition with application to football

3.1 Introduction

In almost all areas of science and technology, sensors are becoming more prevalent. In recent years we have seen applications of sensor technology in fields as diverse as energy saving in smart home environments [38], performance assessment in archery [39], detection of mooring ships [40], early detection of Alzheimer disease [41] and recognition of emotional states [42], to name just a few.

Our main interest lies in the detection of human activities using sensors attached to the body. Sensors generate unannotated raw data, suggesting the use of unsupervised learning methods. If an activity specified in advance is of interest, then supervised learning and labelled data are required. However, the task of labelling activities manually from sensor data is labour-intensive and prone to errors, which creates the need for fast and accurate automated methods.

Human activity recognition (HAR) attracted much attention since its inception in the '90s. A plethora of methods are currently being used to detect human activities [43], with various deep learning techniques leading the charge [44, 45]. In many studies [46–48] only sensors embedded in a smartphone are used to classify user activities. Physical sensors, such as accelerometers or gyroscopes attached directly to a body or video recordings (from a camera), are the most popular sources of data for activity recognition [49–51]. Similarly,

This chapter is based on *M. Ciszewski, J. Söhl, G. Jongbloed. Improving state estimation through projection post-processing for activity recognition with application to football, Stat Methods Appl, 2023* [37].

cameras can be either placed on the subject [52–54] or they can observe the subject [55–57]. Rarely, both camera and inertial sensor data are captured at the same time [58].

The temporal structure of the time series should be taken into account when choosing a method for activity recognition. Simple classification techniques (such as logistic regression or decision trees) ignore time dependencies and will need to be improved after the procedure. Alternatively, methods which are more complicated and more difficult to train have to be deployed. Another challenge lies in the reliability of manual labelling (in case of supervised learning). Quite often it is unreasonable to assume that labels annotating the observed data are exact with regards to timings of transitions from one activity to another [59]. Timing uncertainty can be caused by a deficiency of the manual labelling or the inability to objectively detect boundaries between different activities. This issue is well-known in the literature, for instance, [60] introduced a scalable, parameter-free and domain-agnostic algorithm that deals with this problem in the case of one-dimensional time series.

The main contribution of this paper is the introduction of a post-processing procedure, which improves a result of activity classification by eliminating too short activities. The method requires a single parameter which can be interpreted as the minimum duration of the activites (hence the choice of this parameter is driven by domain knowledge). It allows us to mitigate the problem of activites being fragmented in cases where some domain-specific information about state durations is available. In the current literature, ad hoc techniques are employed for post-processing of human activities and they are particularly suitable when the initial classifier is already performing satisfactory. A method that exemplifies this approach, utilizing majority voting, can be found in the article by [61]. Some more advanced approaches have also been devised in special cases, e.g. the approach proposed by [62], which is limited to neural network classifiers. In comparison to any existing methods, our post-processing procedure ensures removal of all too short events and allows to specify the minimum length of activities accepted in the post-processed result. Based on empirical evidence, the performance of classical machine learning classifiers improves significantly by our method. This enables simple and fast but less accurate classification methods to be upgraded to accurate and fast classifiers.

In order to compare the quality of competing activity recognition methods, an appropriate criterion for evaluating the performance is needed (also to demonstrate the performance of the post-processing procedure we introduce). Below are some commonly used performance measures:

- accuracy, precision, the F-measure [43, 63],
- similarity measures for time series classification [64], such as Dynamic Time Warping or Minimum Jump Costs Dissimilarity,
- custom vector-valued performance metric [65].

Our objective is to design a performance measure that satisfies problem-specific conditions, which will be specified later.

The outline of the paper is as follows. Section 3.2 provides a method for improving classification with a post-processing scheme that uses background knowledge on the specific context. In particular, it validates the state durations and provides an improved classification that satisfies the physical constraints on the state durations imposed by the

context. Section 3.3 introduces specialized performance measures for assessing the quality of classification in general and in activity recognition in particular. The new performance measure also serves the purpose of showing the advantages of the post-processing fairly. Section 3.4 presents an application of the techniques in a simulated setting. The post-processing method was able to improve the estimates significantly. The method achieves similar results in an application to football data.

3.2 Improving classification by imposing physical restrictions

3.2.1 Post-processing by projection

When recognizing human activities, it is often the case that the result of the classification contains *events* (time intervals in which a classification result is constant) that are too short[1]. Usually ad hoc methods are used in order to discard those events, e.g. removal of any short events and replacing them with the next state in the classification, whose length is above a fixed threshold. There are also more advanced approaches, such as the one proposed by [62]. However, this particular method is suitable only when using a neural network as the classifier of choice, it does not ensure that too short events will always be eliminated (no matter what is exactly meant by 'too short') and lastly does not provide an intuitive understanding of the choice of its tuning parameter. Hence, our interest in a more formal method that could be used in combination with any activity classifier. The goal of this section is to introduce a formalized approach to correcting for the classifier's mistakes regarding the activity durations by introducing a novel post-processing procedure.

Consider the set of states $\mathcal{S} = \{1, ..., M\}$ and a metric d on $\mathcal{S}$. Let ρ denote the *discrete metric*[2] on $\mathcal{S}$. Any state-valued function of time will be called a *state sequence*. In reality we are only able to obtain a discrete-time signal, however, the relevant information contained in such a signal is a list of all the state transitions, which can more easily be encoded in a function with continuous argument. Hence, we define $\mathcal{T}$, the set of all càdlàg[3] functions $f : \mathbb{R} \to \mathcal{S}$ with a finite number of discontinuities. We define the *standard distance* induced by a metric d between two state sequences as

$$\text{dist} : \mathcal{T} \times \mathcal{T} \ni (f, g) \to \text{dist}(f, g) = \int_{\mathbb{R}} d(f(t), g(t))dt. \tag{3.1}$$

If d is a metric on $\mathcal{S}$, then dist is a metric on $\mathcal{T}$. The standard distance induced by the discrete metric is the time spent by f in a state different from g.

Now, we define a measure of closeness between functions in $\mathcal{T}$, as our goal is to find a function close enough to a given function in $\mathcal{T}$, while reducing the number of jumps it has (which in turn will eliminate short events in the state sequence). Let $f, g \in \mathcal{T}$. Then we introduce the notation:

$$E_\gamma(f, g) = \text{dist}(f, g) + \gamma \cdot |J(g)|, \tag{3.2}$$

[1] Depending on the application 'too short' might be specified differently.
[2] Distance between two different states is equal 1 and distance from a state to itself is equal 0.
[3] right continuous, left limits exist

where $J(g)$ is the set of all discontinuities of g, $|J(g)|$ is the number of all discontinuities of g and γ is a penalty for a single jump of g.

Given $f \in \mathcal{T}$, our goal is to find any solution $\hat{f} \in \mathcal{T}$ of the minimization problem

$$\hat{f} \in \arg\min_{g \in \mathcal{T}} E_\gamma(f, g). \tag{3.3}$$

As a default, we will use the standard distance induced by the discrete metric.

In order to characterize the solution $\hat{f}$ of problem (3.3) we present the following lemma.

Lemma 3.2.1. *Let $\gamma > 0$ and $f \in \mathcal{T}$. Let J denote the set of all discontinuities of the function f. There exists a solution $\hat{f}$ of the problem* (3.3) *such that it does not contain jumps outside of J.*

Lemma 3.2.1 leads to the conclusion that in search for the solution of the minimization problem we can limit ourselves to a finite set of functions, namely a subset of $\mathcal{T}$ with jumps only allowed at the same locations as the function f. The proof of lemma 3.2.1 can be found in the appendix.

In this minimization problem the choice of the parameter γ plays a crucial role. We will now show an interpretation of the penalty parameter that will ease the process of choosing it. It will also allow us to reformulate problem (3.3). First, we define a new set of functions.

Definition 3.2.1 (Function with bounded minimum duration of states)**.** Given a parameter $\gamma > 0$ we define $\mathcal{G}_\gamma \subset \mathcal{T}$, the set of functions with *bounded minimum duration of states*, such that for $g \in \mathcal{G}_\gamma$ we have

- $g = \sum_{i=1}^{n-1} s_i \mathbb{1}_{[t_i, t_{i+1})}$ for some constant $n \in \mathbb{N}$, a sequence of states $\{s_1, ..., s_{n-1}\}$, such that $s_i \neq s_{i+1}$ for $i = 1, ..., n-2$, and an increasing sequence $t_1 < t_2 < ... < t_n$ (we allow $t_1 = -\infty$ and $t_n = \infty$),
- if $n \geq 2$, then $\forall_{i \geq 2} \quad t_i - t_{i-1} \geq \gamma$.

Lemma 3.2.2 below yields a connection between the penalty γ and the minimum duration of states that we impose on the solution of our minimization problem.

Lemma 3.2.2. *Let $\gamma > 0$ and $f \in \mathcal{T}$. Any solution $\hat{f}$ of problem* (3.3) *is an element of $\mathcal{G}_\gamma$.*

This lemma can be used in practice to select the size of the penalty. The proof of lemma 3.2.2 can be found in the appendix.

Given $f \in \mathcal{T}$, by lemma 3.2.2 the minimization problem (3.3) is equivalent to the minimization problem

$$\hat{f} \in \arg\min_{g \in \mathcal{G}_\gamma} E_\gamma(f, g). \tag{3.4}$$

$\hat{f}$ will be called a projection of f onto $\mathcal{G}_\gamma$.

As mentioned before, the regularization by penalizing high numbers of jumps narrows down the set of possible solutions to a finite nonempty subset of $\mathcal{G}_\gamma$ (thanks to lemma 3.2.1),

which leads to the existence of $\hat{f}$. However, the solution might not be unique, as illustrated by the following example.

Consider $\mathcal{S} = \{0,1\}$, $f = \mathbb{1}_{[0.35,0.45)} + \mathbb{1}_{[0.55,+\infty)}$ and $\gamma = 0.2$. Both $\hat{f}_1 = \mathbb{1}_{[0.35,+\infty)}$ as well as $\hat{f}_2 = \mathbb{1}_{[0.55,+\infty)}$ are projections of f. One could think of it as an issue, however, it reflects well our understanding of the original problem. The assumption is that f has impossibly short windows, because it is uncertain which activity is actually performed in the interval $[0.35, 0.55)$. Looking only at f we are unable to decide which solution is more suitable, hence it is only natural that the method also returns two possible options.

We close with a remark regarding influence of the extreme values of γ on projection $\hat{f}$.

Remark 3.2.1. Let $f \in \mathcal{T}$. If $\gamma = 0$, then $\hat{f} = f$ is the only projection of f. If $\gamma = \infty$ and $E_\gamma(f,g) < \infty$ for some function $g \in \mathcal{T}$[4], then g is constant and equal everywhere to the most common state of f and $\hat{f} = f$[5].

3.2.2 Connection with the shortest path problem

In this section we devise a method for finding a projection in an efficient manner. It will be shown that the problem of finding the shortest path in a particular graph is equivalent to the minimization problem (3.4). This is possible thanks to the lemmas 3.2.1 and 3.2.2, which narrowed down the set of possible solutions to a finite set.

First, we present a lemma which further characterizes a projection of f.

Lemma 3.2.3. *Let $f \in \mathcal{T}$. Suppose $f \equiv c$ on an interval $[a,b]$ for some constant $c \in \mathbb{R}$. If $b-a > 2\gamma$, then $\hat{f} \equiv c$ on $[a,b]$. If $b-a = 2\gamma$, then there exists a projection such that $\hat{f} \equiv c$ on $[a,b]$.*

The proof of lemma 3.2.3 can be found in the appendix.

Remark 3.2.2. If $n > 2$, then there exists a projection such that the second and the second-to-last jump locations of the original function are not the first and the last (resp.) jump locations of this projection.

Remark 3.2.2 will be used when defining a particular graph and the proof can be found in the appendix.

We will assume that f has $n \geq 2$ jumps[6] at time points t_i for $i = 1, ..., n$:

$$f = \sum_{i=0}^{n} s_i \mathbb{1}_{[t_i, t_{i+1})}, \tag{3.5}$$

where $s_i \in \mathcal{S}$ for $i = 0, ..., n$ and $s_i \neq s_{i+1}$ for $i = 0, ..., n-1$. We use the following notation: $t_0 = -\infty$, $t_{n+1} = \infty$. In light of lemma 3.2.3 we assume that

$$t_{i+1} - t_i < 2\gamma \tag{3.6}$$

[4]Note that this is not always true. If the first and the last states of f are different, then any function can be a projection of f.

[5]Note that if $E_\gamma(f,g) < \infty$, then the first and the last states of f are the same and the constant function equal to that state is the only projection

[6]If $n = 0$ or $n = 1$, then $f \in \mathcal{G}_\gamma$ and $\hat{f} = f$.

for $i = 1,...,n-1$. If this is not the case, then consider the coarsest partition of the set J of jumps of f:

$$J = \bigcup_{i=1}^{r} J_i$$

such that for jumps in J_i for $i = 1,...,r$ equation (3.6) holds and $\min J_i - \max J_{i-1} \geq 2\gamma$ for $i = 2,...,r$. For each J_i for $i = 1,...,r$ consider a function $f_i : \mathbb{R} \to S$, such that $f_i \equiv f$ on $[\min J_i - 2\gamma, \max J_i + 2\gamma]$ and the only jumps of f_i lie in J_i. Once a projection $\hat{f}_i$ is found for f_i for all $i = 1,...,r$, we can then consider a function $\hat{f}$, defined as follows

$$\hat{f}(x) = \hat{f}_i(x) \tag{3.7}$$

given $x \in [\min J_i - 2\gamma, \max J_i + 2\gamma]$ for some $i = 1,...,r$. By lemma 3.2.3, there exists a projection which does not change the states longer than or equal 2γ, hence $\hat{f}$ defined as in (3.7) is a projection of f. Given this remark, we can now assume that f is of the form (3.5) and satisfies (3.6).

We will now define a graph for the purpose of showing the connection between the problem of finding a projection $\hat{f}$ and the problem of finding a shortest path in a directed graph. Let $G = (V, A)$ be a directed graph such that the set of vertices V is given by

$$V = \{t_0, t_1, ..., t_n, t_{n+1}\} \tag{3.8}$$

and the set of directed arcs is given by

$$A = \{(t_k, t_l) \in V^2 : t_l - t_k \geq \gamma\} \backslash \{(t_0, t_2), (t_{n-1}, t_{n+1})\}.^7 \tag{3.9}$$

There is a correspondence between each path from t_0 to t_{n+1} and a sequence of jumps in the interval $(t_1 - \gamma, t_n + \gamma)$. A path $(t_0, t_{l_1}, ...t_{l_m}, t_{n+1})$ can be associated with a function g with jumps at t_{l_1}, ..., t_{l_m}, such that $g(t_{l_k})$ is the most common value of f in interval $[t_{l_k}, t_{l_{k+1}})$. The definition (3.9) of the set of directed arcs ensures that all paths in the graph G correspond to at least one function in $\mathcal{G}_\gamma$.

We now introduce a weight function $W : A \to \mathbb{R}_+$ ensuring that the cost of the path coincides with the error $E(f, \cdot)$ of the corresponding function in the interval $(t_1 - \gamma, t_n + \gamma)$. Let $I_k = t_{k+1} - t_k$ for $k = 0, ..., n$. It is noteworthy that $I_0, I_n = \infty$, while $I_k < 2\gamma$ for $k = 1, ..., n-1$. We introduce the penalty for a jump $\phi_k = \gamma$ for $k = 1, ..., n$ and $\phi_{n+1} = 0$. Now we define the weight function W:

$$W((t_k, t_l)) = \sum_{m=k}^{l-1} I_m d(s_{kl}, s_m) + \phi_l, \tag{3.10}$$

for $(t_k, t_l) \in A$, where s_{kl} represents the most common state in the interval $[t_k, t_l)$ of the original function f. The first term equals the $\text{dist}(f, g)$ in $[t_k, t_l]$. The second term adds a penalty for jump at t_l if t_l is finite (the penalty for jump at t_k was added on a previous arc in the path, if $k > 0$).

[7] In case of $n = 2$, both arcs have to be included in set A.

Theorem 3.2.1 (Problem equivalence). *Let $\gamma > 0$ and $(t_1, ..., t_n)$ be the only discontinuities of a function $f \in \mathcal{T}$. Let $G = (V, A, W)$ be a weighted, directed graph as defined in* (3.8), (3.9), (3.10) *above. The task of finding a projection of f onto $\mathcal{G}_\gamma$, as defined in* (3.4), *is equivalent to finding the shortest path from t_0 to t_{n+1} in the graph G.*

The proof of the theorem can be found in the appendix. Now, we will illustrate the method by an example.

Given $\gamma = 0.2$ and $\mathcal{S} = \{0, 1, 2, 3\}$, consider the function $f = \mathbb{1}_{[0.2,0.35)} + 2 \cdot \mathbb{1}_{[0.4,0.55)} + 3 \cdot \mathbb{1}_{[0.55,0.75)} + 2 \cdot \mathbb{1}_{[0.75,+\infty)}$. The graph G, as defined in (3.8), (3.9), (3.10), for f, is shown in figure 3.1. Note that the vertex corresponding to 0.35 is omitted in the graph, since there is no path from the vertex corresponding to $-\infty$ to it (according to the definition (3.9), the arc $(0.2, 0.35)$ is not included).

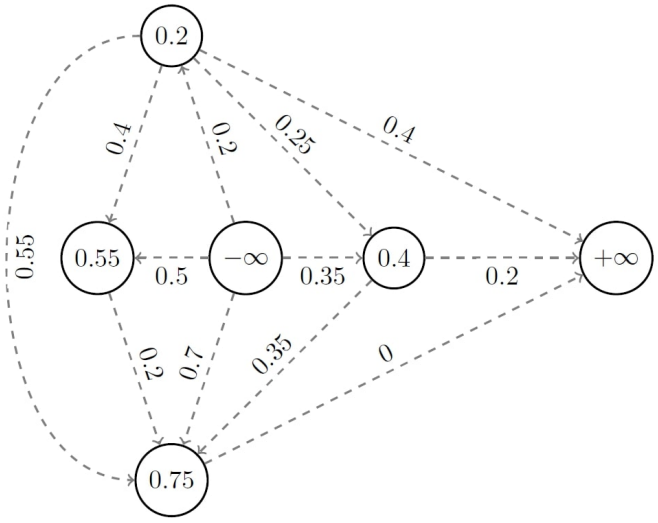


Figure 3.1: Graph G constructed for the function f.

There are nine possible paths from $-\infty$ to $+\infty$. The path $\hat{P} = (-\infty, 0.4, \infty)$ has the cost equal to 0.55 and is the shortest path from $-\infty$ to $+\infty$. Hence we conclude that $\hat{f} = 2 \cdot \mathbb{1}_{[0.4,\infty)}$ is the projection of f onto $\mathcal{G}_{0.2}$ (in this case, it can be shown $\hat{f}$ is the only projection of f).

3.2.3 Binary case

In case the set of states $\mathcal{S}$ consists of only two elements, a stronger result than lemma 3.2.2 can be achieved. The main advantage of the binary case comes from the fact that we do not need to specify the sequence of states since knowing the starting state, each jump signifies a move to the only other available state. First, we present a supporting remark which further strengthens the relation between jumps of a function from $\mathcal{T}$ and its projection.

For the remainder of the section, we will always assume that $\mathcal{S} = \{0, 1\}$[8].

[8]This convention deviates from the notation established in section 3.2.1 as it is more natural to use 0 and 1 as states in the binary case.

Lemma 3.2.4. *Let $\gamma > 0$ and $f \in \mathcal{T}$. Let J denote the set of all discontinuities of the function f. If a function $g \in \mathcal{G}_\gamma$ contains a jump $j \in J(f)$, but in an opposite direction than in f, then g cannot be a projection of f onto $\mathcal{G}_\gamma$.*

Lemma 3.2.5. *Let $\gamma > 0$ and $f \in \mathcal{T}$. Any solution $\hat{f}$ of the problem (3.3) is an element of $\mathcal{G}_{2\gamma}$.*

The proofs of lemma 3.2.4 and lemma 3.2.5 can be found in the appendix. Lemma 3.2.5 leads to the equivalence of the problem (3.4) with the minimization problem:

$$\hat{f} \in \operatorname*{arg\,min}_{g \in \mathcal{G}_{2\gamma}} E_\gamma(f, g). \tag{3.11}$$

The strengthening of lemma 3.2.1 by restricting not only the locations of the jumps but also their directions is a favorable change as it narrows the set of possible solutions.

Lemma 3.2.6. *Let $f \in \mathcal{T}$. Suppose $f \equiv c$ on an interval $[a,b]$ for some constant $c \in \mathbb{R}$. If $b-a > \gamma$, then $\hat{f} \equiv c$ on $[a,b]$. If $b-a = \gamma$, then there exists a projection such that $\hat{f} \equiv c$ on $[a,b]$.*

Proofs of lemma 3.2.6 can be found in the appendix.

Lemma 3.2.6 potentially reduces the number of jumps that have to be considered in the post-processing. Moreover, lemma 3.2.4 reduces the number of arcs when building the graph making the process of finding the shortest path more effective.

Additionally, remark 3.2.2 can also be strengthened.

Remark 3.2.3. If $n > 2$ and all states are shorter than γ (except for the first and the last state), there exists a projection such that the second and the second-to-last jump of the original function are not present in it.

Remark 3.2.3 allows us to ignore the second and the penultimate jump of the projected function when searching for jump locations in the projection. The proof of this remark can be found in the appendix.

The directed graph G has a different set of vertices compared to (3.8):

$$V = \{t_0, t_1, ..., t_n, t_{n+1}\} \backslash \{t_2, t_{n-1}\}^9, \tag{3.12}$$

and of directed arcs compared to (3.9):

$$A = \{(t_k, t_l) \in V^2 \,:\, t_l - t_k \geq 2\gamma \text{ and } l - k \bmod 2 \equiv 1\}. \tag{3.13}$$

Theorem 3.2.2 (Problem equivalence - binary version)**.** *Let $\gamma > 0$ and $(t_1, ..., t_n)$ be the only discontinuities of a function $f \in \mathcal{T}$. Let $G = (V, A, W)$ be a weighted, directed graph as defined in (3.10), (3.12), (3.13). The task of finding a projection of f onto $\mathcal{G}_{2\gamma}$, as defined in (3.11), is equivalent to finding a shortest path from t_0 to t_{n+1} in the graph G.*

Proof of the theorem 3.2.2 can be found in the appendix.

[9] If n=2, then both of those jumps are present in V

3.3 Incorporating domain knowledge into the performance measure of classification

3.3.1 Problem-specific requirements on the performance measure

In order to choose an appropriate performance measure for a given classification task, it is important to understand the problem-specific demands on the result. The standard distance (3.1), which can be understood as a continuous analogue of the most common performance metric, namely the misclassification rate, can often be inadequate to compare the classification results as it is a one-fits-all type of metric and if more is known about the problem, it might not represent the idea of accuracy that users have in mind. On the other hand, there have been other approaches to performance metrics, e.g. [65]. Their approach focuses on characterizing the error in terms of the number of inserted, deleted, merged and fragmented events. Event fragmentation occurs when an event in the true labels[10] is represented by more than one event in the estimated labels[11], whereas merging refers to several events in true labels being represented by a single event in the estimated labels. [65] provide an overview of different performance metrics used in activity recognition proposing a solution to the problem of timing uncertainty as well as event fragmentation and merging. Their solution is based on segments, which are intervals in which neither the true labels nor the estimated labels change the state. If the state in the estimate and the state in the true labels agree in a given segment, they denote it as correctly classified. If that is not true, the segment is classified accordingly as fragmenting segment, inserted segment, deleted segment or merged segment. This provides a deeper level of error characterization, which is then used in different metrics of classifier performance. Their vector-valued performance metric is preferable when in-depth overview of the types of mistakes made by the classifier is needed. We will introduce a novel scalar-valued performance metric, which can be easily compared and includes problem-specific information such as timing uncertainty in the labels.

In this section, we aim at highlighting the main characteristics of the classification of movements based on wearable sensors and at translating them into specific requirements on the performance measure. Our first requirement comes from physical restrictions. The states considered in our application represent human activities, but also in more general contexts they often cannot be arbitrarily short; there is a lower bound on the length of the events in a state sequence. Hence, estimated labels that violate this lower bound indicate a bad performance. The lower bound condition requires two parameters: the lower bound and the penalty for each violation. The lower bound can either be estimated or determined from domain knowledge, while the penalty can be chosen more freely. Through physical restrictions we can see a deeper connection with the method introduced in section 3.2. It is clear that the standard classification methods cannot ensure that the state sequence contains only events longer than a certain level. The post-processing method addresses this issue directly and as a consequence we can expect classifiers to benefit from it in the context of the new performance measure.

[10] If a state sequence corresponds to the true underlying sequence of activities in a time series, then it will be called the *true labels*

[11] An estimate of the true labels will be called the *estimated labels*.

The issue of timing uncertainty should also be addressed when designing the performance measure. To illustrate its importance more clearly, we present an example. Five people were asked to detect boundaries between activities in different time series using a visualization tool. The tool outputs an animated stick figure model[12] given sensor data.

Three time series were selected, each with one of the following activities: running, jumping and ball kick. The start and the end of each activity were recorded by participants. Table 3.1 presents the results of the experiment.

Partic.	Running		Jumping		Kick	
	Start	End	Start	End	Start	End
P1	2	7.3	2.7	5.2	2.5	3.5
P2	2	7.5	2.7	5.2	2.5	3.9
P3	2.3	6.6	2.7	5.1	2.7	3.6
P4	2.3	7.2	2.7	5.3	2.5	4.3
P5	2.2	7.2	2.9	5.4	2.5	4.1
Avg.	2.16	7.16	2.74	5.24	2.54	3.88
Std	0.15	0.34	0.09	0.11	0.09	0.33

Table 3.1: The results of the labelling experiment; all times are in seconds. The two last rows show the average and the sample standard deviation for each boundary

The experiment indicates there is indeed uncertainty regarding the state transitions. Granted that the sample size is very small, we notice more variation in results referring to the end of activities rather than the beginnings. Additionally, we see more variation in the results for the kick than the jumping. So the boundaries of some activities seem to be more difficult to identify than of others.

3.3.2 Globally Time-Shifted distance

The standard distance (3.1) is an unsatisfying measure to compare two state sequences, since it does not incorporate the requirements posed in the previous section. In order to improve it, we start by modelling the timing uncertainty. Let $f \in \mathcal{T}$ be the true labels process and let f have n discontinuities $t_1, ..., t_n$. The locations of the discontinuities are corrupted by additive noise:

$$t_i = T_i + X_i,$$

for all $i = 1, ..., n$, where T_i is the true and unknown location of the i-th jump. In this section we will assume that $X_1 = X_2 = ... = X_n$ (all jumps are moved by the same value; the global time shift), although in general, it is more realistic to assume that $X_1, ..., X_n$ are independent random variables. We will relax this condition later.

We define a class of Globally Time-Shifted distances (GTS distances), loosely inspired by the Skorokhod distance on the space of càdlàg functions [66, pp. 121]. The GTS distances are parametrized by two parameters. The parameter w controls the weight of misclassification

[12] A symbolic representation of the human body using only lines

occurring from the uncertainty of the true labels, while the parameter σ controls the magnitude of the shift of activities.

Definition 3.3.1 (Globally Time-Shifted distance)**.** Let $f, g \in \mathcal{T}$. Given $w \geq 0, \sigma > 0$ and a metric d on S we define a Globally Time-Shifted distance as:

$$GTS_{w,\sigma}(f,g) = \inf_{\epsilon \in [-\sigma,\sigma]} \{\text{dist}(f \circ \tau_\epsilon, g) + w|\epsilon|\},$$

where for $\epsilon > 0$ $\tau_\epsilon : \mathbb{R} \to \mathbb{R}$ is a time shift defined as follows:

$$\tau_\epsilon(t) = t - \epsilon.$$

Depending on the choice of parameters the GTS distance possesses certain properties. For $w > 0$ and $\sigma = \infty$, the GTS distance is an extended metric[13] and a proof of this fact is given in the appendix. If $w > 0$ and $\sigma > 0$, then it is a semimetric meaning that it has all properties required for a metric, except for the triangle inequality.

The main downside of using the GTS distance is the unrealistic assumption on timing uncertainty. However, if we know that the true labels preserve the true state durations then it is a good choice. Consider a function $f \in \mathcal{T}$ with two state transitions located at t_1 and t_2. Let $g \in \mathcal{T}$ also feature two state transitions located at $t_1 - \tau_1$ and $t_2 - \tau_2$. If $\tau_1 \neq \tau_2$, then there is no global time shift that can align the functions f and g. This implies that the true state durations need to be preserved in the estimate in order to align functions using the global time shift.

3.3.3 Locally Time-Shifted distance and the Duration Penalty Term

The global time shift stresses the state durations, which is not always desirable. For instance, if the true labels do not preserve the real state durations, or e.g. if the additive noise terms in the locations of the jumps are independent. Here is an example: figure 3.2 shows f and its approximations g_i for $i = 1, 2, 3$. It is impossible to align f with any of the g_i with a single time shift, however, it would be possible if each state transition could be shifted 'locally'.

Naturally, to accommodate for this issue, a suitable modification would be to replace one global time shift with multiple local time shifts. We now introduce a measure of closeness between state sequeneces which conceptually can be seen as derived from the GTS measure. We will be working with sequences of jumps, but more specifically given two sequences of state boundaries we will combine them together and sort the resulting joint sequence in an increasing order. Subsequent pairs of values in this sequence are determining segments understood as in [65]. We weigh different types of segments and the result is a weighted average of segment lengths, which is supposed to reflect well the error magnitude of the classifier.

We define segments formally and introduce a new distance on $\mathcal{T}$.

Definition 3.3.2 (Segments)**.** Let $f, g \in \mathcal{T}$. The elements of the smallest partition[14] of $\mathbb{R}$ such that in each element of the partition neither f nor g changes state will be called segments.

[13] It may attain the value ∞.

[14] A partition that cannot be made coarser

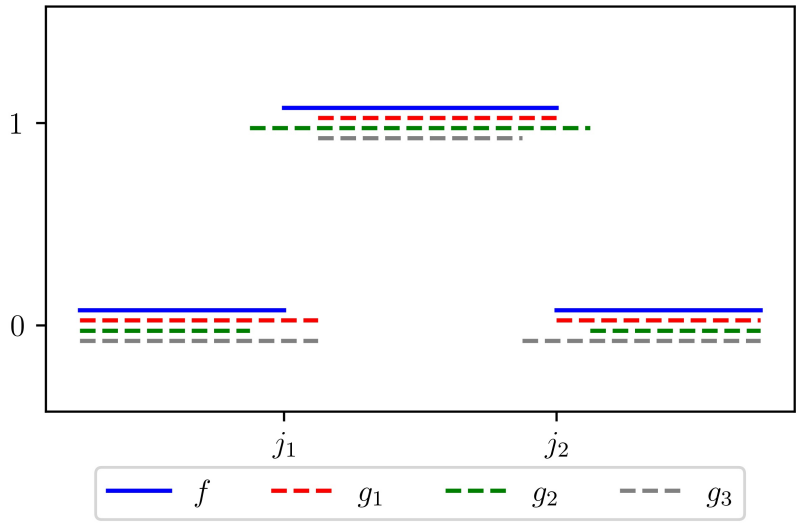


Figure 3.2: The function f represents the true labels with an uncertainty around state boundaries, g_i are the approximations of f

Since functions from $\mathcal{T}$ are piece-wise constant and have a finite number of discontinuities, there is always a finite number of segments. The general form of segments that we will use is as follows:

$$(-\infty, a_1) \cup \bigcup_{i=1}^{l-1} [a_i, a_{i+1}) \cup [a_l, \infty), \tag{3.14}$$

where $a_1 < a_2 ... < a_l$ if f and g are not both constant on the real line. Otherwise there is only one segment, consisting of the whole real line. By convention, $a_0 = -\infty$ and $a_{l+1} = \infty$, and

$$f(a_0) = f(a_1^-) = \lim_{x \to -\infty} f(x),\ f(a_{l+1}) = f(a_l).$$

Definition 3.3.3 (Locally Time-Shifted distance)**.** Let $w \geq 0$, $\sigma > 0$ and d be a metric on S. Let $f, g \in \mathcal{T}$ and their set of segments to be denoted as in (3.14). We define the Locally Time-Shifted distance (LTS distance) as

$$LTS_{w,\sigma}(f,g) = \sum_{i=1}^{l-1} \delta_i (a_{i+1} - a_i) d(f(a_i), g(a_i)),$$

where

$$\delta_i = \begin{cases} w, & a_{i+1} - a_i \leq \sigma,\ f(a_{i-1}) = g(a_{i-1}), f(a_{i+1}) = g(a_{i+1}) \\ 1, & \text{otherwise.} \end{cases}$$

Similarly to the GTS distance, the parameter w controls the weight of misclassification occurring from the uncertainty of the true labels. The case when $w < 1$ is more interesting

to us, since it corresponds to timing uncertainty of the labels. If $w \geq 1$, then we put more importance on the timings of the jumps (opposite to timing uncertainty). The LTS distance is an extended semimetric for $w > 0$ (for a proof, see the appendix). The triangle inequality does not hold in general.

The LTS distance addresses the issue of timing uncertainty in the true labels. Let $\zeta > 0$[15] be the lower bound on the lengths of the events as determined by the domain knowledge (or through estimation if possible). Let $\lambda > 0$ be the penalty for each violation of the lower bound condition. For $f \in \mathcal{T}$ with its discontinuities $t_1, ..., t_n$, we introduce a *duration penalty term*:

$$DP_{\lambda,\zeta}(f) = \lambda \sum_{k=1}^{n-1} \mathbb{1}_{[0,\zeta)}(t_{k+1} - t_k).$$

This term will allow to lower the performance of classifications with unrealistically short events.

In practice, we will need to extend the functions to the real line in order to use the LTS distance as it is defined for functions with domain equal to the whole of $\mathbb{R}$. One natural extension could be to extend the first and the last state of each function indefinitely. However, this solution leads to a problem. Let $M > 0$. Consider two functions $f : [0, M] \to \mathcal{S}$ and $g : [0, M] \to \mathcal{S}$ such that for some $0 < a < M$, $f(t) \neq g(t)$ on $[0, a)$. No matter how small a is, the distance between extended f and g will always be infinite when using this extension, since in this case extended f and g are in different states on the whole half line $(-\infty, a)$. Both functions need to be extended by the same state for the distance to be finite. We extend any function f defined on interval $[0, M]$ to the real line, setting its value to an arbitrary state outside of $[0, M)$. The distance is independent of the chosen state, as on the infinite segments that it introduces f and g are both equal. Without loss of generality, we choose state 1.

$$f^*(t) = \begin{cases} f(t), & t \in [0, M) \\ 1, & t \notin [0, M). \end{cases} \tag{3.15}$$

Notice that this extension does not have the problem stated above as f^* and g^* are equal on the segments that it introduces and does not change the value on the original segments regardless of the choice of the state outside of $[0, M]$.

We combine the LTS distance and the duration penalty term to define the LTS measure of closeness of two state sequences.

Definition 3.3.4. Let f be a function of true labels and g its estimate, both defined on $[0, M]$. The *LTS measure* is defined as:

$$LTS_{w,\sigma,\lambda,\zeta}(f, g) = \exp(-LTS_{w,\sigma}(f^*, g^*)/M - DP_{\lambda,\zeta}(g)).$$

The scaling through the division by M normalizes the LTS distance to the interval $[0, 1]$. The transformation $[0, +\infty) \ni x \to \exp(-x) \in (0, 1]$ maps the sum of the LTS distance and the duration penalty term to the interval $(0, 1]$, while reversing the order as well: g is closer to f if the LTS measure is closer to 1.

[15] Note that ζ is related in its interpretation to the γ parameter introduced in section 3.2.

3.4 APPLICATION TO ACTIVITY RECOGNITION

3.4.1 SIMULATION STUDY

We consider a dataset created using a random procedure, which mimics the behavior of activity recognition classifiers with varying accuracy (depending on the parameters). Let $\mathcal{S} = \{1, 2, 3\}$. Consider a function f representing a 60 second long state sequence:

$$f = \mathbb{1}_{[0,5)} + 2 \cdot \mathbb{1}_{[5,15)} + 3 \cdot \mathbb{1}_{[15,30)} + 2 \cdot \mathbb{1}_{[30,40)} + 3 \cdot \mathbb{1}_{[40,55)} + \mathbb{1}_{[55,60]}.$$

f will be referred to as the correct labels. We introduce noise into f in the following manner:

- two sequences of i.i.d. random variables are considered $\{Y_k\}$ and $\{Z_k\}$, with $Y_k \sim Exp(\mu_1)$ and $Z_k \sim Exp(\mu_2)$ for some parameters $\mu_1, \mu_2 > 0$,
- $\{Y_k\}$ represents the time spent in the correct state, while $\{Z_k\}$ represents the time spent in the incorrect state,
- we use the sequence $Y_1, Z_1, Y_2, Z_2, ...$ to generate noisy labels, where the sequence ends when the sum of all drawn numbers is exceeding 60 seconds,
- for each variable Z_i an incorrect state is chosen randomly out of the remaining two and f is changed to that state on interval $[\sum_{k=1}^{i-1}(Y_k + Z_k) + Y_i, \sum_{k=1}^{i}(Y_k + Z_k))$,
- μ_1 and μ_2 control the duration of the states.

As our performance measure we choose the LTS measure with parameters: $w = 0.6$, $\sigma = 0.35$, $\lambda = 0.0001$, $\zeta = 0.5$, $d = \rho$. The post-processing is performed for the noisy labels with parameter $\gamma = 0.5s$. To demonstrate the utility of the post-processing procedure, we draw the noisy function 1000 times for a given set of parameters (μ_1, μ_2) and compare the accuracy of the noisy labels, the accuracy of the post-processed labels, the LTS measure of the noisy labels and the LTS measure of the post-processed labels.

In the first setting, we fix $\mu_1 = 0.1s$. The procedure is repeated for $\mu_2 \in [0.01, 0.1]$ (100 sample points from the interval are chosen). Figure 3.3 compares the mean accuracy of the noisy labels and the post-processed labels as well as the mean LTS measure of the noisy labels and the post-processed labels.

In the second setting, we fix $\mu_1 = 0.5s$. The procedure is repeated for $\mu_2 \in [0.05, 0.5]$ (100 sample points from the interval are chosen). Figure 3.4 shows the mean accuracy of the noisy labels and the post-processed labels as well as the mean LTS measure of both the noisy labels and the post-processed labels.

In the third setting, we fix $\mu_1 = 1s$. The procedure is repeated for $\mu_2 \in [0.1, 1]$ (100 sample points from the interval are chosen). Figure 3.5 shows the mean accuracy of the noisy labels and the post-processed labels as well as the mean LTS measure of both the noisy labels and the post-processed labels.

All three experiments show the improvement in the accuracy as well as the LTS measure thanks to the use of post-processing. Additionally, we conclude that the post-processing method behaves better when dealing with multiple shorter intervals rather than fewer longer intervals. Moreover, the boxplots show more variability in the performance of

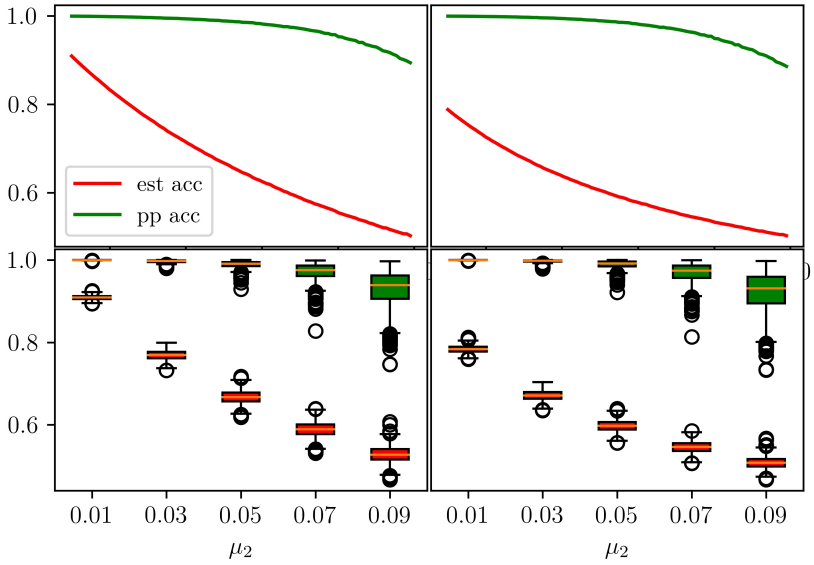


Figure 3.3: The top left plot shows the mean accuracy of the noisy labels (red) and the mean accuracy of the post-processed labels (green) as calculated for different values of μ_2. The top right plot shows the LTS measure of the noisy labels (red) and the post-processed labels (green) as calculated for different values of μ_2. All lines drawn for 100 different values of μ_2. The bottom left boxplot shows the variability of the accuracy amongst the estimates (red) and the post-processed estimates (green). The bottom right boxplot shows the variability of the LTS measure amongst the estimates (red) and the post-processed estimates (green). Boxplots have been constructed for 5 different values of μ_2.

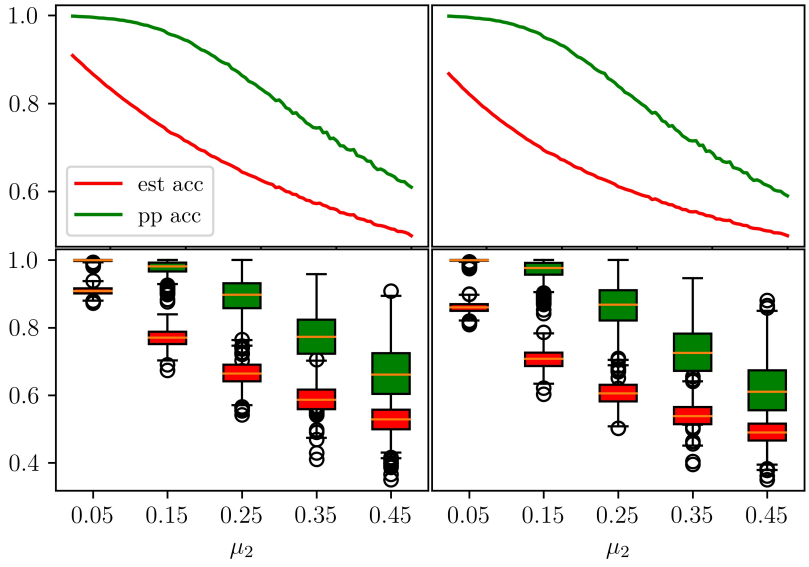


Figure 3.4: The top left plot shows the mean accuracy of the noisy labels (red) and the mean accuracy of the post-processed labels (green) as calculated for different values of μ_2. The top right plot shows the LTS measure of the noisy labels (red) and the post-processed labels (green) as calculated for different values of μ_2. All lines drawn for 100 different values of μ_2. The bottom left boxplot shows the variability of the accuracy amongst the estimates (red) and the post-processed estimates (green). The bottom right boxplot shows the variability of the LTS measure amongst the estimates (red) and the post-processed estimates (green). Boxplots have been constructed for 5 different values of μ_2.

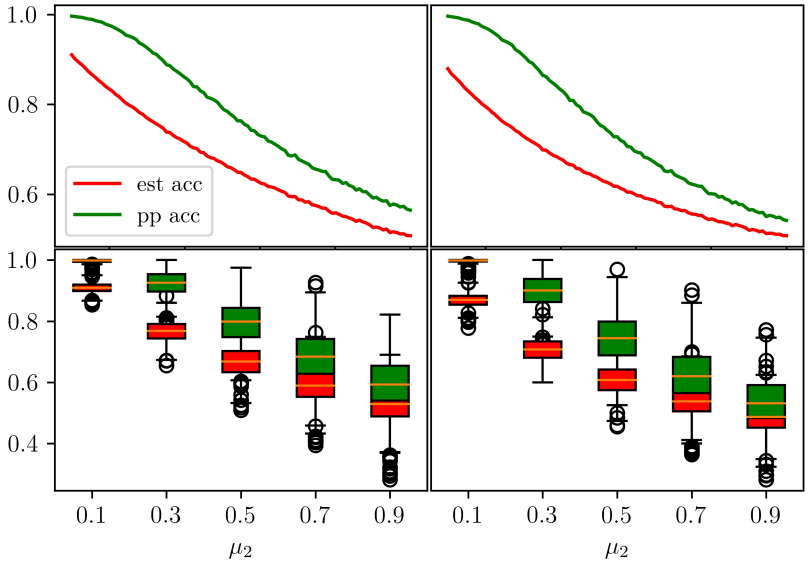


Figure 3.5: The top left plot shows the mean accuracy of the noisy labels (red) and the mean accuracy of the post-processed labels (green) as calculated for different values of μ_2. The top right plot shows the LTS measure of the noisy labels (red) and the post-processed labels (green) as calculated for different values of μ_2. All lines drawn for 100 different values of μ_2. The bottom left boxplot shows the variability of the accuracy amongst the estimates (red) and the post-processed estimates (green). The bottom right boxplot shows the variability of the LTS measure amongst the estimates (red) and the post-processed estimates (green). Boxplots have been constructed for 5 different values of μ_2.

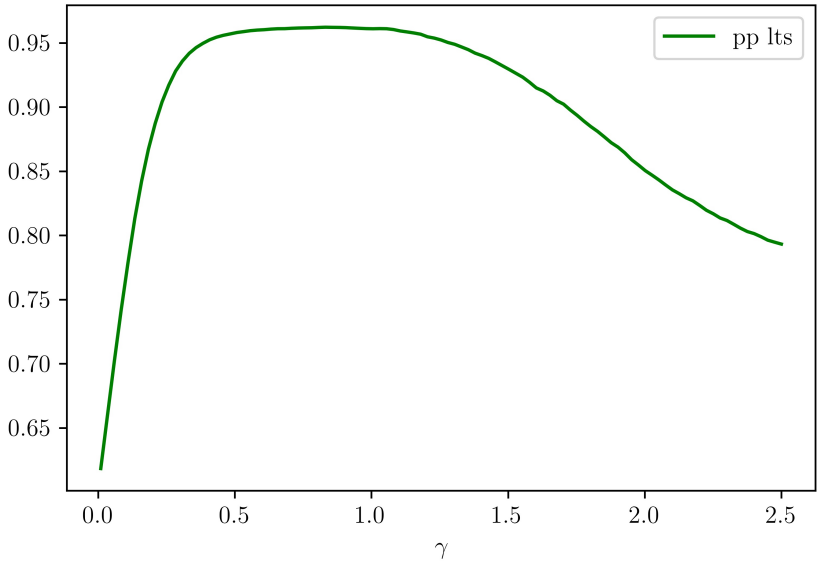


Figure 3.6: The line shows the LTS measure of the post-processed labels drawn for 100 different values of γ. The mean accuracy of noisy labels was equal to 0.556 and the mean LTS measure of noisy labels was equal to 0.602.

the post-processed estimates when dealing with initial estimates with fewer but longer intervals of misclassification. This can be due to the fact that at the level of around 0.5 in accuracy and in the LTS measure, the post-processing starts behaving much worse and is not able to recover the original signal as reliably. It shows the limits of the method and the fact that there is a point at which the method starts to behave worse.

We also investigate the importance of the parameter γ on the results. We fix $\mu_1 = 0.1$, $\mu_2 = 0.08$. The procedure is repeated for $\gamma \in [0.01, 2.5]$ (100 sample points from the interval are chosen). Figure 3.6 shows the mean LTS measure of the post-processed labels.

We conclude that the parameter γ can influence the LTS measure of the post-processed functions $\hat{g}_i$. It needs to be chosen carefully since too low values will lead to accepting unrealistically short events while too high values will eliminate true events. In our case the values of γ between 0.5 and 1 are the most favourable. In practice the minimal length of the events in the true labels can inform on the choice of γ.

We finish the simulation study with a look at the parameters of the LTS measure. We will investigate the weight w first. Let all the other parameters of the LTS measure be set to $\sigma = 0.35$, $\lambda = 0.0001$, $\zeta = 0.5$. We fix $\mu_1 = 0.1, \mu_2 = 0.08, \gamma = 0.5$. The procedure is repeated for 100 different values of w in the interval $[0, 2]$. Figure 3.7 shows the mean LTS measure of the post-processed labels.

Figure 3.7 shows the effect of the parameter w on the LTS measure. As we can see on the y-axis, the values of the LTS measure are quite close together. Hence, we conclude that the choice of w is less important as its effect on the LTS measure is minimal. The

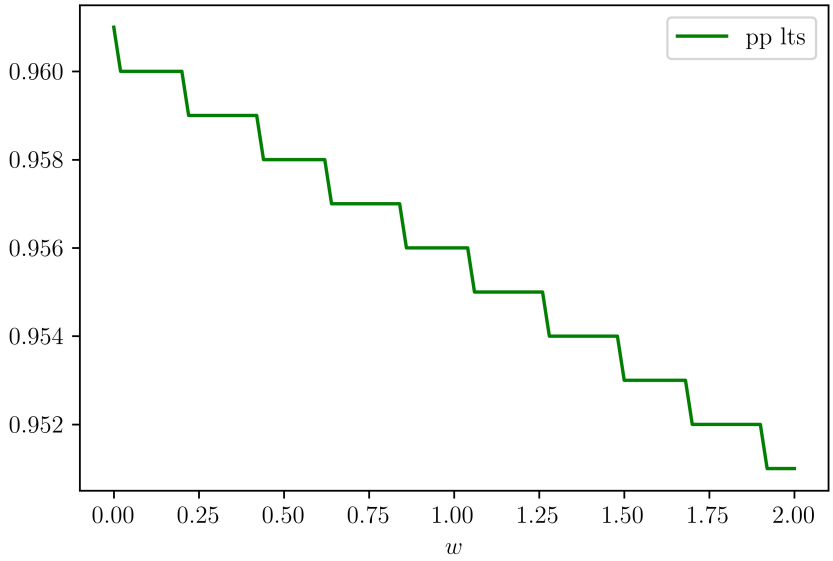


Figure 3.7: The line shows the LTS measure of the post-processed labels drawn for 100 different values of w. The mean accuracy of noisy labels was equal to 0.555.

main reason for this behaviour stems from the fact that σ restricts many of the erroneous intervals and the remaining ones for which w takes effect are quite small. Hence, the effect of w on the LTS measure is not large.

The parameter σ determines the length of the misclassified events up to which they are caused by timing uncertainty of the labels. σ can be chosen based on the domain knowledge, based on the experiment described in section 3.3.1. The parameter ζ is a lower bound on the lengths of the events, hence can be determined by the domain knowledge. Given their clear interpretation, the parameters σ and ζ will not be subjected to the same procedure as the parameter w. Hence, the only parameter left to investigate is λ. As before, we fix $\mu_1 = 1, \mu_2 = 0.8, \gamma = 0.5$. We choose $w = 0.6$. The procedure is repeated for 100 values of λ between 0 and 0.5. Figure 3.8 shows the mean LTS measure of the post-processed labels. We can see that high values of λ can influence the LTS measure significantly, hence choices lower than 0.01 are preferable. We want to avoid that the penalty term is overshadowing the LTS distance.

3.4.2 Application to a football dataset

We will now demonstrate the benefits of the post-processing by projection in a real-life setting, utilizing the LTS measure to compare different methods of classification. [29] give an extensive description of the football dataset of which we give a short summary below.

Eleven amateur football players participated in a coordinated experiment at a training facility of the Royal Dutch Football Association of The Netherlands. Five Inertial Measure-

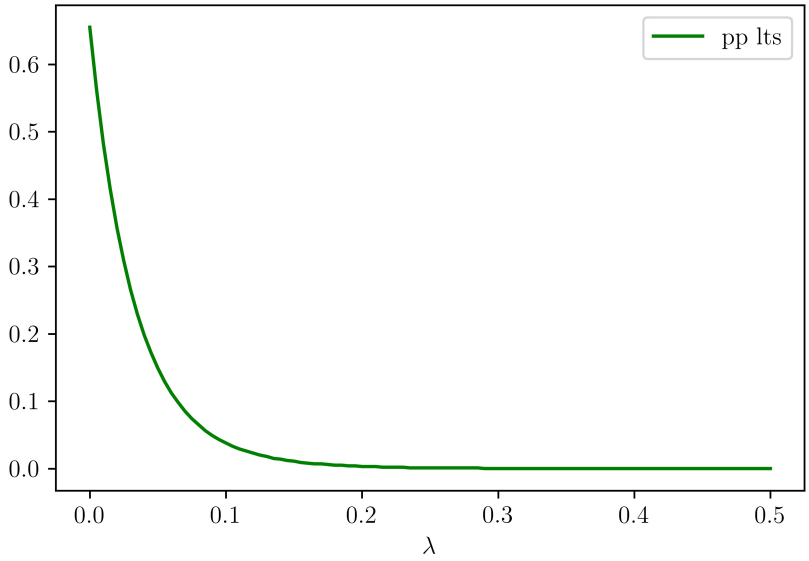


Figure 3.8: The line shows the LTS measure of the post-processed labels drawn for different values of λ. The mean accuracy of noisy labels was equal to 0.56.

ment Units (IMUs) were attached to 5 different body parts: left shank (LS), right shank (RS), left thigh (LT), right thigh (RT) and pelvis (P). Each IMU sensor contains a 3-axis accelerometer (Acc) and a 3-axis gyroscope (Gyro). Athletes were asked to perform exercises on command, e.g. 'jog for 10 meters' or 'long pass'. For each athlete and exercise this resulted in a 30-dimensional time series (5 body parts times 6 features per IMU) of length varying from 4 to 14 seconds. Each athlete performed 70-100 exercises which amounts to nearly 900 time series (each with a sampling frequency of 500 Hz). Time series are labelled with the command given to an athlete, but there are still other activities performed in each of the time series, for example standing still. This causes a problem; ignoring standing periods and treating them as part of the main signal pollutes the data and lowers the quality of the classification. To show the advantages of post-processing by projection, we select only two states: 'standing' and 'other activity' encoded as 0 and 1, respectively. 15 time series (representative of all possible actions performed by athletes) were manually labelled time point by time point in order to be able to train classifiers, and these will form our sample. All 15 time series were chosen from the single athlete.

In pre-processing we are using the sliding window technique on the sensors [67]. This method transforms the original raw data using windows of fixed length d and a statistic of choice T: given a time point t, its neighbourhood of size d is fed to the statistic T for each variable separately. Performing the procedure for each time point results in a time series of the same dimension as the original one, but every observation is equipped with some knowledge about the past and the future through the statistic T and through forming the

neighbourhoods of size d. Regarding the choice of the statistic T one needs to be careful, since the sensors are highly correlated with each other. The information about standing contained in one variable is comparable to the one in another, namely the variance of the signal is low when the person is standing (differences can occur when considering different legs; a low variance on one leg might be misleading since the other leg might already be transitioning into another position).

Leave-5-out cross-validation will be performed in order to select the best performing classification method out of the 7 standard machine learning methods, which will be listed below. A typical approach to k-fold cross-validation with a training sample of size $k-1$ cannot be applied here, since a single time series is not a representative sample of different types of events. 15 time series will be used. In each iteration 10 time series will be randomly chosen for training and 5 for testing. The results are going to be shown for post-processed classifiers, unless specified otherwise. Before cross-validation can be performed, we need to fix the parameters of the performance measure we introduced in section 2. The parameters of the LTS measure are chosen as follows:

- We have limited information regarding how uncertain locations of state transitions are, but based on the small experiment described in section 3.3.1 we select $\sigma = 0.35$ (the largest deviation between different true labels).
- The parameter w is chosen as 0.6, but as shown in section 3.4.1 its choice is not that important.
- The lower bound γ on the duration of activities is selected as the length of the shortest activity in the learning dataset, which is equal to 0.8s in our case.
- A penalty λ represents the cost of additional or missing jumps in a state sequence compared to the true labels. We decide for the penalty $\lambda = 0.01$ in order not to overshadow the LTS distance with too much importance placed on the penalty term (more details on that were given in section 3.4.1, specifically in figure 3.8).

Before assessing classifiers on the training set, one needs to consider an appropriate feature set. Our variables are highly dependent on one another, so we start with feature selection. We perform feature ranking using the Relieff algorithm and select the 6 most relevant features based on the Relieff weights (more details on the method in [68]). Then we test all possible combinations of these features, which is now computationally feasible, in order to find the best set for each of the classifiers. The features selected by the Relieff algorithm are RTGyroX, RTGyroY, RTAccX, RTAccZ, LTAccY, PAccY, where the naming convention is as follows: RTGyroX refers to the x-axis of the gyroscope located on the right thigh and so forth.

Proceeding with the cross-validation we select the following classifiers (with their abbreviations) to be assessed: DT - Decision Tree, kNN - k-Nearest Neighbors, LR - Logistic Regression, MLP - Multi-layer Perceptron, NB - Naive Bayes, RF - Random Forest, SVM - Support Vector Machine. The results of leave-5-out cross-validation are shown in table 3.2. It is striking that the test scores of the post-processed classifiers are at most 0.028 apart. This is due to post-processing by projection. The correction it provides brings all classifiers closer together. This result can be extended even further. The test score of a decision tree

Classifier	OG Test	PP Test
MLP	0.916+/-0.031	0.972+/-0.008
LR	0.898+/-0.034	0.968+/-0.015
kNN	0.59+/-0.05	0.967+/-0.020
RF	0.83+/-0.07	0.966+/-0.017
SVC	0.894+/-0.034	0.966+/-0.017
DT	0.83+/-0.07	0.965+/-0.008
NB	0.88+/-0.04	0.944+/-0.023

Table 3.2: Average of the leave-5-out cross-validation scores for all classifiers using the best sensor set for each of them. The pre-processing consisted of the sliding window technique in combination with summarizing by the standard deviation. The OG Test averages the LTS measure on the test set for the original classifier, while the PP Test is the same value for the post-processed classifier.

ranges from 59% to 86% for different sensor sets before post-processing, while using the post-processing results in a range of test scores from 93% to 96.5% and this is not specific to decision trees only.

The example shows that the post-processing is crucial. Firstly, it increases the accuracy of a given estimator on a given feature set by 35%. Secondly, it diminishes the impact of feature selection as the difference in accuracy between different feature subsets decreases substantially. Feature selection is of course still important as it decreases the computational complexity of the problem and allows to get rid of redundancy in the feature set. However, with methods that only rank features such as Relieff the choice of the threshold we choose to classify a feature as significant or not is less important. Finally and most importantly, the post-processing by projection allows to select a method according to criteria other than the performance, namely the computational speed.

3.5 Conclusion

In this paper we have introduced a post-processing scheme that allows to improve estimates. It finds estimated activities that are too short and eliminates them in an optimal way by finding the shortest path in a directed acyclic graph.

A simulation study is conducted to assess the benefits brought by the post-processing method. Generated noisy labels are improved with the use of the post-processing. The positive effects on the LTS measure are more significant when the noisy sequence contains more short intervals of misclassification.

Real-life football sensor data were used to assess the adequacy of the post-processing scheme in the more realistic setting. It significantly improved the performance of the classifiers. At the same time, the post-processed classifiers are closer to each other in performance than the original ones. This allows placing more importance on other criteria, such as the computational speed of the method. It should be noted that post-processing cannot correct for uncertainty in the classification result of the estimators. It can be seen in figures 3.3, 3.4 and 3.5 that the worse the original estimate the worse the post-processed one (at least as a rule of thumb as there can be cases when it is reversed). However, most

importantly, the results of the application to the football dataset are promising. The post-processing by projection was able to improve the estimators of accuracy ranging from 59% to 86% up to a score of 93% to 96.5%. We note that the lowest score of the post-processed estimates for any given classification method is still higher than the highest score of the original estimates. An alternative to our method could be to integrate the penalization of too short windows into the classifier. This is not an easy idea to realize, since classifiers usually do not consider the duration of activities themselves and they classify in a time linear manner. Nonetheless, if an appropriate scheme were to be defined, it would expand on the theory developed in this paper.

Another contribution are novel measures of classifier performance in the task of activity recognition using wearable sensors. They address the issue of timing offsets as well as unrealistic classifications, while retaining a typical scalar output of a performance measure allowing for easy comparisons between classifiers.

3.6 Appendix

3.6.1 Proofs

Proof of lemma 3.2.1 Let $\hat{f}$ be a solution of problem (3.3) for a given function f. Assume that $\hat{f}$ contains a jump t outside of the set $J(f)$. Denote the jump or one of the jumps closest to t in the original function f by t_k. Without loss of generality we assume t_k is located left of t (t_k exists otherwise f is constant and $\hat{f} = f$). Let t_a and t_b denote the jump preceding and resp. following t in the projection $\hat{f}$. Let t_{k+1} denote the jump following t_k in the original function f. Let s_1 be the state in which the original function stays in the interval $[t_k, t_{k+1})$ and let s_2 be the state from which the projection $\hat{f}$ jumps at t and left s_3 be the state to which the projection $\hat{f}$ jumps at t.

We will consider multiple cases and in each of them we will present a modification to $\hat{f}$ that either shows that $\hat{f}$ cannot be a projection or that there exists a function which is not worse than $\hat{f}$ and does not contain a jump at t. The configurations of the cases are depicted in Fig. 3.9.

1. $s_1 \neq s_2$
   (a) $t_a < t_k$. Moving the jump t to t_k does not increase the error (and potentially lowers it, if $s_3 = s_1$).
   (b) $t_a \geq t_k$. We move the jump t_a to t, which results in lowering the error by at least γ. Then we go back to the beginning of the proof with redefined state s_2 and jump t_a.
2. $s_1 = s_2$
   (a) $t_b \geq t_{k+1}$. Moving the jump t to t_{k+1} lowers the error by at least $t_{k+1} - t$ since $s_3 \neq s_1$.
   (b) $t_b < t_{k+1}$. We move the jump t to t_b, which results in lowering the error by penalty term γ and $t_b - t$, since $s_1 = s_2$. We go back to the case $s_1 = s_2$ with t moved to t_b and t_b moved to the next jump in $\hat{f}$ (if it does not exist, then $t_b = \infty$). Eventually the jump t can be moved to t_{k+1} (when case 2(a) is reached).

Loops occurring in cases 1(b) and 2(b) are not problematic, since with each iteration the number of jumps of the solution is reduced, eventually cases 1(a) or 2(a) are reached. □

Proof of lemma 3.2.2 Let $\hat{f}$ be a solution of the problem 3.3 for a given function f. Assume that for certain $\tilde{\gamma} < \gamma$, $\hat{f} \in \mathcal{G}_{\tilde{\gamma}}$ and $\hat{f} \notin \mathcal{G}_{\gamma}$. Hence there exist two jumps t_k and t_l of f and $\hat{f}$ (which follows from lemma 3.2.1), such that $\tilde{\gamma} < t_l - t_k < \gamma$. Since the state lasts less than γ, it can be removed (in the sense that one of the jumps is removed and either the previous state or following state is longer by $t_l - t_k$) with a gain in error of less than γ and decrease in error of exactly γ, which means we found a function with lower error than $\hat{f}$. This contradiction ends the proof. □

Proof of lemma 3.2.3 Let f be a function with two neighboring jumps t_1, t_2 and the state s_1 between them. Assume $t_2 - t_1 \geq 2\gamma$. Since the interval is longer than or equal to 2γ it satisfies the condition of the class $\mathcal{G}_{\gamma}$. Let us assume that the projection $\hat{f}$ of f contains two neighbouring jumps t_a and t_b such that $t_a \leq t_1 < t_2 \leq t_b$ and the state in the interval $[t_a, t_b)$ is $s_2 \neq s_1$. We introduce notation $\alpha := t_1 - t_a$ and $\beta := t_b - t_2$. If $\alpha, \beta \geq \gamma$, then introducing the jumps at t_1 and t_2 with the state s_1 between them is possible, because the condition of the class $\mathcal{G}_{\gamma}$ is satisfied. Moreover, the error is decreased if $t_2 - t_1 > 2\gamma$ and is not increased if $t_2 - t_1 = 2\gamma$. If $\alpha \geq \gamma$ and $\beta < \gamma$, then introducing a jump at t_1 such that the state following it is s_1 is possible. Moreover, the error is decreased. Analogously when $\alpha < \gamma$ and $\beta \geq \gamma$. If $\alpha, \beta < \gamma$, then changing state s_2 to s_1 reduces the error.

In all cases, we have shown that there exists a projection that does not change the state longer than 2γ. □

Proof of remark 3.2.2 Let $\hat{f}$ be a projection of f onto $\mathcal{G}_{\gamma}$. Let t_1 and t_2 be the first two jumps in the original function f. Let s_1 and s_2 be the first two states in the original function f. If $\hat{f}$ had the first jump at t_2 from the state s_1, then a function g equal to $\hat{f}$ outside of interval $[t_1, t_2)$, but such that the jump from state s_1 is moved to the location of the jump t_1 has an error lower than or equal that of $\hat{f}$. If $\hat{f}$ had the first jump at t_2 from a state $s_i \neq s_1$, then the error is infinite (since the value of $\hat{f}$ differs from f on the interval $(-\infty, t_1)$) and $\hat{f}$ cannot be a projection.

The argument is analogous for the penultimate jump. □

Proof of theorem 3.2.1 We use lemma 3.2.1 to prove that a projection of a function from $\mathcal{T}$ onto $\mathcal{G}_{\gamma}$ can only have jumps at the same positions as the jumps in the original function. This leads to the fact that finding the shortest path in the graph is equivalent to finding $\hat{f}$. □

Proof of lemma 3.2.4 Let $\hat{f}$ be a projection of f onto $\mathcal{G}_{\gamma}$. Let t_k and t_{k+1} be two consecutive jumps of f. Assume that $\hat{f}$ contains a jump t_k, but in opposite direction than in f. From lemma 3.2.1 we know that the next jump of $\hat{f}$ can occur at the earliest at t_{k+1}. This means that in the interval $[t_k, t_{k+1})$ the projection $\hat{f}$ is equal to $1 - f$. In this case, moving the jump at t_k to t_{k+1} (or in the case of $t_{k+1} \in J(\hat{f})$ removing both jumps) reduces the error

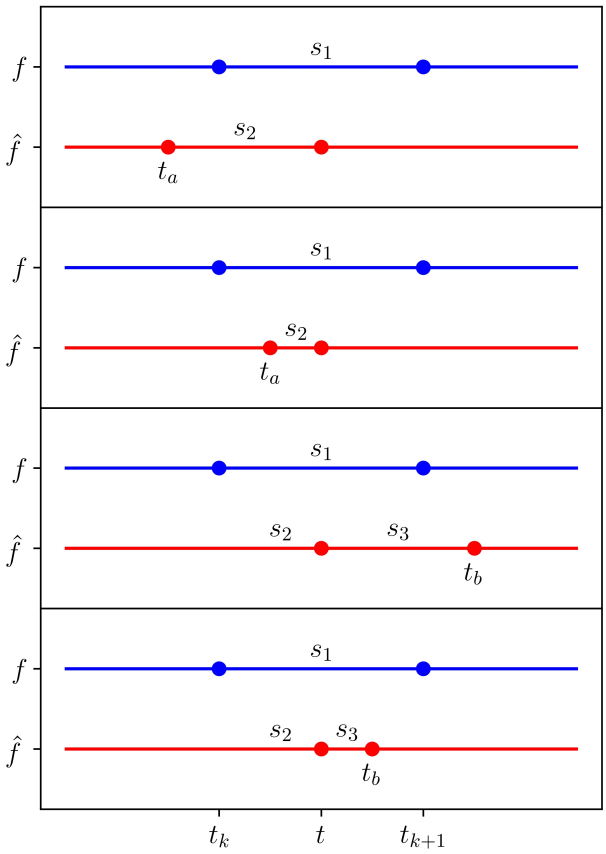


Figure 3.9: Illustration supporting the proof of lemma 3.2.1. Plots correspond (from top to bottom) to cases 1a, 1b, 2a, 2b respectively

by $t_{k+1} - t_k$. Hence, we conclude, a jump from f can only be present in its projection if it is in the same direction as in f. □

Proof of lemma 3.2.5 The proof of this lemma is analogous to the proof of lemma 3.2.2. The possibility of strengthening the previous result comes from the fact that we can remove two jumps at once, in effect reducing the error by 2γ.

Proof of lemma 3.2.6 The proof of this lemma is analogous to the proof of lemma 3.2.3

Proof of remark 3.2.3 Let $\hat{f}$ be a projection of f onto $\mathcal{G}_{2\gamma}$. Let t_1 and t_2 be the first two jumps in the original function f. Let 0 and 1 be the first two states in the original function f without loss of generality. By assumption $t_2 - t_1 < 2\gamma$ (note that without this assumption both jumps could be included in a projection). Since $[t_1, t_2)$ is not a valid activity (shorter than γ), if $\hat{f}$ has a jump at t_2, it does not have a jump at t_1. If $\hat{f}$ had a jump at t_2 from the state 0, then a function g equal to $\hat{f}$ outside of interval $[t_1, t_2)$, but such that the jump from state 0 is moved to the location of the jump t_1 has lower error than $\hat{f}$. If $\hat{f}$ had a jump at t_2 from the state 1, then the error is infinite (since the value of $\hat{f}$ differs from f on the interval $(-\infty, t_1)$) and $\hat{f}$ cannot be a projection.

The argument is analogous for the penultimate jump. □

Proof of theorem 3.2.2 We use lemmas 3.2.1 and 3.2.4 to prove that a projection of a function from $\mathcal{T}$ onto $\mathcal{G}_\gamma$ can only have jumps at the same positions and in the same directions as the jumps in the original function. This leads to the fact that finding the shortest path in the graph is equivalent to finding $\hat{f}$. □

GTS distance with $w > 0$ and $\sigma = \infty$ is an extended metric We will show that:

$$GTS_w(f, g) = \inf_{\epsilon \in \mathbb{R}} \{\text{dist}(f \circ \tau_\epsilon, g) + w|\epsilon|\}$$

is an extended metric on $\mathcal{T}$.

0. Since for any ϵ, $\text{dist}(f \circ \tau_\epsilon, g) \geq 0$ and $w|\epsilon| \geq 0$ we conclude that the GTS_w is non-negative.

1. It is obvious to see that $GTS_w(f, f) = 0$ for any $f \in \mathcal{T}$. Now let us assume that for some $f, g \in \mathcal{T}$ we have $GTS_w(f, g) = 0$. This implies that

$$\exists_{(\epsilon_n)} \ \text{dist}(f \circ \tau_{\epsilon_n}, g) + w|\epsilon_n| \xrightarrow{n \to \infty} 0.$$

Since $\text{dist}(f \circ \tau_{\epsilon_n}, g) + w|\epsilon_n|$ is an upper bound of $\text{dist}(f \circ \tau_{\epsilon_n}, g)$ and $w|\epsilon_n|$, we have

$$|\epsilon_n| \xrightarrow{n \to \infty} 0,$$

$$\int_{\mathbb{R}} d(f \circ \tau_{\epsilon_n}(t), g(t)) d\lambda(t) \xrightarrow{n \to \infty} 0.$$

From Fatou's lemma we have

$$\int_{\mathbb{R}} \liminf_{n\to\infty} d(f(t-\epsilon_n), g(t))d\lambda(t) = 0,$$

where λ is the Lebesgue measure on $\mathbb{R}$. Because f and g are càdlàg, this implies that for almost all t we have $f(t-) = g(t)$ or $f(t) = g(t)$ and so we conclude that $f = g$.

2. Let $f, g \in \mathcal{T}$, we have

$$\begin{aligned} GTS_w(f,g) =& \inf_{\epsilon}\{\text{dist}(f\circ\tau_\epsilon, g) + w|\epsilon|\} = \inf_{\epsilon}\{\text{dist}(g\circ\tau_{-\epsilon}, f) + w|-\epsilon|\} \\ =& \inf_{-\epsilon}\{\text{dist}(g\circ\tau_\epsilon, f) + w|\epsilon|\} = \inf_{\epsilon}\{\text{dist}(g\circ\tau_\epsilon, f) + w|\epsilon|\} \\ =& GTS_w(g,f), \end{aligned}$$

hence we conclude that GTS_w is symmetric.

3. Letting $f, g, h \in \mathcal{T}$, we have

$$\begin{aligned} GTS_w(f,g) =& \inf_{\epsilon}\{\text{dist}(f\circ\tau_\epsilon, g) + w|\epsilon|\} \\ =& \inf_{\epsilon_1,\epsilon_2}\{\text{dist}(f\circ\tau_{\epsilon_1}\circ\tau_{\epsilon_2}, g) + w|\epsilon_1+\epsilon_2|\} \\ \leq& \inf_{\epsilon_1,\epsilon_2}\{\text{dist}(f\circ\tau_{\epsilon_1}\circ\tau_{\epsilon_2}, h\circ\tau_{\epsilon_2}) + \text{dist}(h\circ\tau_{\epsilon_2}, g) + \\ & + w|\epsilon_1| + w|\epsilon_2|\} \\ =& \inf_{\epsilon_1,\epsilon_2}\{\text{dist}(f\circ\tau_{\epsilon_1}, h) + w|\epsilon_1| + \text{dist}(h\circ\tau_{\epsilon_2}, g) + w|\epsilon_2|\} \\ =& \inf_{\epsilon_1}\{\text{dist}(f\circ\tau_{\epsilon_1}, h) + w|\epsilon_1|\} + \inf_{\epsilon_2}\{\text{dist}(h\circ\tau_{\epsilon_2}, g) + w|\epsilon_2|\} \\ =& GTS_w(f,h) + GTS_w(h,g), \end{aligned}$$

which shows that GTS_w satisfies the triangle inequality and that concludes the proof. □

The LTS distance with $w > 0$ is a semimetric Let $w > 0, \sigma > 0$ and a metric d on $\mathcal{S}$ be fixed. We observe that $LTS_{w,\sigma}$ is nonnegative. Symmetry of $LTS_{w,\sigma}$ follows directly from the definition. It only remains to show that $LTS_{w,\sigma}(f,g) = 0$ if and only if $f = g$ for $f, g \in \mathcal{T}$.

We have

$$LTS_{w,\sigma}(f,f) = 0,$$

because there is only one segment (as defined in 3.3.2). Assume now that $LTS_{w,\sigma}(f,g) = 0$ and $f \neq g$. In that case, there exists more than one segment.

$$\begin{aligned} LTS_{w,\sigma}(f,g) = \sum_{i=1}^{l-1} \delta_i(a_{i+1} - a_i)d(f(a_i), g(a_i)) = 0 \\ \Rightarrow \forall_{i=1,2,3,\ldots,l-1} \quad f(a_i) = g(a_i), \end{aligned}$$

which implies that $f = g$, which contradicts the assumption. We conclude that $LTS_{w,\sigma}(f,g) = 0$ iff $f = g$, which completes the proof. □

4

Imputation methods for sensor reduction

4.1 Introduction

In the field of sports analytics, efficiency needs to go hand in hand in the appropriate incorporation of diverse feedback from both coaches and players. This encompasses not only performance metrics and tactical observations provided by coaches but also subjective insights and experiences shared by players, all of which are important for improving our data analysis models. In Chapters 2 and 3 we considered applications to football activity recognition using sensor data and observed advantages of using sensor technology. Sensor trousers worn by the players measure acceleration and angular velocity on different body parts. As seen in fig. 2.1c, the sensor trousers prototype covers both the upper as well as lower legs and the respective sensors are woven into the fabric of the prototype. However, the conventional football shorts do not go lower than the knees for comfort and freedom of movement. Hence, it is reasonable to consider a version of the trousers which covers only upper legs, removing lower legs sensors in the process. In addition, it is natural that producing such trousers would incur lower costs as well. In order to make this possible, we consider in this chapter a scenario in which parts of the sensor data are excluded from the analysis and observe the effect it may have on the accuracy of the predictive model.

Removing parts of the data has the potential to significantly alter the outcomes of data-driven studies. It is usually worth employing imputation techniques when faced with gaps in the data, in order to preserve the integrity and the structure of the entire dataset. While our previous exploration in chapters 2 and 3 concentrated on football activity recognition, primarily from a modeling standpoint, our current focus shifts towards a critical data-centered inquiry. We examine how the outcomes of our study in chapter 2 evolve in the situation where specific sensors are removed from the dataset and in which ways we can mitigate it.

Our exploration extends beyond merely identifying the impact of missing shank data on the study results. We also discuss strategies for imputing this absent data, considering alternative sources such as the thighs and pelvis as the guide. Constructing physically-inspired relevant features based on the existing sensors becomes crucial in maintaining the

robustness of our study, allowing us to adapt to the potential and likely changes in sensor configurations and ensuring the continued relevance and reliability of our findings.

The problem of imputation usually considers data that are missing only in part mostly due to equipment malfunction, lack of compliance or human error. This is notably different from our case, but we review the existing literature on the topic in order to draw inspiration for our problem. In this brief literature review, we identify a selection of popular ideas in this field and what they might offer to our particular application. For a comprehensive survey, see [69].

In general, multiple approaches are available to tackle the issue of missing data. One approach is to delete all entries with missing values, sometimes only if specific features of the entry are missing [70]. Most approaches focus on imputing the data in some way. This can be done in a multitude of ways. Simple imputation, using a single attribute such as mean or median of all the available values, is easy to use, but often it produces unrealistic results [71]. Hence this approach is not recommended for most datasets (especially in the case of larger datasets). Imputation using regression is one of the most popular [72]. It uses all the complete entries to estimate the regression coefficients, then missing data are replaced by the predicted values based on the regression model [73]. Another approach utilizes a k-nearest neighbor model (kNN) [74]. In this case, a missing value within a dataset is estimated by identifying the entries in training dataset, which are the closest to the entry with the missing value. Two entries are considered to be close if the features that neither is missing are close [75]. There is a variety of other methods which we will not consider in this chapter.

The sensor data are time series recorded with high frequency (500 Hz), which encourages treating them as functions rather than time series. To best utilize the nature of the sensor data, we employ functional data analysis techniques such as functional regression and functional principal component analysis (fPCA). This chapter is structured as follows. Section 4.2 discusses the methods used for imputation of missing data, including fPCA and functional regression (fReg). Section 4.3 presents the results of applying our activity recognition neural network on the full, the imputed and the reduced dataset. Section 4.4 presents the conclusions of the study and the recommendations based thereof.

4.2 Framework

We consider two different approaches to the imputation of data. One is based on fPCA. It uses linear regression to recover the missing fPCA scores needed for the reconstruction of the missing sensor. The other is based on fReg, where the missing sensors, treated as function, are recovered directly. The next two subsections recap chapters 8 and 12 from the classic book on functional data analysis by Ramsay and Silverman [76], to succinctly describe the two methods: fReg and fPCA.

4.2.1 Functional regression

Let $\{\mathbf{X}(t)\}_{t\in[0,T]}$ be a vector-valued function

$$\mathbf{X}(t) = (X_1(t), ..., X_p(t)), \tag{4.1}$$

where t represents the time. Let X_i for some specific $i = 1, ..., p$ be the function we would like to remove and X_j for $j \neq i$ be the *predictors*.

Consider a *concurrent functional regression model*

$$X_i(t) = \alpha(t) + \sum_{j:\, j\neq i} X_j(t)\cdot\beta_j(t) + \epsilon(t),$$

where α and β_j are regression coefficient functions on $[0,T]$, while ϵ is the error term function. This simple model is point-wise in nature. The conditional expectation of $X_i(t)$ given the functions $\{X_j(s) : j \neq i,\ s \in [0,t]\}$ only depends on the values of these functions at time t.

One can also adopt a more global approach to regression with the *global functional regression model*:

$$X_i(t) = \alpha(t) + \sum_{j:\, j\neq i} \int_0^T X_j(s)\cdot\beta_j(s,t)\, ds + \epsilon(t).$$

Here, the conditional expectation of $X_i(t)$ given the functions $\{X_j : j \neq i\}$ depends linearly on all values of these functions on $[0,T]$. Additional complexity is introduced into the model; $\beta_j(s,t)$ is now bivariate and its interpretation is now different: it describes a relationship of the predictor X_j at time s and the response X_i at time t. This model will be used during the study and we will refer to it as fReg in abbreviation.

In the context of our application, the functional regression model will be used in a straightforward manner. If m components (corresponding to sensors) are removed from vector $X(t)$, then m regression models will be constructed, all based on the $p-m$ remaining sensors.

Now consider N observations of the random function $\mathbf{X}$, denoted by

$$x^l(t) = (x_1^l(t), \ldots, x_p^l(t)),$$

for $l = 1,...,N$. To fit the model, a common approach is to use a basis expansion for the regression coefficient functions. Since the functions are bivariate in the global functional regression model, the expansion needs to be double, i.e. it is in terms of two separate basis functions, one expressed in terms of argument s and one expressed in terms of argument t. An expansion is also needed for the intercept function α. For details on basis expansion of α and β functions, we refer to [76]. To find the coefficients of the expansion of those functions, an objective criterion needs to be minimized:

$$\begin{aligned} LMSSE(\alpha,\beta) &= \int_0^T \sum_{l=1}^N \left[X_i^l(t) - \alpha(t) - \sum_{j:\, j\neq i} \int_0^T X_j^l(s)\cdot\beta_j(s,t)\, ds \right]^2 dt \qquad (4.2) \\ &= \int_0^T \sum_{l=1}^N \left[X_i^l(t) - \sum_{r=1}^{K_2} a_r\cdot\theta_r(t) - \sum_{j:\, j\neq i} \int_0^T X_j^l(s)\cdot \sum_{q=1}^{K_1}\sum_{r=1}^{K_2} b_{qr}\cdot\eta_q(s)\cdot\theta_r(t)\, ds \right]^2 dt \end{aligned}$$

where $\{\eta_1,\ldots,\eta_{K_1}\}$ and $\{\theta_1,\ldots,\theta_{K_2}\}$ are two sets of basis functions on $[0,T]$, and $\{a_1,\ldots,a_{K_2}\}$, $\{b_{11},b_{12},\ldots,b_{K_1(K_2-1)},b_{K_1K_2}\}$ are coefficients. To effectively minimize expression (4.2), restriction on the number of basis functions used for the expansion of α and β is considered to avoid overfitting.

4.2.2 Functional PCA

Functional PCA is one of the most important methods for obtaining characteristics of functional data. FPCA extracts functions which explain modes of variation of functional data. This process is incremental; the first function depicts the dominant mode of variation of the functional data, the kth function depicts the dominant mode of variation that is orthogonal to the $k-1$ previous functions. It is an effective way of studying the variation present in the dataset without obtaining covariance and correlation functions. Consider an inner product of weight function β with a function X:

$$\int_0^T \beta(s)X(s)\,ds, \tag{4.3}$$

where β and X are square integrable. The goal of fPCA is to find modes of variations in the function X, represented by weight functions, which allows us to better understand the ways in which the functions that compose the dataset differ from each other. Let x^l be observations of function X for $l = 1,\dots,N$ defined on $[0,T]$. Function β_1 maximizes the expression

$$\frac{1}{N}\sum_{l=1}^{N}\left(\int_0^T \beta_1(s)x^l(s)\,ds\right)^2 \tag{4.4}$$

under the condition that

$$\int_0^T \beta_1(s)^2\,ds = 1. \tag{4.5}$$

For weight function β_m, with $m = 2,\dots,K$, the above scheme also applies but with orthogonality constraints:

$$\int_0^T \beta_k(s)\beta_m(s)\,ds = 0,\ k < m. \tag{4.6}$$

Weight functions are used for the calculation of functional principal components (4.3) because they influence how the variability in different parts of the function space is weighted.

There is a different way to motivate the fPCA, as seen in [76], namely by defining the problem: find exactly K orthonormal functions β_k such that a certain basis expansion of each observation x^l approximates the curve as closely as possible. There the expansion is of the form

$$\hat{x}^l(t) = \sum_{k=1}^{K} f_{lk}\beta_k(t), \tag{4.7}$$

where f_{lk} is the principal component value equal to

$$\int_0^T x^l(t)\beta_k(t)\,dt.$$

For each individual curve, we consider an objective criterion as the integrated squared error $\|x^l - \hat{x}^l\|^2 = \int_0^T \left(x^l(s) - \hat{x}^l(s)\right)^2 ds$ and globally we consider the sum of integrated squared errors:

$$PCASSE = \sum_{l=1}^{N} \|x^l - \hat{x}^l\|^2. \tag{4.8}$$

Given this objective criterion the problem can be stated as follows: what choice of orthonormal basis minimizes criterion (4.8)? The answer is that the functions $\beta_1, \ldots, \beta_K$ defined as functions maximizing expression (4.4) and satisfying conditions (4.5) and (4.6) (for β_m with $m > 1$) minimize criterion (4.8). It is important to note that this is merely a motivation for the fPCA and not a method of finding the weight functions, since changing the order of the weight functions would also satisfy criterion (4.8). In practice, curves x^l are used to estimate the weight functions β_m by solving a functional eigenanalysis problem, see [76] for more details.

Going back to our problem of sensor reduction, we will use fPCA to reduce the dimensionality of our problem. Instead of functions of time, each variable will be represented by K scalar values f_{lk} for each curve l. Using these scalar values and the orthonormal basis β_m, we can reconstruct the functions of time on their whole domain. When sensor X_l is removed, we will need to recover the values f_{lk} using regression on all the remaining principal component values. Using the recovered values the original sensor can be reconstructed using expansion (4.7).

4.3 Results

The football data used for this study are the same as the data introduced in section 3.4.2. It is important to note that data from just one specific player is considered here. Throughout this chapter, the term neural network will always refer to a specific architecture. This architecture contains both a convolutional and a recurrent layer in its design. The convolutional layer calculates a multitude of inner products between part of the input signal and a filter, which runs through the whole signal, creating in effect a smaller, more condensed signal. The recurrent layer carries on the information from not only the current layer but also prior ones, allowing for long term dependencies in the signals to be preserved. The convolutional part of the network combines the convolutions which use the same filters for all the signals and the convolutions with the same filters for the signals of the same sensor type. The recurrent part of the network uses bidirectional LSTM to carry information of

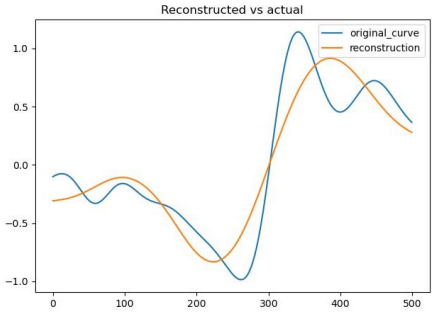


Figure 4.1: An example of reconstruction using fPCA. The blue line denotes the curve of one sensor from the data and the orange line denotes the reconstruction of this curve using fPCA.

the input signal from the past. This architecture was chosen due to its good performance on football data and was already presented in chapter 2.

General activity	A_1	A_2	A_3
Running	Jog, sprint	Jog, sprint	3 speeds of running
Turning	180° turn, 90° turn		
Jumping	Standing jump	Standing jump	Standing jump
Passing	Short pass	Short pass	Short pass
Shooting	Shot after run-up	Shot after run-up	Shot after run-up

Table 4.1: The table shows the activities included in each of the three sets A_1, A_2, A_3.

The study is set up as follows

- three sets of activities A_1, A_2 and A_3 are selected as shown in table 4.1,
- we consider multiple sensors to remove, just left or right shank or both and then impute the missing data using either fPCA or fReg based on the current data set, see 4.1 for an example,
- finally we conduct five runs of cross-validation with the dataset split into training (70%) and testing subsets (30%) and using the neural network and the imputed data for prediction.

Table 4.2 shows the results of the study for different sets of activities, different methods of imputation and different sensors removed/replaced. Compared to the prediction based on the full dataset, we see that the quality of the prediction drops regardless of the method of imputation. Notably, in all cases, the model performs better using imputed data than the reduced dataset with sensors missing. It performs especially well on A_2, which is the most limited dataset in terms of the activities considered. There we see the smallest drop in accuracy from the full dataset to the imputed ones. Both methods used for imputation perform equally well.

Removed sensor	Imputation method	A_1	A_2	A_3
None	None	81.4%	96.7%	84.0%
Left shank	fPCA	80.1%	96.1%	82.2%
	fReg	79.4%	95.8%	81.9%
	None	77.2%	95.1%	79.7%
Right shank	fPCA	80.2%	95.9%	82.0%
	fReg	79.7%	95.7%	82.1%
	None	77.7%	95.3%	80.3%
Both shanks	fPCA	78.0%	93.2%	78.3%
	fReg	75.9%	92.5%	77.2%
	None	74.4%	90.4%	75.3%

Table 4.2: The table shows the mean test accuracy of the neural networks on the 3 different datasets and using different methods of imputation.

4.4 Conclusions and discussion

This chapter tackles a feature reduction problem in the context of sensor data in football. Usually, when discussing imputation of missing data, one faces the fact that the data are already incomplete, which makes the problem more difficult. Certain sensors may potentially be removed from the sensor trousers, whether due to cost considerations, for athlete's comfort or for compliance with traditional clothing. We examine the impact of removing sensors on the dataset and explore methods for effectively recovering missing sensor data using the remaining sensors.

We use two different methods of recovering the sensor signal. One uses functional PCA scores to predict the scores of the missing curves and recover the signal using basis expansion, the other uses a functional regression model and a direct recovery from the remaining functional variables. Based on the results of the study, we conclude that both methods perform comparably well. First, they allow higher performance of the predictive model than on the reduced dataset. Second, both methods are quite similar in terms of the accuracy that they achieve and it is hard to distinguish between their performance.

A further study could explore various activity sets. Testing alternative predictive models can further improve upon our imputation method. While the current study focused on a single player, extending the investigation to encompass multiple players could shine the light on whether this method could be employed on a wider scale. When applying this method to new players with incomplete data, it is important to acknowledge that the method may not be as accurate due to the physical differences between new players and players whose data was used to train the model. A study focused on the effects of mixing sensor data from different players could give an indication of the robustness of the method.

5

Testing for no effect in regression problems: a permutation approach

5.1 Introduction

With the ubiquity of data often the question whether a response Y can be predicted based on predictors X arises. The rise of highly capable machine learning and deep learning techniques increases the abilities to fit any kind of data. However, the abilities to fit pure noise are increasing as well. We propose a method to test whether a model is only fitting noise. It extends testing for no effect from linear to nonlinear models. No sample splitting is performed so the power of the test can rely on the size of the whole sample. No nested sequence of models is needed, in fact, no alternative models are needed at all.

Our method is based on recombining the pairings between predictors and responses through permutations. In this way artificial reference datasets are created and the performance of the model on the original data can then be assessed by comparing it to the performances on the artificial reference datasets. The purpose of our test is to ascertain whether the model is capable of fitting the data more effectively than mere random noise. Our method is not restricted to linear models since it is not a test for specific parameters in the model. Rather it tests for the ability of a model to predict the responses.

The main contribution of this paper is a rigorous formulation of a permutation test for dependence between model predictions and responses. The test uses R^2 as test statistic but can be performed with any measure of goodness of fit in regression analysis. Because of its interpretability, R^2 is our test statistic of choice, but this can be adapted if necessary. The method generates new pairings of (X_i, Y_j) conditional on the X_i for $i = 1, ..., n$ and Y_j for $j = 1, ..., n$. This paper introduces a new formulation of the null hypothesis and provides a rigorous justification for a permutation test that has been described in various forms in the literature, for instance in the two-sample problem ([78–81]), the stochastic dominance

This chapter is based on *M. Ciszewski, J. Söhl, A.J.R. Leenen, B. van Trigt, G. Jongbloed. Testing for no effect in regression problems: a permutation approach, arXiv, 2023* [77].

problem ([82, 83]) or the subgroup discovery problem ([84]). The main use case for this method is in the initial stages of the data analysis to test whether a given model does only fit noise or is able to capture some essential structure in the data.

The outline of the paper is as follows. Section 5.2.1 formulates the problem and introduces necessary notation. Section 5.2.2 provides the historical context and an overview of the current state of the literature on permutation tests in regression problems. Section 5.2.3 contains the formal formulation of the null hypothesis, theoretical considerations and the succinct description of the method. Section 5.3.1 presents a simulation study, where the permutation test is demonstrated in various scenarios for predictors and responses. Section 5.3.2 contains the application of the permutation test to sensor data of tennis serves in order to demonstrate the method in practice and showcase its power in a real-life scenario.

5.2 Methodology

5.2.1 Problem description

Consider a regression setting. Given an observed pair (X,Y), where X is a random vector and Y is a real random variable. Y is modelled as:

$$Y = f(X) + \epsilon,$$

where ϵ is a centered random variable independent of X and $(f(X))_{f\in\mathcal{F}}$ for some class of functions $\mathcal{F}$. An example of $\mathcal{F}$ could be a set of all linear functions corresponding to a linear regression model with fixed number of variables or a set of functions that can be described by a neural network. Nonparametric classes of functions can also be considered, for instance a set of log-concave functions. For the remainder of the paper, we will focus on R^2 as goodness-of-fit measure.

Since the actual relationship between X and Y is not known in practice, a chosen class of functions $\mathcal{F}$ through which that relationship is described does not need to be appropriate. The class of functions $\mathcal{F}$ is misspecified if it does not contain the true f, while if it contains too many functions, the model might be overfitting by memorizing the noise ϵ. In a real world scenario, we are often facing datasets that feature high-dimensional, time-dependent or functional variables. The question whether there is a relationship between X and Y and which model to choose for describing it, is crucial. In this paper, we focus on the following aspect:

- can a given class of functions $\mathcal{F}$ distinguish Y from pure noise?

Consider this simple example. Let X_1, X_2 be independent standard normal variables and $Y = X_1^2 + X_2^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.01)$. Consider a multi-layered neural net as a model of choice to predict Y using X_1 and X_2. For small sample sizes shuffling the vector of responses and applying our prediction model to this shuffled dataset can yield values of R^2 higher than values of R^2 calculated for the prediction model applied to the original dataset. Ten random samples of size 10 were drawn. Five yielded higher values of R^2 for at least one shuffled dataset than for the original pairing (we considered 200 shuffles of the sample). In applied settings, where the sample size is fixed and difficult to increase, this presents an inherent issue. Sample size has an immediate influence on the credibility of the model and needs to be taken into account.

Related problems have been addressed before in the literature in different settings and with a variety of solutions. In this paper we focus on the permutation test. To provide context to our solution, we will give a brief review of the existing literature on permutation tests, before we specify the precise null hypothesis.

5.2.2 Permutation tests in the existing literature

Historically, the idea of permutation tests existed before it was feasible to realize them on a larger scale and one of the main pioneers in that regard was Ronald Fisher. Fisher's interest in randomization as a concept can be traced back to his 1925 *Statistical Methods for Research Workers* [85]. His initial idea related to the experimental design and to getting rid of researcher's bias in the setup of experiments [86]. At that time the use of randomization concerned only experiments with small numbers of samples (as it would be near impossible to apply it in case of large sample sizes). The application of randomization to hypothesis testing first emerged in the 1930s as introduced by Ronald Fisher in the form of a permutation test as the test for comparing the distributions of two independent groups. The use of this test, commonly known as the Fisher-Pitman permutation test, has since been extended to many other settings [87].

The rise of automated computing made these types of tests feasible to use on a larger scale. Since the 1970s, permutation tests have started to enjoy larger popularity. A good example of the renewed interest in permutation tests can be found in [88], which gives a justification based on the principle of unbiasedness for permutation tests. It also gives good reasons for the transition from the t-test to a permutation test. The paper [89] discusses the growing popularity of the permutation test as a substitute for the ANOVA F-test and investigates its robustness. Tests of partial regression coefficients in a linear model are considered in [90]. This paper explains the exact test and then compares the distributions of the test statistics under the various permutation methods proposed. The authors use the classic null hypothesis when testing for no effect. An excellent survey of various papers in the biomedical field can be found in [91]. It showcases the appeal of both the randomization in experimental design (as opposed to random sampling) as well as the permutation tests for differences in location. The increased popularity of permutation tests can be seen in other fields as well, e.g. in psychological research [87].

The applications of permutation tests have expanded beyond the comparison of statistical models to a variety of problems. However, the formal formulation of the null hypothesis can be challenging and subject to variations among authors, particularly when permutation tests are utilized as a problem-solving approach. A subset of publications uses the concept of exchangeability, i.e. the joint distribution of the observations is invariant under permutations of the order of the predictors in their formulation of the null hypothesis in the permutation test. Most commonly the null hypothesis is expressed in terms of exchangeability in two-sample problems, e.g. [78–81]. There have also been studies with regards to the robustness of permutation tests with respect to the assumption of exchangeability, i.e. in cases where exchangeability cannot be assured, what can be said about asymptotic properties of the test. These ideas are explored well, e.g. in [92] and [93]. More recently, a permutation test without the assumption of exchangeability has been applied to the two-sample problem [94]. The null hypothesis is the equality of two population means. Their split sample permutation t-tests are asymptotically exact and can

be extended to testing hypothesis about one population.

When discussing permutation tests, the question arises whether a model applies. Many permutation tests are specifically designed to tackle problems within linear regression models. However, there are applications to other models as well. A permutation test for no effect in a functional linear regression model has been proposed in [95]. In this case the randomization technique allows for the simulation of the conditional distribution of the test statistic, which otherwise would be difficult to obtain. Wide attention has been given to the the permutation approach in the stochastic dominance problem in testing for ordered categorical variables, introduced in [82]. Further developments can be found, for instance in [83]. In the problem of testing heterogeneity in two-sample categorical variables, [96] proposes a permutation test. Here, the null hypothesis considered is simply the equality of heterogeneity of two population distributions. In a subgroup discovery problem, [84] employs a randomization technique. The test devised for this purpose is based on a null hypothesis that the quality measure (for instance R^2) of a given subgroup is generated by the distribution of false discoveries (DFD), which arises from the central limit theorem applied to a set of quality measure values on some baseline subsets. Permutation tests have also been applied to linear mixed models, more specifically as a test for random effects. For instance, [97] presents two permutation tests, one based on the best linear unbiased predictors and another based on the restricted likelihood ratio test statistics. Both methods involve weighted residuals, with the weights determined by the among- and within-subject variance components. The lack of flexibility in permutation tests, particularly in the context of experimental design of the model, is considered by [98, 99]. Their main focus is on the tests of no effect in a general linear model and their main achievement is the unification of a diverse set of results. The outcome is a single permutation strategy with a single generalized measure. It is worth noting that, once again, the assumption of exchangeability is considered in the formulation of the null hypothesis for the permutation test. Permutation approaches have also been applied to nonparametric ANOVA designs as seen in [100]. There the synchronized permutation method is extended specifically to unbalanced two-level ANOVA. The problem of testing independence given a sample from a bivariate distribution has been considered in [101]. The method used there relies on studentizing the sample correlation which leads to a permutation test that is exact under independence while asymptotically controls the probability of type 1 errors. Permutation tests have also been used in an application of a multivariate regression analysis to a study of factors influencing mental health during the COVID-19 lockdown period [102]. In this particular study, a combined permutation test on data collected in a survey is applied. The null hypothesis is that the regression coefficients are zero. An application to weighted regression models can be found in [103]. A comparison of X-permutation and Y-permutation and their variability in the weights is given there, inspired by the nature of the weighted regression models in which the permutation of the response variables and the permutation of the predictors do not lead to the same result. An extensive overview of permutation tests, their theoretical properties as well as a vast number of applications, can be found in [104].

Overall, the main appeal of permutation tests stems from the fact that they do not require any distributional assumptions on the population. The lack of assumptions is increasingly more interesting to researchers as deep learning methods become more popular since they likewise do not rely on distributional assumptions. Permutation tests are completely

data-driven as pointed out by [87]. This can be very appealing as the data is the main factor in shaping the distribution of the test statistic, i.e. the test statistic can be chosen to be more easily interpretable without focusing on its distribution.

Our work differs from previous work most in the formulation of the null hypothesis. Based on the publications mentioned in this section, we have seen a few different null hypotheses. Some involve the concept of exchangeability, some equality of means or zeroing of the coefficients. In contrast to this, we focus on the concept of independence, which is not widely used for permutation tests. Permutation tests of independence have existed before, e.g. [105]. However, we do not test independence of two random variables X and Y, but rather we state the null hypothesis in terms of the model and whether it is able to capture the dependence.

The choice of the null hypothesis can also be directly connected to the model considered in the problem. For instance, it is natural to use zeroing of the functional coefficient as the null hypothesis when considering a functional linear regression model [95]. We do not restrict ourselves to any particular model in our work, we only consider the model as given and the null hypothesis is not specifically tailored to the model.

The subsequent section will concentrate on establishing a rigorous theoretical foundation for our permutation test.

5.2.3 Permutation approach to testing for no effect

Our goal is to investigate whether a class of functions $\mathcal{F}$ can capture any dependence between X and Y. We consider a test with null hypothesis stated as follows:

$$H_0 : Y \text{ is independent of } (f(X))_{f\in\mathcal{F}}. \tag{5.1}$$

H_0 represents the problem as described in section 5.2.1. If it were true, then our class of functions $\mathcal{F}$ will not be able to capture the relation between X and Y in a meaningful manner. Considering a dataset with permuted responses will be no different to class of functions $\mathcal{F}$ under H_0. If H_0 is false, then the class of functions $\mathcal{F}$ will be able to capture some aspects of the relation between X and Y, although it does not guarantee that the model is suitable and readily applicable.

The null hypothesis H_0 as stated in (5.1) does not guide the choice of the test statistic. In order to choose a suitable test statistic, further understanding of H_0 is needed.

Proposition 5.2.1. *Let $(X_1,Y_1),...,(X_n,Y_n)$ be an i.i.d. sample of (X,Y). If Y is independent of $(f(X))_{f\in\mathcal{F}}$, then for all $i = 1,\dots,n$ the conditional distribution of*

$$\big((f(X_i),Y_{\tau(i)})\big)_{f\in\mathcal{F}} \tag{5.2}$$

given the empirical measure P_n^X of $X_1,...,X_n$ and the empirical measure P_n^Y of $Y_1,...,Y_n$ is the same for all permutations τ of set $\{1,...,n\}$.

Proof. Let $i \in \{1,\dots,n\}$ be fixed. For a given finite collection of functions $f_1, f_2, ..., f_m \in \mathcal{F}$ and a permutation τ, the conditional joint distribution of $(f_1(X_i),Y_{\tau(i)}),...,(f_m(X_i),Y_{\tau(i)})$ given P_n^X and P_n^Y is the same as the joint distribution of

$$(f_1(X_i),Y_i),...,(f_m(X_i),Y_i), \tag{5.3}$$

thanks to the assumption of independence of $(f(X))_{f\in\mathcal{F}}$ and Y. Note that (5.3) is invariant with respect to the permutations of Y_i. This statement will also be true if extended to a joint distribution of (5.2) thanks to Kolmogorov extension theorem ([106]), hence the distribution of joint conditional distribution of (5.2) given P_n^X and P_n^Y is invariant with respect to the permutation of Y_i. □

Before proposition 5.2.1 is translated into a result in terms of R^2, we formally define R^2. Consider n realizations of (X,Y) and denote them as $(x_1,y_1),...,(x_n,y_n)$. Let L be a loss function and $\hat{f}$ be an empirical risk estimator in the sense that

$$\hat{f} = \arg\min_{f\in\mathcal{F}} \sum_{i=1}^{n} L(f(x_i), y_i). \tag{5.4}$$

Let $\hat{f}(x_i)$ denote the prediction of y_i for $i = 1,...,n$. Then

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{f}(x_i))^2}{\sum_i (y_i - \bar{y})^2}, \tag{5.5}$$

where $\bar{y}$ is the mean of the y_i. This definition of R^2 is the natural one if the loss function L in equation (5.4) is chosen to be the squared error loss. In the context of R^2, proposition 5.2.1 implies the following result.

Proposition 5.2.2. *Let $(X_1,Y_1),...,(X_n,Y_n)$ be an i.i.d. sample of (X,Y). Assume that Y is independent of $(f(X))_{f\in\mathcal{F}}$. Fix a permutation τ of $\{1,...,n\}$ and a loss function L defining an empirical risk estimator as in* (5.4)*. Then, conditionally on the empirical measure P_n^X of $X_1,...,X_n$ and the empirical measure P_n^Y of $Y_1,...,Y_n$, the distribution of R^2 calculated based on data $\{(X_i,Y_{\tau(i)})\}$ using the aforementioned emprirical risk estimator does not depend on τ.*

Proof. Proposition 5.2.1 implies that the conditional distribution of

$$\left(\sum_{i=1}^{n} (Y_{\tau(i)} - f(X_i))^2, \sum_{i=1}^{n} L(f(X_i), Y_{\tau(i)}) \right)_{f\in\mathcal{F}} \tag{5.6}$$

given P_n^X and P_n^Y is the same for all permutations τ of set $\{1,...,n\}$. This is a two-dimensional empirical process indexed by class of functions $\mathcal{F}$. Plugging in the arg min of the second component into the first component still gives a distribution that does not depend on τ. Hence, combining the definition (5.4) of $\hat{f}$ and (5.5) of R^2, we conclude that for each permutation τ, R^2 calculated for $\{(X_i,Y_{\tau(i)})\}$ is sampled from the same distribution conditioned on P_n^X and P_n^Y. □

This allows us to consider R^2 as a viable choice for the test statistic. Under the null hypothesis, the R^2 as calculated for (x_i,y_i) is sampled from the same distribution as the R^2 calculated for $(x_i,y_{\tau(i)})$ for some permutation τ. The test itself is based on permutations of the pairings (x_i,y_i). We reject H_0 only if the observed R^2 is much larger than "most" of the R^2 obtained via random permutations. Essentially we compare the observed R^2 to the distribution of R^2 under H_0 given specific realizations of X and Y, but not their pairings. It

is notable that R^2 can also be replaced by some other statistic, as long as it can be calculated using the sample $\{(f(x_i), y_i)\}_i$. Proposition 5.2.1 permits other statistics to be used instead of R^2. Taking R^2 as the test statistic is equivalent to taking empirical risk with respect to quadratic loss as the test statistic. In that sense, the other tests can also be constructed by considering empirical risks with respect to other losses, e.g. absolute loss or Huber loss.

If the class of functions $\mathcal{F}$ contains the constant functions and the predictors are optimized with respect to the quadratic loss, then R^2 calculated for a given $\mathcal{F}$ is always non-negative. This is true, since given set of observations $\{y_i\}_{i=1,...,N}$, we can always choose $f(X) \equiv \frac{1}{N}\sum_{i=1}^{N} y_i$ which yields $R^2 = 0$. Note that including the constants in $\mathcal{F}$, does not disturb the independence of Y and $(f(X))_{f\in\mathcal{F}}$, since Y is always independent of a set of constant random variables. While R^2 is always non-negative in linear regression models (if the intercept is included), that is not the case for instance in the setting of neural nets.

Given a chosen α level*, the precise implementation of the test is as follows:

1. given original pairings of (x_i, y_i), calculate the R^2 of class of functions $\mathcal{F}$, which we will denote as r_0^2,**
2. find the distribution of R^2 under the null hypothesis conditionally on observed x_i and $y_{(i)}$ for $i = 1, ..., n$ (approximated by the empirical distribution function of R^2 values based on a uniform sample of permutations of original pairings (x_i, y_i); for each sample $\{(x_i, y_{\tau(i)}) : 1 \le i \le n\}$, where τ is a permutation, R^2 is calculated; notably, the model is refit for each permuted sample),
3. if $r_0^2 > q_{1-\alpha}$, where $q_{1-\alpha}$ is the $1-\alpha$ quantile of the empirical distribution of R^2 values, then we reject the null hypothesis, otherwise we do not reject it.

Any tuning parameters used in point (1) and (2) are not adjusted for each permutation. This implementation assumes that R^2 is the statistic of choice, but it can be adapted to suit other statistics as well. The reason we prioritize R^2 is primarily because of its benefits in terms of interpretability and ease of use. It is also important to note that in practice, determining the distribution of R^2 under the null hypothesis will not be exact in most cases. To obtain the exact distribution we need to run through $n!$ permutations. Even for $n > 10$ the computational cost of such an operation is prohibitively expensive and sampling from the true distribution is more reasonable.

R^2 is bounded by 1 from above for any model. The proximity of R^2 values calculated from the permuted data or original R^2 values to 1 or to each other can provide insight into goodness of fit of a model. The closer the values of R^2 for the permuted data to 1, the greater the capability of the model to fit to the noise. Close proximity of $q_{1-\alpha}$ to r_0^2 in case of $r_0^2 > q_{1-\alpha}$ and r_0^2 small implies that the model's predictive ability may not be satisfactory even though the null hypothesis is rejected by the test. The test is widely applicable, because of its general form and easily adaptable to different types of models. It also provides an interesting commentary on the predictive abilities of a chosen model. In the event that the quantile $q_{1-\alpha}$ for one model, $\mathcal{F}_1$, significantly exceeds the same quantile for another model, $\mathcal{F}_2$, we conclude that $\mathcal{F}_1$ is either overfitting, indicating a need for

*default $\alpha = 0.05$

**The specific method of prediction of $\hat{Y}_i$ is stated in 5.4.

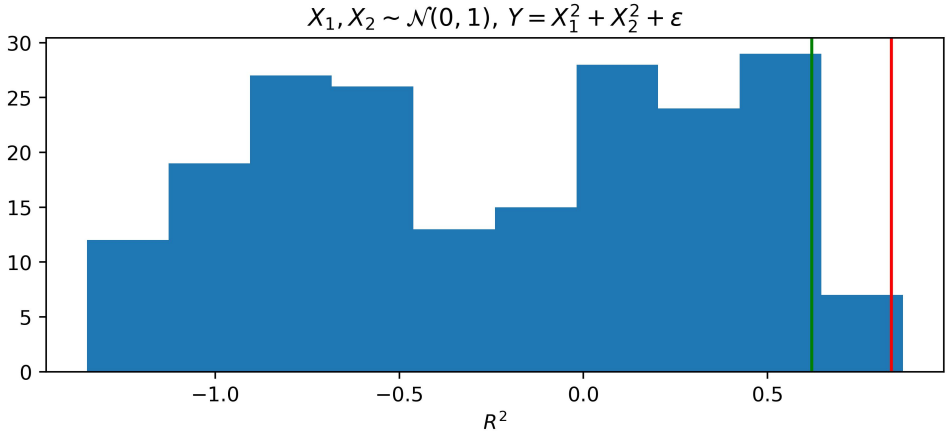


Figure 5.1: Histogram of the distribution of generated R^2 using permutation of y values. The model considered here is a 3-layered neural net. The sample size is 10. The red line denotes the observed R^2 for the true pairings of X and Y, the green line denotes the 95%-quantile of the empirical distribution of R^2 (approximation using 200 permutations).

5

reduction of the set of independent variables or model simplification, or is better able to extract meaningful information from unrelated data.

We close this section with a continuation of the example from section 5.2.1. Let X_1, X_2 be independent standard normal variables and $Y = X_1^2 + X_2^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.01)$. We consider a neural net as a model of choice to predict Y using X_1 and X_2. A random sample of size 10 is drawn. We conduct the permutation test. As seen in figure 5.1, the test rejects the null hypothesis. However, in the case of 2 out of 200 permutations the model achieves higher R^2 than in the case of the original pairings. Even though the model is capable to capture the relationship between X_1, X_2 and Y, there are permutations of the vector of responses that can lead to a better performance of the model.

5.3 Application

5.3.1 Simulation study

We apply our permutation test in multiple scenarios. This section will specifically focus on simulated datasets to assess the test's performance on datasets with varying dependence levels between X and Y and two different class of functions $\mathcal{F}$. An empirical example will be considered in section 5.3.2. In all scenarios we consider the R^2-based test.

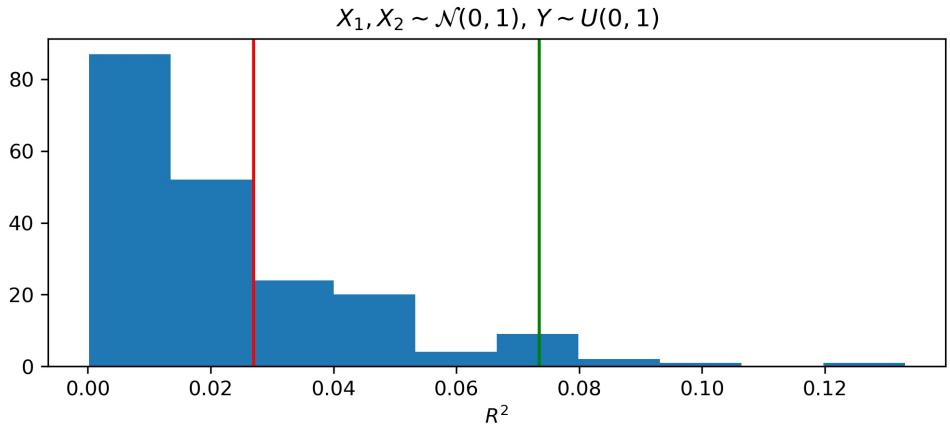
Two different models will be used to fit the data throughout this section. One of them is a linear regression model, which models the relationship between a random vector X and a random variable Y in a linear manner: $Y = \beta \cdot X + \epsilon$. The parameter vector β will always be estimated using the least squares method. Regardless of the length of vector X, the class of functions associated with this model will be referred to as $\mathcal{F}_{\text{LR}}$. The other model we consider is a neural net. A neural net is a collection of neurons arranged into

layers, with neurons from different layers connected to each other. Typically, a neural net consists of an input layer, multiple hidden layers and an output layer. The estimation of neural nets' parameters, the weights associated with neurons and edges between them, is done by feeding multiple training sets of inputs and outputs into the net. Weights are adjusted each time based on a predefined cost function. Class of functions associated with neural nets will be referred to as $\mathcal{F}_{\text{NN}}$ with the number of neurons on each layer specified as a k-tuple, where k refers to the number of layers, e.g. $\mathcal{F}_{\text{NN}}(30,30,30)$ is a neural net with 3 hidden layers, each of which contains 30 neurons.

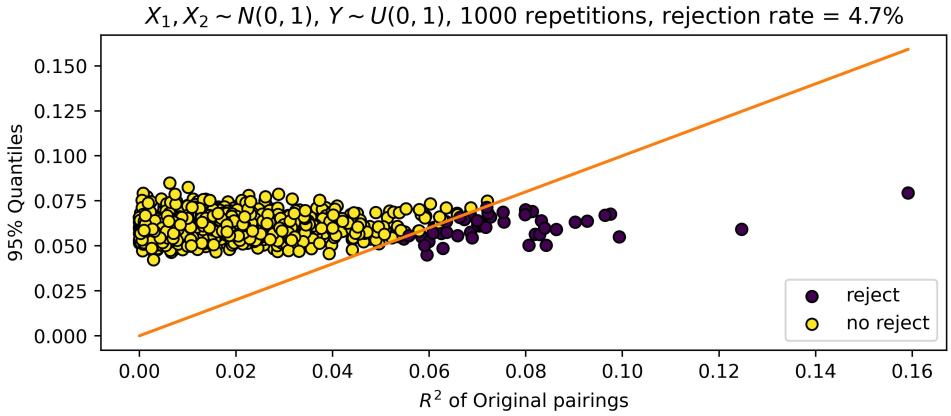
In the first two examples, we will compare the permutation test to two existing methods: Spearman's rank correlation coefficient (also referred to as Spearman's ρ) and Kendall rank correlation coefficient (also referred to as Kendall's τ). Both are statistics used to measure the rank correlation between two variables and both can be used as test statistics in a test for independence of two variables. Since, our examples have more than one explanatory variable, multiple statistics will be given. It is worth noting that both statistics are not applicable when there is no natural ordering in the data, e.g. in the case of functional data when datapoints are functions.

Let $X_1, X_2 \sim \mathcal{N}(0,1)$ and $Y \sim U([0,1])$ be independent random variables. We consider two models and two classes of functions associated with them: $\mathcal{F}_{\text{LR}}$ and $\mathcal{F}_{\text{NN}}(30,30,30)$ and a sample of size 100. In both cases the null hypothesis is not rejected, see fig. 5.2a and 5.3a. We also consider 1000 repetitions of the experiment in the same setup to see the behavior of the test on a larger number of examples. As seen in fig. 5.2b and 5.3b, the null hypothesis is rejected in most repetitions for both models, namely 4.7% for the linear model and 4.5% for the neural net. This shows that the rejection of the null hypothesis can still happen even in case of independence. Most importantly, the rejection rate is close to the confidence level $\alpha = 5\%$. Spearman's ρ test rejects the null hypothesis of independence of X_1 and Y in 5.2% of all cases and rejects the independence of X_2 and Y in 5% of all cases. Kendall's τ test rejects the null hypothesis of independence of X_1 and Y in 5.2% of all cases and rejects the independence of X_2 and Y in 5.3% of all cases. For both of these tests, the rejection rate is also close to the confidence level.

Now, let $X_1 \sim \mathcal{N}(1,1), X_2 \sim \mathcal{N}(0,1)$ be independent and $Y = \log|X_1| + X_2^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0,1)$ is the noise. Consider a sample of size 100. For both $\mathcal{F}_{\text{LR}}$ and $\mathcal{F}_{\text{NN}}(30,30,30)$, the permutation test rejects the null hypothesis, since the values of R^2 for the original pairings are much higher than for any of the permuted pairings. For the behavior of the test in a single example see fig. 5.4a and 5.5a. In this case the neural net outperforms the linear model significantly, thanks to its complexity. Fig. 5.4b and 5.5b show that the rejection rate in this case is quite high when repeating the experiment 1000 times, close to 95% for the linear model and 94% for the neural net. This particular example illustrates the test's applicability in the case of a functional relation between predictors and responses. The model is not just fitting the noise, there is some relation between predictors and responses. It might not be captured well using a linear regression model, but the model is still able to capture more than pure noise. Spearman's ρ test and Kendall's τ test have also been performed in this example, but they show a slight difference from what we see in the case of the permutation test. Spearman's ρ test rejects the null hypothesis of independence of X_1 and Y in 99.6% of all cases and rejects the independence of X_2 and Y in 9.5% of all cases. Similarly, Kendall's τ test rejects the null hypothesis of independence of X_1 and Y

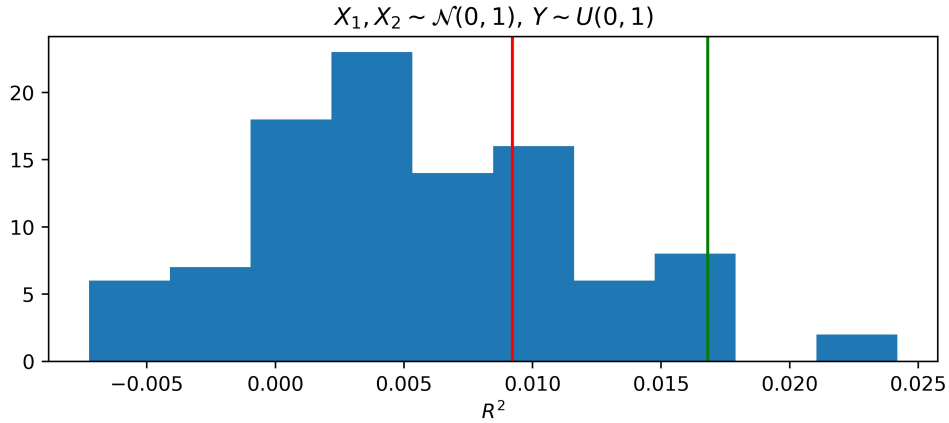


(a) Histogram of the distribution of generated R^2 using permutation of y values. The model considered here is linear regression with the class of functions $\mathcal{F}_{\text{LR}}$. The sample size is 100. The red line denotes the observed R^2 for the true pairings of X and Y, the green line denotes the 95%-quantile of the empirical distribution of R^2 (approximation using 200 permutations).

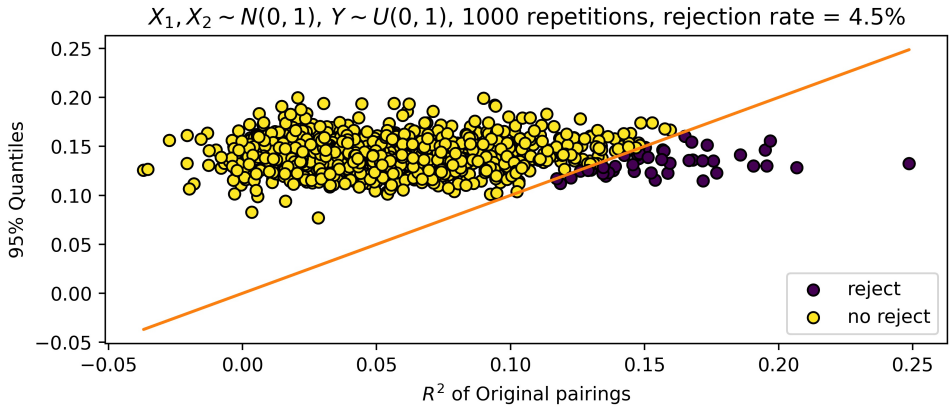


(b) Scatterplot of the R^2 values for the original pairings against the 95% quantiles of the empirical distribution of R^2. The orange line shows the identity function.

Figure 5.2: Results of the permutation test for $\mathcal{F}_{\text{LR}}$ with data generated in a following manner $X_1, X_2 \sim \mathcal{N}(0,1)$ and $Y \sim U([0,1])$.



(a) Histogram of the distribution of generated R^2 using permutation of y values. The model considered here is a 3-layered neural net with the class of functions $\mathcal{F}_{\text{NN}}(30,30,30)$. The sample size is 100. The red line denotes the observed R^2 for the true pairings of X and Y, the green line denotes the 95%-quantile of the empirical distribution of R^2 (approximation using 200 permutations).



(b) Scatterplot of the R^2 values for the original pairings against the 95% quantiles of the empirical distribution of R^2. The orange line shows the identity function.

Figure 5.3: Results of the permutation test for $\mathcal{F}_{\text{NN}(30,30,30)}$ with data generated in a following manner $X_1, X_2 \sim \mathcal{N}(0,1)$ and $Y \sim U([0,1])$.

in 99.7% of all cases and rejects the independence of X_2 and Y in 11.7% of all cases. This shows that the relationship between X_1 and Y is easier to capture than the relationship between X_2 and Y, and with high probability the test will indicate that X_1 and Y are not independent. The relationship between X_2 and Y is not as easy to capture using Kendall's τ or Spearman's ρ, which is to be expected due to the application of a nonlinear function with a minimum at the mean of X_2 when defining Y.

For the remaining scenarios in this section, we consider only the linear regression model with the class of functions $\mathcal{F}_{\text{LR}}$. We inspect the influence of changing the distribution slightly in the test in order to ensure the statistical analysis using the test is reliable and accurate. For $a \in \mathbb{R}$ let $X_1 \sim \mathcal{N}(a,1), X_2 \sim \mathcal{N}(0,0.1)$ be independent and $Y = \log|X_1| + X_2^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0,0.1)$ is the noise. Consider a sample of size 100. Note that the variance of X_2 has been decreased in comparison to the previous example. Only for values of a close to 0, the null hypothesis is not rejected (fig. 5.6a). This makes sense, since the logarithm changes most rapidly close to 0 and for those arguments it is difficult to fit a linear function which describes this relationship well. This pattern is the same with average rejection rate of H_0 when repeating the experiment 100 times for each value of a, see fig. 5.6b. For values of a greater than 0.6, the H_0 is almost never rejected. When the variance of X_2 increases to 0.5, the null hypothesis is no longer rejected for some values of a larger than 5 (fig. 5.7). This particular case shows the influence of available information on rejecting the null hypothesis. The less informative predictors are the more likely it is not to reject the null hypothesis; we can see that as the parameter a increases, the $\log|X_1|$ becomes flatter slowly losing its predictive value. Meanwhile, the influence of X_2^2 on the value of Y increases and given that the model can only predict linearly in X_2, the power of the test decreases.

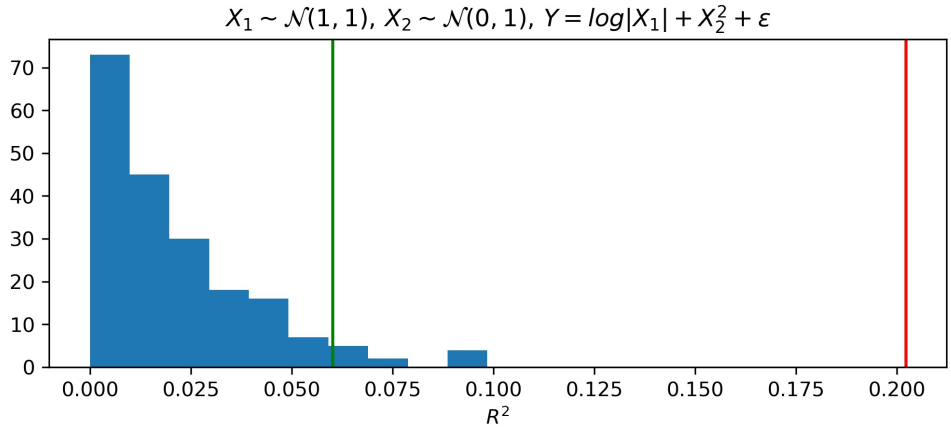
Fig. 5.8 and 5.9 show explicitly the influence of the sample size on the test's capability to reject H_0 for the linear regression model with the class of functions $\mathcal{F}_{\text{LR}}$. In the case when H_0 is true (fig. 5.8), the null hypothesis is rejected at a rate of 2-8% on average regardless of the sample size.[†] In the case when H_0 is false (fig. 5.9), specifically with $Y = \log(X) + \epsilon$ for $\epsilon \sim \mathcal{N}(0,1)$, the null hypothesis is rejected much less for smaller sample sizes and the rejection rate increases as the sample size increases reaching close to 95% at sample size 300. We can conclude that the power of our test increases until the sample size of around 300, at which point the type II error is particularly low. Meanwhile, the rejection of a true null hypothesis is rare, even for the smallest of sample sizes.

Using the bivariate normal distribution with varying correlation, we can empirically detect the point at which the test rejects H_0 for the linear regression model with the class of functions $\mathcal{F}_{\text{LR}}$ as the variables become more and more dependent. Let $0 \leq \rho \leq 1$ and $X, Y \sim N(\mu, \Sigma)$, such that

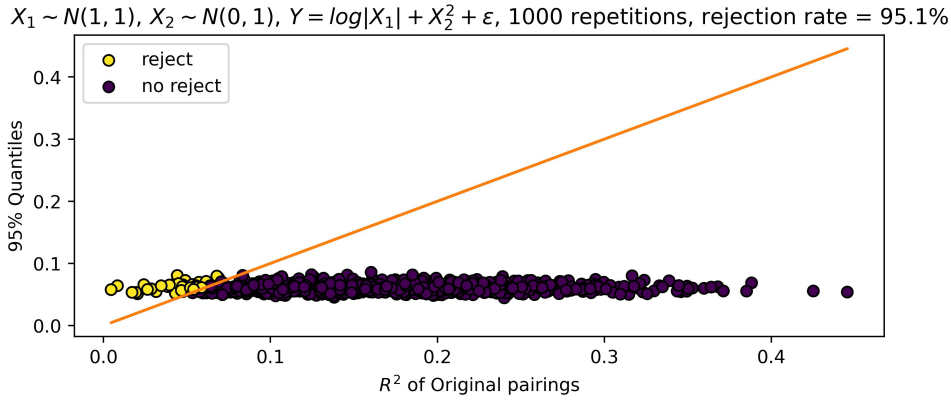
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}.$$

Fig. 5.10 shows that as the correlation reaches 0.3, the test starts to reject H_0 almost always in case of sample size $n = 100$.[‡] We conclude that for sample size $n = 100$, the dependence is only detectable reliably by the test when the correlation between variables is greater than 0.3. This particular example shows that for a given sample size a certain threshold of

[†]Again the figure is showing the error rate for Pitman's test as a function of sample size.
[‡]Note that this figure is showing the error rate for Pitman's test as a function of sample size [107, 108].

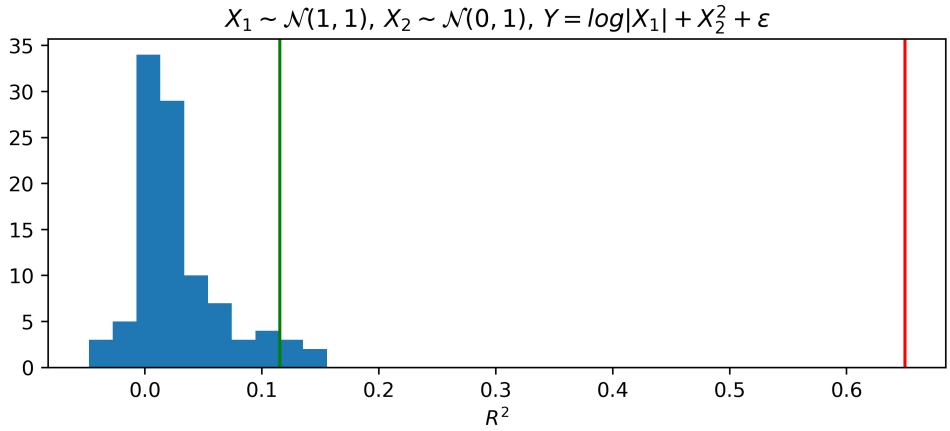


(a) Histogram of the distribution of generated R^2 using permutation of y values. The model considered here is linear regression with the class of functions $\mathcal{F}_{\text{LR}}$. The sample size is 100. The red line denotes the observed R^2 for the true pairings of X and Y, the green line denotes the 95%-quantile of the empirical distribution of R^2 (approximation using 200 permutations).

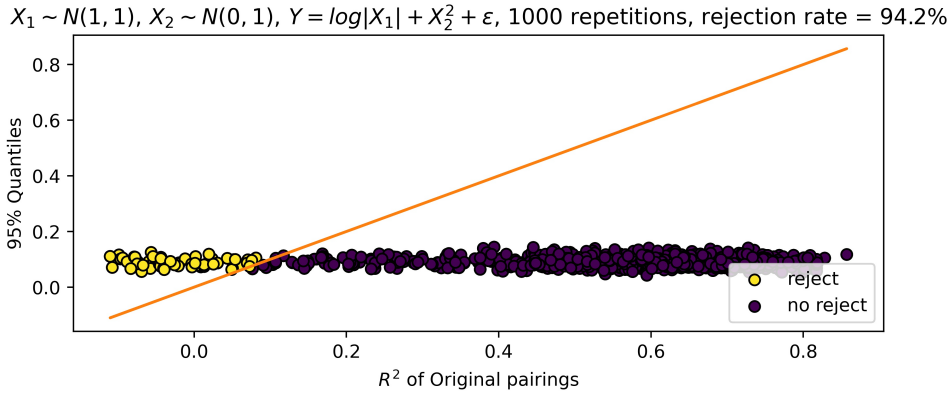


(b) Scatterplot of the R^2 values for the original pairings against the 95% quantiles of the empirical distribution of R^2. The orange line shows the identity function.

Figure 5.4: Results of the permutation test for $\mathcal{F}_{\text{LR}}$ with data generated in a following manner $X_1 \sim \mathcal{N}(1,1), X_2 \sim \mathcal{N}(0,1)$ and $Y = \log|X_1| + X_2^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0,1)$.

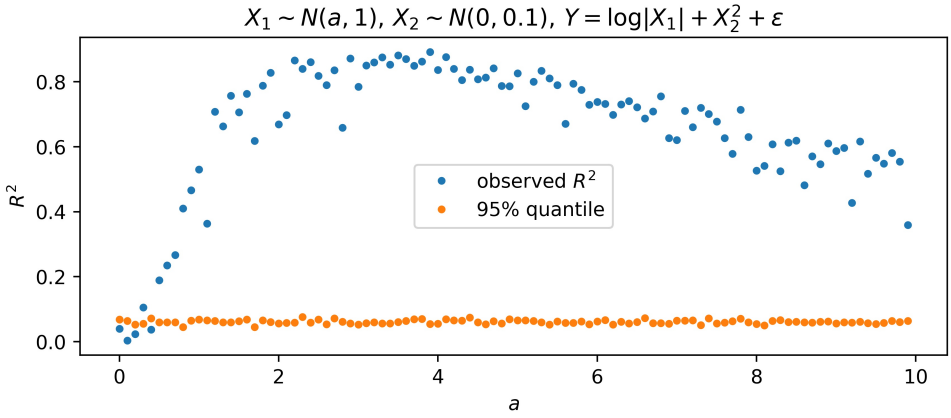


(a) Histogram of the distribution of generated R^2 using permutation of y values. The model considered here is a 3-layered neural net with the class of functions $\mathcal{F}_{\text{NN}}(30,30,30)$. The sample size is 100. The red line denotes the observed R^2 for the true pairings of X and Y, the green line denotes the 95%-quantile of the empirical distribution of R^2 (approximation using 200 permutations).

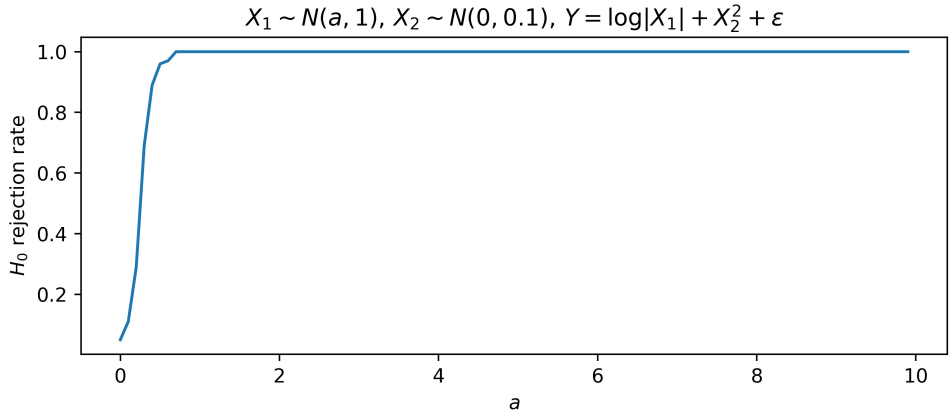


(b) Scatterplot of the R^2 values for the original pairings against the 95% quantiles of the empirical distribution of R^2. The orange line shows the identity function.

Figure 5.5: Results of the permutation test for $\mathcal{F}_{\text{NN}(30,30,30)}$ with data generated in a following manner $X_1 \sim \mathcal{N}(1,1), X_2 \sim \mathcal{N}(0,1)$ and $Y = \log|X_1| + X_2^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0,1)$.

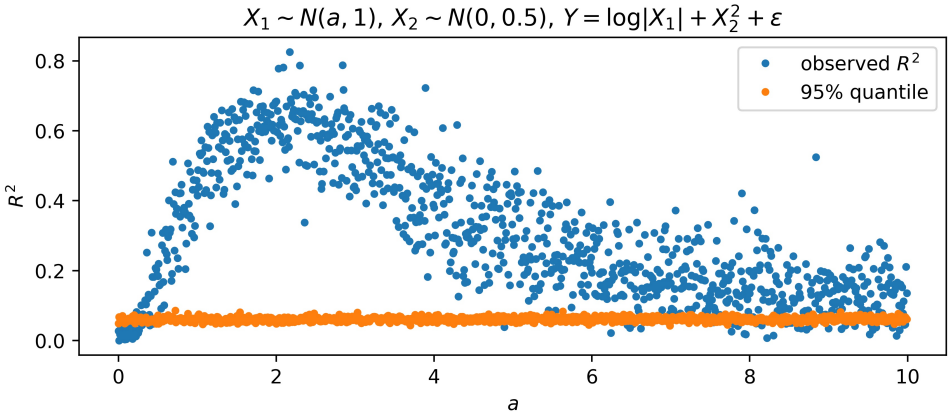


(a) The plot shows the results of performing the permutation test for linear regression model with the class of functions $\mathcal{F}_{\mathrm{LR}}$. The sample size is 100. The blue dots show the observed R^2 and the orange dots show the 95% quantile of the empirical distribution of generated R^2 (approximation using 200 permutations). The test has been performed for values of a ranging between 0 and 10.

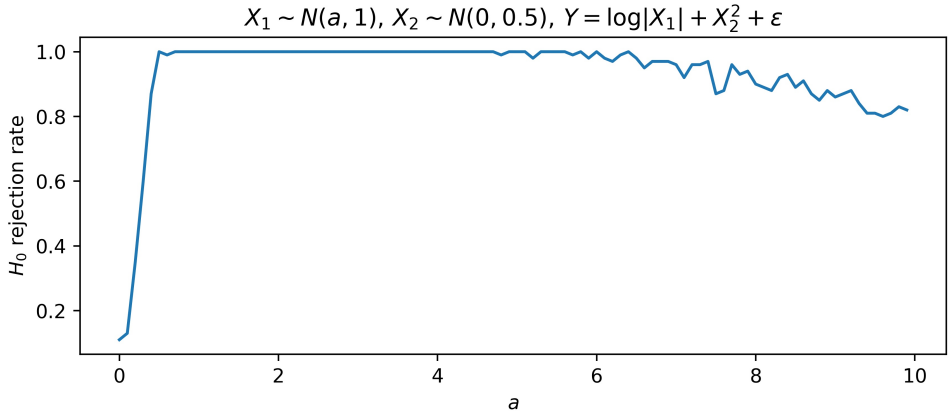


(b) Average rejection rate of H_0 with parameter a varying from 0 to 1. For each a 100 repetitions were made.

Figure 5.6: Results of the permutation test for $\mathcal{F}_{\mathrm{LR}}$ with data generated in a following manner $X_1 \sim \mathcal{N}(a,1), X_2 \sim \mathcal{N}(0,0.1)$ and $Y = \log|X_1| + X_2^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0,0.1)$.

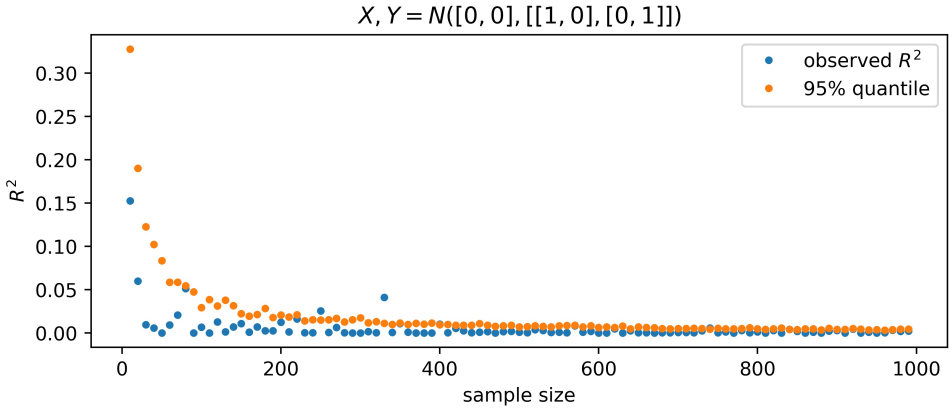


(a) The plot shows the results of performing the permutation test for linear regression model with the class of functions $\mathcal{F}_{\text{LR}}$. The sample size is 100. The blue dots show the observed R^2 and the orange dots show the 95% quantile of the empirical distribution of generated R^2 (approximation using 200 permutations). The test has been performed for values of a ranging between 0 and 10.

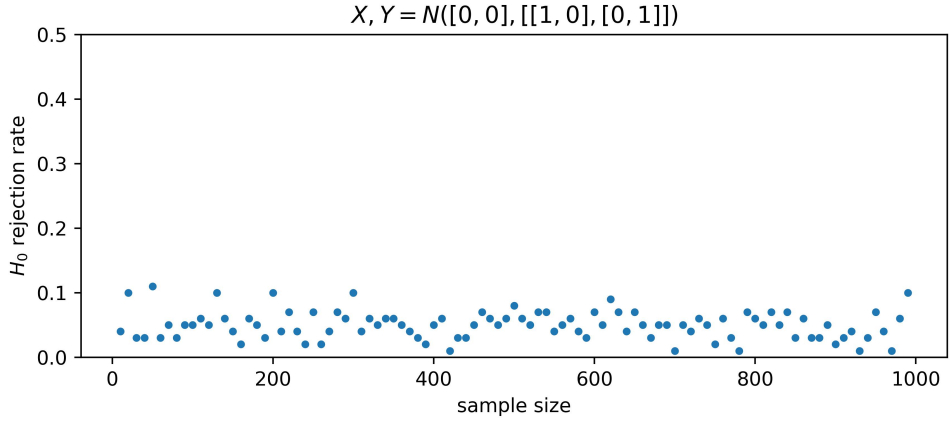


(b) Average rejection rate of H_0 with parameter a varying from 0 to 10. For each a 100 repetitions were made.

Figure 5.7: Results of the permutation test for $\mathcal{F}_{\text{LR}}$ with data generated in a following manner $X_1 \sim \mathcal{N}(a,1), X_2 \sim \mathcal{N}(0,0.5)$ and $Y = \log|X_1| + X_2^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0,0.1)$.

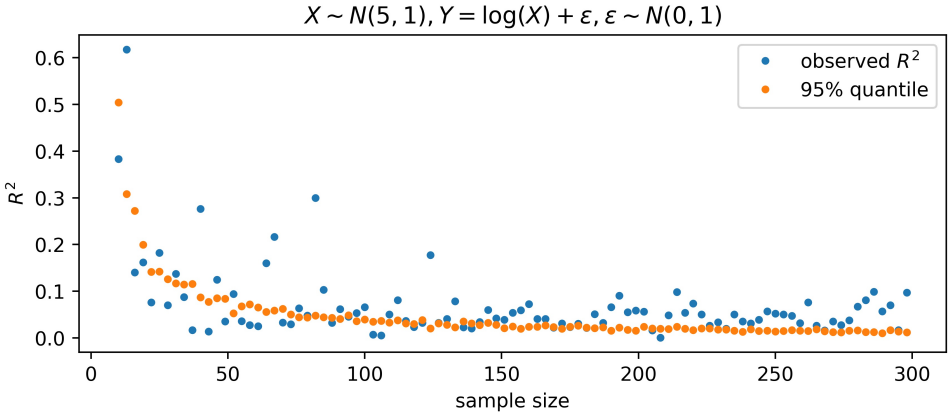


(a) The plot shows the results of performing the permutation test for linear regression model with the class of functions $\mathcal{F}_{\text{LR}}$. The blue dots show the observed R^2 and the orange dots show the 95% quantile of the empirical distribution of generated R^2 (approximation using 200 permutations). The test has been performed for sample sizes ranging between 10 and 1000.

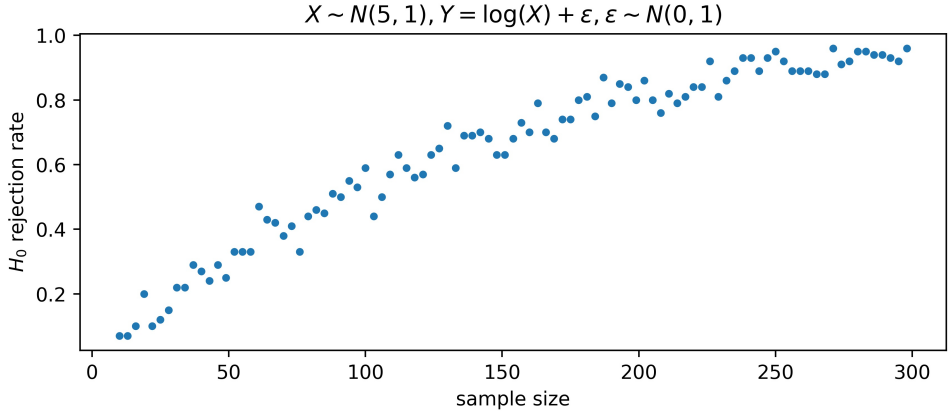


(b) Average rejection rate of H_0 with sample size varying from 10 to 1000. For each sample size 100 repetitions were made.

Figure 5.8: Results of the permutation test for $\mathcal{F}_{\text{LR}}$ with data generated in a following manner $X, Y \sim N(\mu, \Sigma)$, $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.



(a) The plot shows the results of performing the permutation test for linear regression model with the class of functions $\mathcal{F}_{\text{LR}}$. The blue dots show the observed R^2 and the orange dots show the 95% quantile of the empirical distribution of generated R^2 (approximation using 200 permutations). The test has been performed for sample sizes ranging between 10 and 300.



(b) Average rejection rate of H_0 with sample size varying from 10 to 300. For each sample size 100 repetitions were made.

Figure 5.9: Results of the permutation test for $\mathcal{F}_{\text{LR}}$ with data generated in a following manner $X \sim \mathcal{N}(5,1)$ and $Y = \log|X| + \epsilon$, where $\epsilon \sim \mathcal{N}(0,1)$.

correlation exists at which the test starts to reject the null hypothesis. As the correlation increases the rejection becomes more and more likely for a given sample size.

Lastly, we present a comparison of our permutation test with a permutation test found in [104]. This is also a test for no effect, but specifically in the linear regression model. Its formulation requires a sample of n i.i.d. observations $\{(X_1,Y_1),\dots,(X_n,Y_n)\}$ from a bivariate variable (X,Y). We assume that the variables are linked by a linear regression $\mathbb{E}(Y|X=x)=\alpha+\beta\cdot x$, where $\alpha,\beta\in\mathbb{R}$. The null hypothesis considered for this test is $\beta=0$, under the assumption that responses Y_i can be permuted with respect to covariate X. The test statistic is $T^*_\beta=\sum_i X_iY_i$ and the permutation of Y_i is used when approximating the distribution of the test statistic under H_0. We refer to this test as the permutation test for linear regression after the naming convention in [104]. Note that in practice the only difference between our approaches is the choice of test statistic. In their case, the choice of the test statistic is driven by the null hypothesis. In our test, the test statistic can be chosen freely as long as it can be calculated using the sample $\{(f(x_i),y_i)\}_i$, which technically means we could use T^*_β as the test statistic. In that sense, we can view our test as the generalization of the test for linear regression.

We continue using bivariate normal variables X and Y. We compare the average rejection rate of H_0 for both tests with parameter ρ varying from 0 to 1. Fig. 5.11 shows the comparison between the tests. For sample size $n=100$, the permutation test for linear regression detects the dependence for a slightly smaller ρ than our permutation test, but both reach the rejection rate of 1 at $\rho\approx 0.4$. We conclude that a context-specific test statistic, in this case T^*_β, outperforms more general statistic. At the same time, our test can use T^*_β as the test statistic.

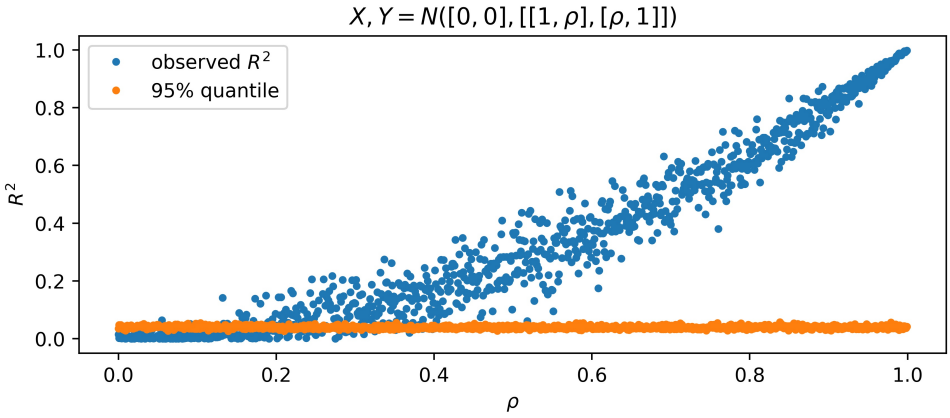
5.3.2 Tennis serve dataset

This section concerns an application of the permutation test to a tennis serve dataset. Seven professional athletes wearing inertial measurement units (IMUs) performed tennis serves. Each athlete followed a protocol of first and second serves. Sensors were placed on 4 body parts: lower and upper arms, trunk and pelvis as can be seen in fig. 5.12. Each IMU contained a triaxial accelerometer and triaxial gyroscope. The data consists of 7 uninterrupted time series of 24-dimensional data (4 body parts × 2 types of sensors × 3 axes). The dataset is further described in the Master thesis ([2]).

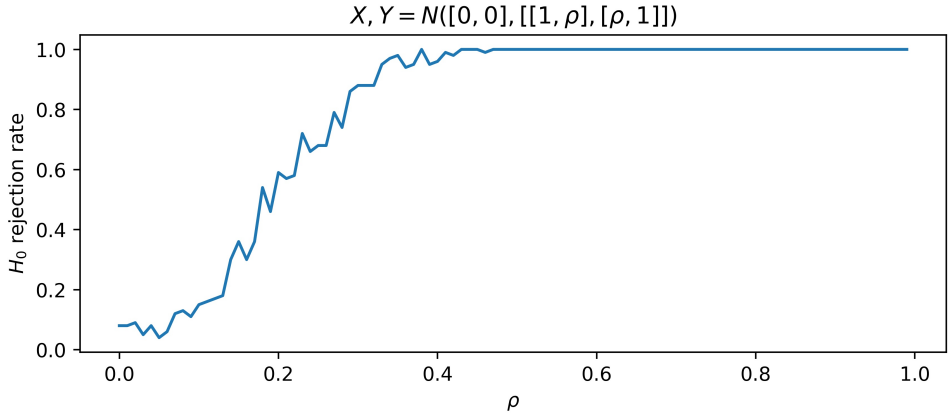
Additionally, a dataset containing personal characteristics of the players and performance characteristics of each serve has been included. The personal characteristics are the sex, age, height and weight of the players. The performance characteristics are the ball velocity, an indication of whether the ball went in or out and the velocity-accuracy index (VA index). The VA index for a single serve was introduced and motivated by [109] and is defined as follows:

$$\text{VA index}=\frac{(\text{ball velocity (kph)})^2}{100}\times\frac{\text{achieved points}}{9}, \tag{5.7}$$

where achieved points refer to the number of points assigned to a serve based on its closeness to a target area on the court (see fig. 5.13). The number of points assigned to a serve is based on a new Serve Tennis Test (STT) adapted from [109]. Originally, the point system was devised based on the ellipses in the serve box where aces were hit in male tennis



(a) The plot shows the results of performing the permutation test for linear regression model with the class of functions $\mathcal{F}_{\text{LR}}$. The sample size is 100. The blue dots show the observed R^2 and the orange dots show the 95% quantile of the empirical distribution of generated R^2 (approximation using 200 permutations). The test has been performed for values of correlation ρ ranging between 0 and 1.



(b) Average rejection rate of H_0 with parameter ρ varying from 0 to 1. For each ρ 100 repetitions were made.

Figure 5.10: Results of the permutation test for $\mathcal{F}_{\text{LR}}$ with data generated in a following manner $X,Y \sim N(\mu,\Sigma)$, $\mu = \begin{bmatrix}0\\0\end{bmatrix}$ and $\Sigma = \begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}$.

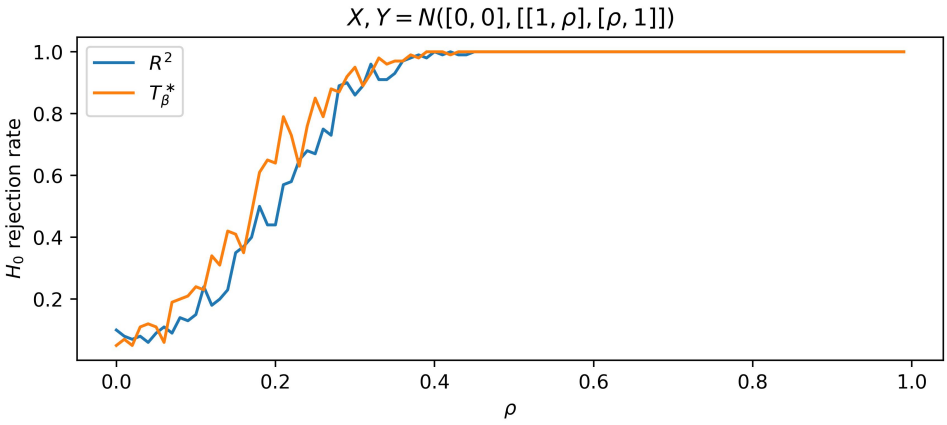


Figure 5.11: Average rejection rate of H_0 with parameter ρ varying from 0 to 1. For each ρ 100 repetitions were made. Two different tests were considered. Blue line was generated using R^2 as the test statistic, while the orange line was generated using T^*_β as the test statistic.

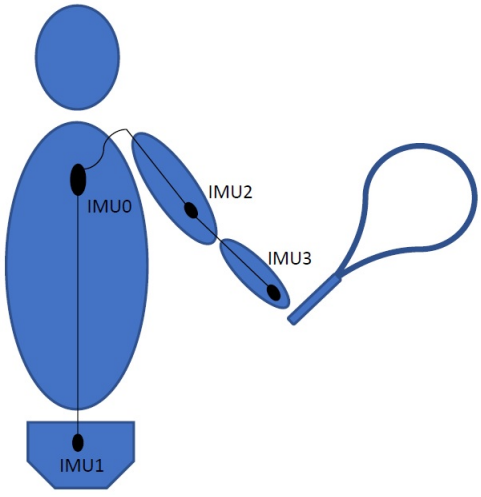


Figure 5.12: Segment model of right-handed player and racquet (back view, frontal plane).

matches during the Australian Open ([110]). However, the system has been improved upon since then. The points are discrete. Nine points are given for hitting the center of the target area. Six and three points are assigned for areas further from the center. One point is assigned for a ball much further from the target area, but still a valid ball, while zero points are given to a serve which did go out. Each participant performed approximately 48 serves. In total, 29.6% of serves were faults (and as a result had a VA-index 0).

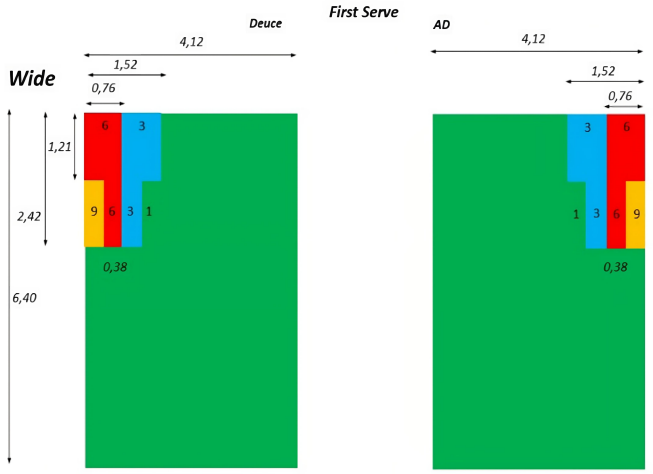
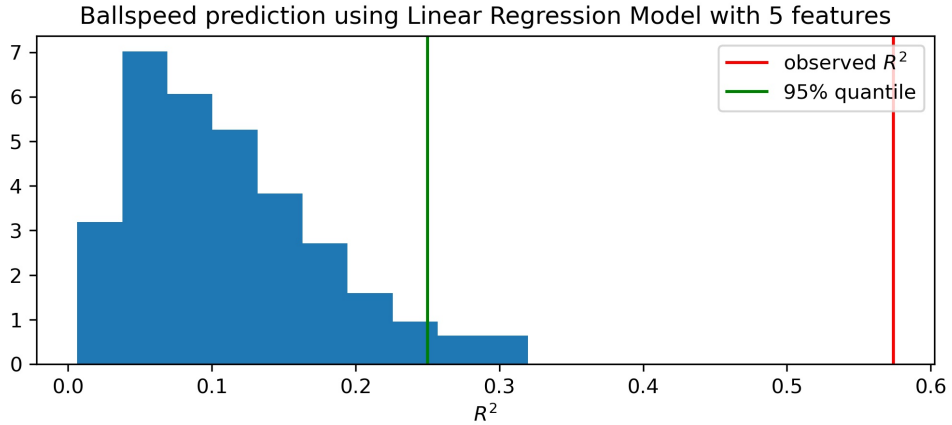


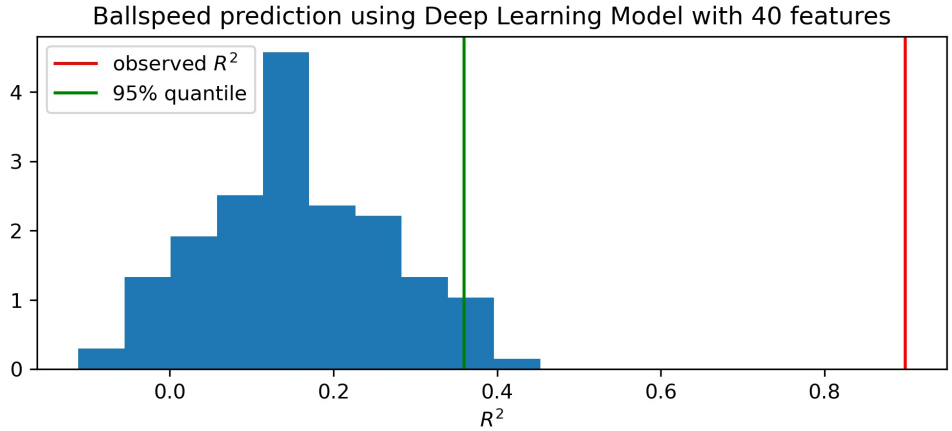
Figure 5.13: Target areas for the tennis serve. The scenario considered here is a serve in the wide direction. The points given on each target area correspond to the number of accuracy points needed to calculate the VA index of the serve.

We will use the tennis serve dataset in order to demonstrate an application of the permutation test to real life data. We will focus on the prediction of ball speed and VA-index prediction. The functional predictors have been transformed into vectors, using a Fourier basis representation, in order to be able to use the linear regression model with the class of functions $\mathcal{F}_{\text{LR}}$ and the neural net with the class of functions $\mathcal{F}_{\text{NN}}(300, 300, 300)$. The choice to use Fourier coefficients as predictors was the most natural way of incorporating information from the time series. First, a prediction of ball speed was considered. The permutation test rejected the null hypothesis in cases of both models as seen in fig. 5.14a and 5.14b. The test rejects the null hypothesis for both models, although higher values of R^2 achieved by the neural net for the original pairings suggest greater capabilities of that model to detect the dependence.

In the case of prediction of the VA-index as defined in (5.7), the permutation test did not reject the null hypothesis for the linear regression model with the class of functions $\mathcal{F}_{\text{LR}}$ as well as for the neural net model with the class of functions $\mathcal{F}_{\text{NN}}(300, 300, 300)$. Fig. 5.15a shows results for the linear regression model and fig. 5.15b shows results for the neural net. The values of R^2 are quite low for both models and for many permutations of y-values the generated R^2 is much higher than the observed R^2 for the true pairings. These results convince us that a good prediction using the linear regression model or the neural net model is not possible at the moment. The issue may lie with the current size of the dataset



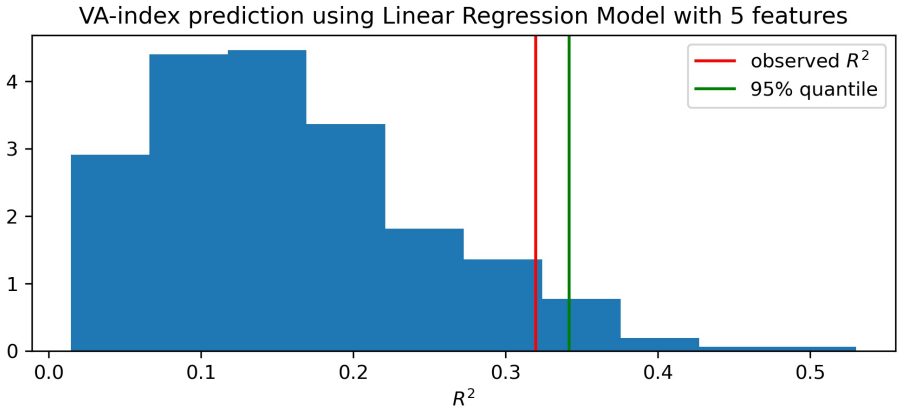
(a) Histogram of the distribution of generated R^2 using permutation of y values. The sample size is 46. The red line denotes the observed R^2 for the true pairings of X and Y, the green line denotes the 95%-quantile of the empirical distribution of R^2 (approximation using 200 permutations).



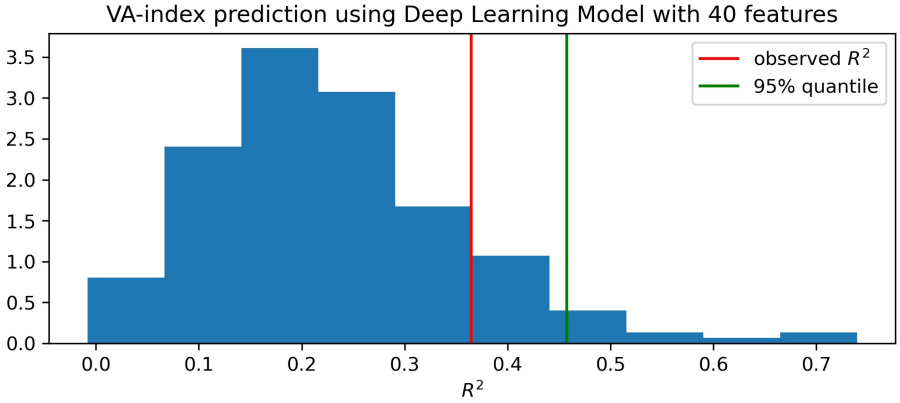
(b) Histogram of the distribution of generated R^2 using permutation of y values. The sample size is 46. The red line denotes the observed R^2 for the true pairings of X and Y, the green line denotes the 95%-quantile of the empirical distribution of R^2 (approximation using 200 permutations).

Figure 5.14: Results of the permutation test for the ball speed prediction using $\mathcal{F}_{\text{LR}}$ and $\mathcal{F}_{\text{NN}}(300,300,300)$.

or the number of serves per player or simply because the relation as can be described by the neural net is not strong. The fact that the number of Fourier coefficients used in this prediction was increased to achieve more favourable R^2 for the original pairings of (x_i, y_i) (at least in the case of the deep learning model), shows how complex this task is and additional information is needed in the data to increase the R^2.



(a) Histogram of the distribution of generated R^2 using permutation of y values. The sample size is 34. The red line denotes the observed R^2 for the true pairings of X and Y, the green line denotes the 95%-quantile of the empirical distribution of R^2 (approximation using 200 permutations).



(b) Histogram of the distribution of generated R^2 using permutation of y values. The sample size is 34. The red line denotes the observed R^2 for the true pairings of X and Y, the green line denotes the 95%-quantile of the empirical distribution of R^2 (approximation using 200 permutations).

Figure 5.15: Results of the permutation test for the VA index prediction using $\mathcal{F}_{\text{LR}}$ and $\mathcal{F}_{\text{NN}}(300, 300, 300)$.

5.4 Conclusion and discussion

This paper concerns the theoretical foundations and the application of the permutation approach for testing whether a model can capture dependence structure between predictors and responses. The test is a tool to determine whether a model is able to fit the data better than pure noise. We are mostly interested whether X has any effect on Y and we pursue that interest with the help of a chosen, fixed model. The null hypothesis is formulated in terms of independence of Y and $(f(X))_{f\in\mathcal{F}}$ and in this form cannot be found in previous literature. Proposition 5.2.1 allows us to consider the test as a permutation test formally and proposition 5.2.2 allows us to consider R^2 as a test statistic. This approach is data-centered and the results of the test depend on just one model without the need to directly compare between different models. We also do not require sample splitting thus the test can rely on the power of the whole sample size, which can be vital in datasets of smaller size. Our findings are supported through an application to the tennis serve dataset. In this case, it gave evidence that a seemingly well-fitting model is not necessarily trustworthy. The prediction is either not possible with the given sensor data and model or a larger sample size is needed to predict the VA-index more accurately.

6

Conclusion

Sports are changing rapidly thanks to technology, which revolutionizes the way in which the sports are played and watched. In tennis, the Hawkeye system has reduced the number of errors made by the umpires. In football, the introduction of the goal-line technology ensures fair judgment on the position of the ball with respect to the goal line. More than that, athletes use technological advancements in order to train better and avoid major injuries.

The main focus of this thesis was to utilize sensor technology to extract meaningful insights from sensor data in football and tennis. New methods and procedures inspired by problems arising during the investigation have been introduced when necessary. The procedures used and methods developed for this project had to serve a very specific purpose in the application. In many ways, this thesis would not have been possible if not for collaborations within the Citius Altius Sanius (CAS) program. The CAS program itself aimed to use sensor technology to address two specific problems: injury prevention and performance improvement. Developments accomplished within this thesis relied on the data provided by other parties within CAS and the funds provided by the sponsors of CAS.

Two main goals have been set out to be accomplished within this thesis. First, to provide an accurate and state-of-the-art football activity recognition model. Second, to predict two tennis serve performance metrics, namely ball speed and the VA index using the sensor data. The goal of delivering an activity recognition model in football has been accomplished and the results can be found in chapter 2. Chapters 3 and 4 present developments made while developing activity recognition models that are relevant to the topic and can be extended to other sports as well. The results of predicting the serve performance metrics in tennis can be seen in chapter 5, where a permutation test has been developed to determine the capability of our model to predict the performance metrics. The following paragraphs provide short summaries of each of the main results of this thesis.

Deep learning models were used to accurately recognize football-specific activities based on sensor measurements. Multiple neural network architectures have been considered and tested. Most of them consisted in some combination of different types of convolutional layers and recurrent layers. Many of the models were capable of achieving high accuracies and faster evaluation times compared to the traditional machine learning algorithms. The

proposed methodology demonstrates the potential for widespread application in other sports and activities as well.

A novel post-processing scheme was developed to improve the prediction of football activity recognition models, particularly in the presence of unrealistically short or misclassified activities. The method significantly enhanced the performance of classifiers, offering a practical solution for refining activity recognition outcomes in real-life scenarios. Not unlike the deep learning models, the post-processing scheme may be used for a variety of human activity recognition tasks. Additionally, novel quality measures were introduced to allow greater customization and the integration of domain knowledge. The measures were specifically designed for use with predictive models in activity recognition problems.

The prototype wearable sensor trousers used for collection of sensor data cover both the upper as well as the lower legs of the athletes. This is not typical for the professional football players. For the sake of their comfort, and additionally due to cost issues, a shorter version of the sensor trousers that covers only the upper legs can be produced instead. It is very important to consider the impact of the missing sensor data on predictive models. Given the existence of the large dataset produced using longer trousers, it is worth considering leveraging those data to produce a model to impute the missing sensor data using only the upper legs sensors. We considered the data of only one player. There are various approaches for handling missing data. We consider methods that utilize the functional nature of the sensor data. Using functional PCA scores and functional regression models led to promising results in recovery of the missing sensor signals. These methods showcased the feasibility of maintaining predictive model performance despite the removal of sensors from the dataset.

To predict the tennis serve performance metrics using sensor data, we considered two models: a linear regression model and a deep learning model. The results of predicting ball speed were satisfactory, but the prediction of the velocity-accuracy index (VA index) proved more challenging. A permutation test was considered to evaluate the models performance. The test has shown that the models we have considered were not able to capture the dependence structures between predictors and responses in this particular case. This data-driven approach highlighted the importance of robust model evaluation, particularly in scenarios where traditional performance metric like R^2 may be misleading. A contribution was also provided with regards to the permutation tests: a new formulation of the null hypothesis. The new null hypothesis has been defined in terms of independence and its relation to the permutation test has also been proved in the thesis.

Moving forward, there are several options for future research and applications based on the findings of this thesis:

- Continued refinement of deep learning architectures and functional data analysis techniques to enhance the accuracy and robustness of predictive models for activity recognition and prediction of various metrics related to the performance of the athlete, such as the VA index in tennis.

- Use of the activity recognition model in other sports and generic activities unrelated to sports are possible due to adaptability of the model and the methods developed in this thesis. For this to be possible a labeled dataset needs to be provided, since the deep learning model requires this input for training. Additionally, the methods

developed here were specifically tailored to sensor data and might not work as well with different types of data.

- Further investigation into the impact of removing certain sensors from the trousers, especially on previously unseen data, produced using different training drills or mock matches and produced by different players.
- Direct comparison of the effect of using sensor data versus video recordings on the accuracy of the activity recognition model.
- Application of the activity recognition model for data collected using smartphones and other sensors that are more widely available.

Lastly, it is worth addressing the issue of injury prevention, as it is one of the two main problems which CAS aimed to solve. The problem of activity recognition, extensively considered in this thesis, does not directly translate into solutions for injury prevention. Injury data is difficult to obtain, since it would require an extensive collection process. Additionally, injuries occur infrequently, so even a large case study may not yield a large amount of injury data. This necessitates the exploration of alternative methods for preventing injuries. One way would be to combine the player feedback with the activity recognition methods and recognize strenuous exercises. By collecting questionnaires filled out by athletes, one could gauge the level of intensity of the exercise. This would still require setting a personalized threshold for high injury risk activities. Alternatively, injury data can be collected directly from the medical records to identify patterns and risk factors. This can be further supported by data from physiotherapy, where data of the recovering athletes can serve as a proxy for data about real injuries. Another approach would be to calculate the load on specific muscles using sensors. This is a more autonomous approach that does not require subjective feedback from the athlete and allows for determining the intensity of the exercise.

To conclude, this thesis applied machine learning techniques and statistical theory at large to sensor data within the context of sports. Developments presented here range from purely applied projects, as seen in chapters 2 and 4 to a mix of application and theory, as seen in chapters 3 and 5. Two main problems were addressed in this thesis: the activity recognition in football and the prediction of serve performance metrics in tennis. Some additional methods were also proposed, such as the post-processing scheme or the novel quality measures for activity recognition problems. The methods are not constrained to the applications considered here and can be considered for other data sources or sports.

Bibliography

References

[1] Annemarijn Steijlen, Jeroen Bastemeijer, Linda Plaude, Paddy French, Andre Bossche, and Kaspar Jansen. Development of sensor tights with integrated inertial measurement units for injury prevention in football. In *Proceedings of the 6th International Conference on Design4Health*, volume 2, pages 219–228, Amsterdam, The Netherlands, 2020. D4H.

[2] Erik Faneker. The kinetic chain and serve performance in elite tennis players (Master Research Project). Master's thesis, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, 2021.

[3] Rafael Cuperman, Kaspar M.B. Jansen, and Michał G. Ciszewski. An end-to-end deep learning pipeline for football activity recognition based on wearable acceleration sensors. *Sensors*, 22:1347, 2022. https://doi.org/10.3390/s22041347.

[4] Moaed A. Abd, Rudy Paul, Aparna Aravelli, Ou Bai, Leonel Lagos, Maohua Lin, and Erik D. Engeberg. Hierarchical tactile sensation integration from prosthetic fingertips enables multi-texture surface recognition. *Sensors*, 21:4324, 2021. https://doi.org/10.3390/s21134324.

[5] Salwa O. Slim, Ayman Atia, Marwa M.A. Elfattah, and Mostafa-Sami M. Mostafa. Survey on human activity recognition based on acceleration data. *Intl. J. Adv. Comput. Sci. Appl.*, 10, 2019. https://doi.org/10.14569/IJACSA.2019.0100311.

[6] LuKun Wang and RuYue Liu. Human activity recognition based on wearable sensor using hierarchical deep LSTM networks. *Circuits Syst. Signal Process.*, 39:837–856, 2020. https://doi.org/10.1007/s00034-019-01116-y.

[7] Rimas Adaskevicius. Method for recognition of the physical activity of human being using a wearable accelerometer. *Elektronika ir Elektrochnika*, 20:127–131, 2014. https://doi.org/10.5755/j01.eee.20.5.7113.

[8] Andrea Mannini and Angelo Maria Sabatini. Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10:1154–1175, 2010. https://doi.org/10.3390/s100201154.

[9] Sanne I. de Vries, Marjolein Engels, and Francisca Galindo Garre. Identification of children's activity type with accelerometer-based neural networks. *Med Sci Sports Exerc.*, 43:1994–9, 2011. https://doi.org/10.1249/MSS.0b013e318219d939.

[10] Andrey Ignatov. Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. *Appl. Soft Comput.*, 62:915–922, 2018. https://doi.org/10.1016/j.asoc.2017.09.027.

[11] Sojeong Ha, Jeong-Min Yun, and Seungjin Choi. Multi-modal Convolutional Neural Networks for activity recognition. In *Proc. of the 2015 IEEE Int. Conf. on Syst. Man. Cybern.*, pages 3017–3022, Hong Kong, China, 2015. IEEE.

[12] Tahmina Zebin, Patricia J. Scully, and Krikor B. Ozanyan. Evaluation of supervised classification algorithms for human activity recognition with inertial sensors. In *2017 IEEE Sensors*, page 1–3, Glasgow, UK, 2017. IEEE.

[13] Peter Blank, Julian Hoßbach, and Dominik Schuldhaus. Sensor-based stroke detection and stroke type classification in table tennis. In *ISWC '15: Proceedings of the 2015 ACM International Symposium on Wearable*, pages 93–100, Osaka, Japan, 2015. ACM.

[14] Damien Connaghan, Phillip Kelly, Noel E. O'Connor, Mark Gaffney, Michael Welsh, and Cian O'Mathuna. Multi-sensor classification of tennis strokes. In *IEEE Sensors*, pages 1437–1440, Limerick, Ireland, 2011. IEEE.

[15] Benjamin Groh, Thomas Kautz, Dominik Schuldhaus, and Bjoern M. Eskofier. IMU-based trick classification in skateboarding. In *Proc. of the KDD Workshop on Large-Scale Sports Analytics*, Sydney, Australia, 2015. ACM.

[16] Thomas Kautz, Benjamin H. Groh, Julius Hannink, Ulf Jensen, Holger Strubberg, and Bjoern M. Eskofier. Activity recognition in beach volleyball using a Deep Convolutional Neural Network. *Data Min. Knowl. Discov.*, 31:1678–1705, 2017. https://doi.org/10.1007/s10618-017-0495-0.

[17] Dominik Schuldhaus, Constantin Zwick, Harald Körger, , Eva Dorschky, Robert Kirk, and Bjoern M. Eskofier. Inertial sensor-based approach for shot/pass classification during a soccer match. In *Proc. of the KDD Workshop on Large-Scale Sports Analytics*, Sydney, Australia, 2015. ACM.

[18] Xinyu Liu. Tennis stroke recognition: Stroke classification using inertial measuring unit and machine learning algorithm in tennis. Master's thesis, Technische Universiteit Delft, Delft, The Netherlands, 2020.

[19] Libin Jiao, Rongfang Bie, Hao Wu, Yu Wei, Jixin Ma, Anton Umek, and Anton Kos. Golf swing classification with multiple deep convolutional neural networks. *Int. J. Distrib. Sens. Netw.*, 14, 2018. https://doi.org/10.1177/155014771880218.

[20] Dominik Schuldhaus. *Human activity recognition in daily life and sports using inertial sensors.* PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2019.

[21] Cheng Xu, Duo Chai, Jie He, Xiaotong Zhang, and Shihong Duan. InnoHAR: A deep neural network for complex human activity recognition. *IEEE Access*, 7:9893–9902, 2019. https://doi.org/10.1109/ACCESS.2018.2890675.

[22] Kun Xia, Jianguang Huang, and Hanyu Wang. LSTM-CNN architecture for human activity recognition. *IEEE Access*, 8:56855–56866, 2020. https://doi.org/10.1109/ACCESS.2020.2982225.

[23] Mingqi Lv, Wei Xu, and Tieming Chen. A hybrid deep convolutional and recurrent neural network for complex activity recognition using multimodal sensors. *Neurocomputing*, 362:33–40, 2019. https://doi.org/10.1016/j.neucom.2019.06.051.

[24] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16:115, 2016. https://doi.org/10.3390/s16010115.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[26] David H. Hubel and Torsten N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.*, 160:106–154, 1962. https://doi.org/10.1113/jphysiol.1962.sp006837.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9:1735–1780, 1997. https://doi.org/10.1007/s10739-006-9119-z.

[28] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1310–1318, Atlanta, GA, USA, 2013. PMLR.

[29] Erik Wilmes, Cornelis J. de Ruiter, Bram J. C. Bastiaansen, Jasper F. J. A. van Zon, Riemer J. K. Vegter, Michel S. Brink, Edwin A. Goedhart, Koen A. P. M. Lemmink, and Geert J. P. Savelsbergh. Inertial sensor-based motion tracking in football with movement intensity quantification. *Sens.*, 20:2527, 2020. https://doi.org/10.3390/s20092527.

[30] Erik Wilmes. Measuring changes in hamstring contractile strength and lower body sprinting kinematics during a simulated soccer match. Master's thesis, Technische Universiteit Delft, Delft, The Netherlands, 2019.

[31] Daniel Berrar. *Cross-Validation*, pages 542–545. Elsevier, Amsterdam, The Netherlands, 2019.

[32] Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization. https://doi.org/10.48550/arXiv.1412.6980, 2015. arXiv.

[33] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Efficient backprop*, pages 9–48. Springer, Berlin/Heidelberg, Germany, 2012.

[34] 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 José del R. Millàn. Collecting complex activity datasets in highly rich networked sensor environments. In *Proceedings of the 2010 Seventh International Conference on Networked Sensing Systems (INSS)*, page 233–240, Kassel, Germany, 2010. IEEE.

[35] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explor. Newsl.*, 12:74–82, 2011. https://doi.org/10.1145/1964897.1964918.

[36] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 437–442, Bruges, Belgium, 2013. ESANN.

[37] Michał Ciszewski, Jakob Söhl, and Geurt Jongbloed. Improving state estimation through projection post-processing for activity recognition with application to football. *Stat Methods Appl*, 32:1509–1538, 2023. https://doi.org/10.1007/s10260-023-00696-z.

[38] Wesllen S. Lima, Eduardo Souto, Thiago Rocha, Richard W. Pazzi, and Ferry Pramudianto. User activity recognition for energy saving in smart home environment. In *Proc. of the 2015 IEEE Symp. on Comput. and Commun. (ISCC)*, pages 751–757, New York, NY, 2015. IEEE.

[39] Markus Eckelt, Franziska Mally, and Angelika Brunner. Use of acceleration sensors in archery. *Proc.*, 49:98, 2020. https://doi.org/10.3390/proceedings2020049098.

[40] Maurits Waterbolk, Jasper Tump, Rianne Klaver, Rosalie van der Woude, Daniel Velleman, Joost Zuidema, Thomas Koch, and Elenna Dugundji. Detection of ships at mooring dolphins with Hidden Markov Models. *Transp. Res. Rec.*, 2673:0361198119837495, 2019. https://doi.org/10.1177/0361198119837495.

[41] R. Varatharajan, Gunasekaran Manogaran, M. K. Priyan, and Revathi Sundarasekar. Wearable sensor devices for early detection of alzheimer disease using dynamic time warping algorithm. *Clust. Comput.*, 21:681–690, 2018. https://doi.org/10.1007/s10586-017-0977-2.

[42] Agata Kołakowska, Wioleta Szwoch, and Mariusz Szwoch. A review of emotion recognition methods based on data acquired via smartphone sensors. *Sens.*, 20:6367, 2020. https://doi.org/10.3390/s20216367.

[43] Oscar D. Lara and Miguel A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Commun. Surv. & Tutor.*, 15:1192–1209, 2013. https://doi.org/10.1109/SURV.2012.110112.00192.

[44] L. Minh Dang, Kyungbok Min, Hanxiang Wang, Md. Jalil Piran, Cheol Hee Lee, and Hyeonjoon Moon. Sensor-based and vision-based human activity recognition: A

comprehensive survey. *Pattern Recognit.*, 108:107561, 2020. https://doi.org/10.1016/j.patcog.2020.107561.

[45] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognit. Lett.*, 119:3–11, 2019. https://doi.org/10.1016/j.patrec.2018.02.010.

[46] Charissa Ann Ronao and Sung-Bae Cho. Recognizing human activities from smartphone sensors using hierarchical continuous hidden markov models. *Int. J. of Distrib. Sens. Netw.*, 13:1550147716683687, 2017. https://doi.org/10.1177/1550147716683687.

[47] Nicole A. Capela, Edward D. Lemaire, and Natalie Baddour. Feature selection for wearable smartphone-based human activity recognition with able bodied, elderly, and stroke patients. *PLoS One*, 10:e0124414, 2015. https://doi.org/10.1371/journal.pone.0124414.

[48] Carlos Aviles-Cruz, Eduardo Rodriguez-Martinez, Juan Villegas-Cortez, and Andrés Ferreyra-Ramirez. Granger-causality: An efficient single user movement recognition using a smartphone accelerometer sensor. *Pattern Recognit. Lett.*, 125:576–583, 2019. https://doi.org/10.1016/j.patrec.2019.06.029.

[49] Ramona Rednic, Elena Gaura, James Brusey, and John Kemp. Wearable posture recognition systems: Factors affecting performance. In *Proc. of 2012 IEEE-EMBS Int. Conf. on Biomed. and Health Inform.*, pages 200–203, New York, NY, 2012. IEEE.

[50] Chun Zhu and Weihua Sheng. Motion- and location-based online human daily activity recognition. *Pervasive and Mob. Comput.*, 7:256–269, 2011. https://doi.org/10.1016/j.pmcj.2010.11.004.

[51] Maria Cornacchia, Koray Ozcan, Yu Zheng, and Senem Velipasalar. A survey on activity detection and classification using wearable sensors. *IEEE Sens. J.*, 17:386–403, 2016. https://doi.org/10.1109/JSEN.2016.2628346.

[52] Lu Li, Hong Zhang, Wenyan Jia, Zhi-Hong Mao, Yuhu You, and Mingui Sun. Indirect activity recognition using a target-mounted camera. In Peihua Qiu, Yong Xiang, Yongsheng Ding, Demin Li, and Lipo Wang, editors, *Proc. of the 2011 4th Int. Congr. on Image and Signal Process.*, pages 487–491, New York, NY, 2011. IEEE.

[53] Michael S. Ryoo and Larry Matthies. First-person activity recognition: What are they doing to me? In *Proc. of the 2013 IEEE Conf. on Comput. Vis. and Pattern Recognit.*, pages 2730–2737, New York, NY, 2013. IEEE.

[54] Yoshihiro Watanabe, Tetsuo Hatanaka, Takashi Komuro, and Masatoshi Ishikawa. Human gait estimation using a wearable camera. In *Proc. of the 2011 IEEE Workshop on Appl. of Comput. Vis.*, pages 276–281, New York, NY, 2011. IEEE.

[55] Kai-Tai Song and Wei-Jyun Chen. Human activity recognition using a mobile camera. In *Proc. of the 2011 8th Int. Conf. on Ubiquitous Robot. and Ambient Intell. (URAI)*, pages 3–8, New York, NY, 2011. IEEE.

[56] Ivan Laptev, Marcin Marszałek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Proc. of the 2008 IEEE Conf. on Comput. Vis. and Pattern Recognit.*, pages 1–8, New York, NY, 2008. IEEE.

[57] Yan Ke, Rahul Sukthankar, and Martial Hebert. Efficient visual event detection using volumetric features. In *Proc. of the Tenth IEEE Int. Conf. on Comput. Vis. (ICCV'05)*, volume 1, pages 166–173, New York, NY, 2005. IEEE.

[58] Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *Proc. of the 2015 IEEE Int. Conf. on Image Process. (ICIP)*, pages 168–172, New York, NY, 2015. IEEE.

[59] Jamie A. Ward, Paul Lukowicz, and Gerhard Tröster. Evaluating performance in continuous context recognition using event-driven error characterisation. In Mike Hazas, John Krumm, and Thomas Strang, editors, *Location- and Context-Awareness*, pages 239–255, Berlin, Heidelberg, 2006. Springer.

[60] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. Matrix profile IV: Using weakly labeled time series to predict outcomes. In Peter Boncz and Ken Salem, editors, *Proc. of the VLDB Endow.*, volume 10, page 1802–1812. VLDB Endowment, 2017.

[61] Reza Shakerian, Meisam Yadollahzadeh-Tabari, and Seyed Yaser Bozorgi Rad. Proposing a Fuzzy Soft-max-based classifier in a hybrid deep learning architecture for human activity recognition. *IET Biom.*, 11:171–186, 2022. https://doi.org/10.1049/bme2.12066.

[62] Manuel Gil-Martín, Rubén San-Segundo, Fernando Fernández-Martínez, and Javier Ferreiros-López. Improving physical activity recognition using a new deep learning architecture and post-processing techniques. *Eng. Appl. Artif. Intell.*, 92:103679, 2020. https://doi.org/10.1016/j.engappai.2020.103679.

[63] Wesllen Sousa Lima, Eduardo Souto, Khalil El-Khatib, Roozbeh Jalali, and João Gama. Human activity recognition using inertial sensors in a smartphone: An overview. *Sens.*, 19:3213, 2019. https://doi.org/10.3390/s19143213.

[64] Joan Serrà and Josep Lluis Arcos. An empirical evaluation of similarity measures for time series classification. *Knowl.-Based Syst.*, 67:305–314, 2014. https://doi.org/10.1016/j.knosys.2014.04.035.

[65] Jamie A. Ward, Paul Lukowicz, and Hans W. Gellersen. Performance metrics for activity recognition. *ACM Trans. on Intell. Syst. and Technol.*, 2:6, 2011. https://doi.org/10.1145/1889681.1889687.

[66] Patrick Billingsley. *Convergence of Probability Measures.* John Wiley & Sons, Inc., Hoboken, NJ, second edition, 1999.

[67] Thomas G. Dietterich. Machine learning for sequential data: A review. In Terry Caelli, Adnan Amin, Robert P. W. Duin, Dick de Ridder, and Mohamed Kamel, editors, *Struct., Syntactic, and Stat. Pattern Recognit.*, volume 2396 of *Lecture Notes in Computer Science*, pages 15–30, Berlin, Heidelberg, 2002. Springer.

[68] Igor Kononenko, Edvard Šimec, and Marko Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with RELIEFF. *Appl. Intell.*, 7:39–55, 1997. https://doi.org/10.1023/A:1008280620621.

[69] Tressy Thomas and Enayat Rajabi. A systematic review of machine learning-based missing value imputation techniques. *Data Technol. and Appl.*, 55:558–585, 2021. https://doi.org/10.1108/DTA-12-2020-0298.

[70] Craig K. Enders. *Applied missing data analysis.* Guilford Press, New York, NY, first edition, 2010.

[71] Shahidul Islam Khan and Abu Sayed Md Latiful Hoque. Sice: an improved missing data imputation technique. *J Big Data*, 7, 2020. https://doi.org/10.1186/s40537-020-00313-w.

[72] Tlamelo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. A survey on missing data in machine learning. *Journal of Big Data*, 8, 2021. https://doi.org/10.1186/s40537-021-00516-9.

[73] Roderick J. A. Little. Regression with missing x's: A review. *Amer. Stat. Assoc.*, 87:1227–1237, 1992.

[74] Per Jonsson and Claes Wohlin. An evaluation of k-nearest neighbour imputation using Likert data. In *Proc. of the 10th International Symposium on Software Metrics*, pages 108–118, Chicago, IL, 2004. IEEE.

[75] sklearn.impute.KNNImputer description. https://web.archive.org/web/20240127115457/https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html. Accessed: 2024-03-28.

[76] Jim Ramsay and Bernard Silverman. *Functional Data Analysis.* Springer, New York, NY, second edition, 2005.

[77] Michał Ciszewski, Jakob Söhl, Ton Leenen, Bart van Trigt, and Geurt Jongbloed. Testing for no effect in regression problems: a permutation approach. *to appear in Statistica Neerlandica*, 2024. https://doi.org/10.48550/arXiv.2305.02685.

[78] Phillip I. Good. Extensions of the concept of exchangeability and their applications. *J. Mod. Appl. Stat. Methods*, 1:243–247, 2002. https://doi.org/10.22237/jmasm/1036110240.

[79] Daniel Commenges. Transformations which preserve exchangeability and application to permutation tests. *J. Nonparametr. Stat.*, 15:171–185, 2003. https://doi.org/10.1080/10485250310000893l0.

[80] Yifan Huang, Haiyan Xu, Violeta Calian, and Jason C. Hsu. To permute or not to permute. *Bioinform.*, 22:2244–2248, 2006. https://doi.org/10.1093/bioinformatics/btl383.

[81] Alan D. Hutson and Gregory E. Wilding. Maintaining the exchangeability assumption for a two-group permutation test in the non-randomized setting. *J. Appl. Stat.*, 39:1593–1603, 2012. https://doi.org/10.1080/02664763.2012.661707.

[82] Rosa Arboretti Giancristofaro and Stefano Bonnini. Moment-based multivariate permutation tests for ordinal categorical data. *J. Nonparametr. Stat.*, 20:383–393, 2008.

[83] Rosa Arboretti Giancristofaro and Stefano Bonnini. Some new results on univariate and multivariate permutation tests for ordinal categorical variables under restricted alternatives. *Stat. Methods and Appl.*, 18:221–236, 2009.

[84] Wouter Duivesteijn and Arno Knobbe. Exploiting false discoveries - statistical validation of patterns and quality measures in subgroup discovery. In *Proc. 2011 IEEE 11th Int. Conf. Data Min.*, pages 151–160, Vancouver, BC, Canada, 2011. IEEE.

[85] Ronald A. Fisher. *Statistical methods for research workers.* Oliver & Boyd, Edinburgh, Scotland, first edition, 1925.

[86] Nancy S. Hall. R. A. Fisher and his advocacy of randomization. *J. Hist. Biol.*, 40:295–325, 2007. https://doi.org/10.1007/s10739-006-9119-z.

[87] Kenneth J. Berry, Paul W. Mielke Jr., and Howard W. Mielke. The Fisher-Pitman permutation test: an attractive alternative to the *F* test. *Psychol. Rep.*, 90:495–502, 2002. https://doi.org/10.2466/pr0.2002.90.2.495.

[88] Anders Oden and Hans Wedel. Arguments for Fisher's permutation test. *Ann. Statist.*, 3:518–520, 1975. https://doi.org/10.1214/aos/1176343082.

[89] Robert J. Boik. The Fisher-Pitman permutation test: A non-robust alternative to the normal theory *F* test when variances are heterogeneous. *Br. J. Math. Stat. Psychol.*, 40:26–42, 1987. https://doi.org/10.1111/j.2044-8317.1987.tb00865.x.

[90] Marti J. Anderson and John Robinson. Permutation tests for linear models. *Aust. N. Z. J. Stat.*, 43:75–88, 2001.

[91] John Ludbrook and Hugh Dudley. Why permutation tests are superior to *t* and *F* tests in biomedical research. *Am. Stat.*, 52:127–132, 1998. https://doi.org/10.2307/2685470.

[92] Joseph P. Romano. On the behavior of randomization tests without a group invariance assumption. *J. Am. Stat. Assoc.*, 85:686–692, 1990. https://doi.org/10.2307/2290003.

[93] Richard L. Schmoyer. Permutation tests for correlation in regression errors. *J. Am. Stat. Assoc.*, 89:1507–1516, 1994. https://doi.org/10.2307/2291013.

[94] Shunpu Zhang. The split sample permutation *t*-tests. *Neuroimage*, 139:3512–3524, 2009. https://doi.org/10.1016/j.jspi.2009.04.004.

[95] Hervé Cardot, Aldo Goia, and Pascal Sarda. Testing for no effect in functional linear regression models, some computational approaches. *Commun. Stat. Simul. Comput.*, 33:179–199, 2004. https://doi.org/10.1081/SAC-120028440.

[96] Rosa Arboretti Giancristofaro, Stefano Bonnini, and Fortunato Pesarin. A permutation approach for testing heterogeneity in two-sample categorical variables. *Stat. and Comput.*, 19:209–216, 2009.

[97] Oliver E. Lee and Thomas M. Braun. Permutation tests for random effects in linear mixed models. *Biometr.*, 68:486–493, 2012.

[98] Anderson M. Winkler, Gerard R. Ridgway, Matthew A. Webster, Stephen M. Smith, and Thomas E. Nichols. Permutation inference for the general linear model. *Neuroimage*, 92:381–397, 2014. https://doi.org/10.1016/j.neuroimage.2014.01.060.

[99] Anderson M. Winkler, Matthew A. Webster, Diego Vidaurre, Thomas E. Nichols, and Stephen M. Smith. Permutation inference for the general linear model. *Neuroimage*, 123:253–268, 2015.

[100] Sonja Hahn and Luigi Salmaso. A comparison of different synchronized permutation approaches to testing effects in two-level two-factor unbalanced anova designs. *Stat. Papers*, 58:123–146, 2017.

[101] Cyrus J. DiCiccio and Joseph P. Romano. Robust permutation tests for correlation and regression coefficients. *J. Am. Stat. Assoc.*, 112:1211–1220, 2017.

[102] Stefano Bonnini and Michela Borghesi. Relationship between mental health and socio-economic, demographic and environmental factors in the covid-19 lockdown period - a multivariate regression analysis. *Mathematics (MDPI)*, 10:3237, 2022.

[103] Cajo J. F. ter Braak. Predictor versus response permutation for significance testing in weighted regression and redundancy analysis. *J. Stat. Comp. Simul.*, 92:2041–2059, 2021. https://doi.org/10.1080/00949655.2021.2019256.

[104] Fortunato Pesarin and Luigi Salmaso. *Permutation tests for complex data: theory, applications and software.* John Wiley & Sons, Chichester, UK, first edition, 2010.

[105] C. B. Bell and K. A. Doksum. Distribution-free tests of independence. *Ann. Math. Statist.*, 38:429–446, 1967.

[106] Andrey N. Kolmogorov. *Foundations of the theory of probability.* Chelsea Publishing Company, New York, second edition, 1956.

[107] E.J.G. Pitman. Significance tests which may be applied to samples from any populations. *Suppl. J. Royal Stat. Soc.*, 4:119–130, 1937.

[108] E.J.G. Pitman. Significance tests which may be applied to samples from any populations. ii. the correlation coefficient test. *Suppl. J. Royal Stat. Soc.*, 4:225–232, 1937.

[109] Nikki Kolman, Barbara Huijgen, Tamara Kramer, Marije Elferink-Gemser, and Chris Visscher. The Dutch Technical-Tactical Tennis Test (D4T) for talent identification and development: psychometric characteristics. *J. Hum. Kinet.*, 30:127–138, 2017. https://doi.org/10.1515/hukin-2017-0012.

[110] David Whiteside and Machar Reid. Spatial characteristics of professional tennis serves with implications for serving aces: A machine learning approach. *J. Sports Sci.*, 35:648–654, 2017.

Curriculum Vitæ

Michał Grzegorz Ciszewski

1995/05/11	Date of birth in Zawiercie, Poland

Education

10/2019-10/2023	PhD student, Statistics group, Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands, *Applications of Statistical Theory to Sensor Data Analysis* Supervisor: Dr. Jakob Söhl Promotors: Prof. dr. ir. Geurt Jongbloed, Dr. Jakob Söhl
10/2017-07/2019	MSc Applied Mathematics, Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, Poland *Bayesian Density Estimation* Supervisor: Prof. dr hab. inż. Zbigniew Szkutnik Promotor: Prof. dr hab. inż. Bolesław Kacewicz
10/2014-07/2017	BSc Mathematics, Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, Poland *Change-point in linear regression* Supervisor: Dr Konrad Nosek

Academic Service

Co-supervisor	MSc Rafael Cuperman Coifman. *Football activity recognition: A deep learning approach to football activity recognition based on Inertial Measurement Units signals.* October 2020-June 2021, supervised jointly with prof. dr. ir. Kaspar Jansen MSc Ricardo Tebbens. *Football activity recognition: Improving and testing football activity recognition based on signal data using deep learning.* April 2022-March 2023, supervised jointly with prof. dr. ir. Kaspar Jansen
Teaching assistant	NB2171 Statistiek 2019/20 (Q3)

	TN3104WI Statistiek 2019/20, 2020/21, 2021/22, 2022/23 (Q3)
	TB131B Differentiaalvergelijkingen en Lineaire Algebra 2020/21, 2022/23 (Q1)
	EE1M21 Linear Algebra and Analysis B 2021/22 (Q2)
Lecturer	CSE1210 Probability Theory and Statistics 2021/22 (Q4)

List of Publications

1. *Rafael Cuperman, Kaspar Jansen and* ***Michał Ciszewski***: An end-to-end deep learning pipeline for football activity recognition based on wearable acceleration sensors, Sensors, 22:1347, 2022.

2. ***Michał Ciszewski****, Jakob Söhl and Geurt Jongbloed*: Improving state estimation through projection post-processing for activity recognition with application to football, Stat Methods Appl, 32:1509–1538, 2023.

3. ***Michał Ciszewski****, Jakob Söhl, Ton Leenen, Bart van Trigt and Geurt Jongbloed*: Testing for no effect in regression problems: a permutation approach, to appear in Statistica Neerlandica, 2024.

Included in this thesis.