Development and Evaluation of a Machine-Learning Based Fall Detection System for Prosthetic Knees

Heiðrún Dís Magnúsdóttir

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6236
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2024

Abstract

This thesis explores the feasibility of integrating a fall detection system into microprocessor-controlled prosthetic knees using onboard sensors, with a focus on optimizing machine learning models for real-time operational efficiency within the limited computational capacities of such devices. Initial investigations utilized the public UMAFall dataset to gain insights into fall detection methodologies and preprocessing techniques. This study also examined the potential for combining the UMAFall dataset with device-specific data to enhance model robustness and performance.

Several machine learning models, including Support Vector Machines (SVM), Logistic Regression (LR), and Random Forests (RF), were evaluated for their ability to accurately detect falls and for their suitability in terms of computational footprint when deployed in a prosthetic device environment. The models were initially trained with 42 features, which increased to 56 after incorporating pitch and roll estimations into the device-specific dataset. This study further experimented with reducing the feature set to 10 core features to examine the impact on model size and efficiency.

Results indicate that feature reduction significantly decreases model size while maintaining high accuracy, with SVM and LR models showing the most substantial reduction in size, making them ideal candidates for on-device implementation. The Random Forest model, although effective in fall detection, demonstrated a less significant reduction in size, posing challenges for its practical deployment in prosthetic knees with strict hardware limitations.

Acknowledgements

I want to thank my supervisor at Össur, Stefán Páll Sigurþórsson, who provided me with this project along with guidance and encouragement to implement my own strategies. I also want to thank Bo Bernhardsson at LTH for his supervision, feedback, and valuable insights. I am grateful for the opportunity to conduct my thesis at Össur hf, appreciating both the welcoming community I encountered and the facilities provided. Lastly, my thanks go to my family and friends for their unwavering support.

Contents

1

Introduction

1.1 Motivation

Fall detection has become a feature of many smart devices in recent years to increase the safety and well-being of individuals, with a particular emphasis on the older population, as falls can have a significant impact on their health. These smart devices, equipped with fall detection technology, have been used to provide quick assistance and response to fall incidents, thereby mitigating the severity of injuries.

Despite advancements in fall detection technology, there remains a lack of solutions tailored for amputees. This project aims to address this gap by focusing on fall detection for individuals with lower limb prosthetic devices. By integrating AI into prosthetics, the aim is to adapt fall detection to the specific needs and challenges faced by prosthetic users.

Causes for amputation include traumatic injuries, infections, diabetes, cancer, and other diseases. Referring to data from [Ziegler-Graham et al., 2008] from 2005, there are approximately 500 amputations carried out every day in the United States alone. The leading causes of lower limb amputations are vascular diseases, accounting for 54% of cases. Trauma follows closely with 45% and cancer accounts for under 2% of cases.

The consequences of an amputation are substantial. Physical challenges of lower limb amputation include mobility issues, phantom pain, soreness, muscular imbalance, and more. The psychological implications are equally significant. Amputees often find themselves grieving the lost limb, they feel anxiety, depression, and other difficult emotions following amputation.

The integration of fall detection in microprocessor-controlled prosthetics not only aims to enhance safety by reducing the risk of falls but also adds functional value. An internal trigger that stores data upon a fall event allows for a deeper analysis of factors leading to falls. Moreover, it could enable the prosthetic to actively respond

to fall incidents by triggering an alarm or alerting emergency services, providing immediate response and support to the user. The ultimate goal is to enhance the safety and well-being of amputees, improving their quality of life and their relationship with their prosthetic devices.

1.2 Related work

According to state of the art in fall detection technologies there are three categories of devices used: wearable devices, camera-based devices, and ambiance devices [Tanwar et al., 2022]. Wearable devices refer to sensors that individuals carry which can detect abrupt changes in motion or orientation indicating a fall. Camera-based devices utilize video surveillance or motion-sensing cameras to observe and analyze movements and determine if a fall has occurred. Ambiance devices equip environmental sensors such as infra-red sensing, vibration sensors, or lasers to monitor changes that may signify a fall.

Each detection approach has its pros and cons and there are a lot of options for fall detection but it depends on the requirements of each specific use case which device type should be used. For this research, the state-of-the-art sensor device approach was explored since the application will be on microprocessor-controlled prosthetics which include sensors such as IMUs, load cells, and angle sensors.

To name a few studies done in the past, Shawen et al. [Shawen et al., 2017] aimed to use data from mobile phones to detect falls. They developed a classifier using data from both able-bodied individuals and amputees and found that it could effectively distinguish between falls and daily activities in individuals with transfemoral amputations (TFA). This research is significant as it suggests that data from non-amputees can be used to detect falls in amputees. While this study utilized mobile phone data for fall detection, this research focuses on using onboard sensor data from prosthetic devices. This approach takes advantage of the unique positioning and capabilities of these sensors, potentially providing a more accurate and personalized fall detection system.

In a 2021 review paper by Usmani, Saboor, Haris, Khan, and Park [Usmani et al., 2021], the authors explored advancements in both fall detection and fall prevention systems, particularly for the older population, leveraging Machine Learning (ML) algorithms. They noted that while a majority of studies focus on fall detection, a smaller portion addresses fall prevention. The researchers highlighted the frequent use of Support Vector Machines (SVMs) due to their ability to effectively manage high-dimensional data and their memory efficiency. However, they also pointed out limitations, such as the real-life applicability of these systems and the tendency to collect data from younger age groups, despite the primary goal being the detection

and prevention of falls in the older population.

Different methods for fall detection include combining different measurement methods such as mobile phone + IMU sensors [Casilari and A.Santoyo-Ramón, 2018], heart rate + IMU sensor [Nho et al., 2020], eyewear that contains IMU sensors [Lin et al., 2020], and optimal sensor locations have been researched by [Tang et al., n.d.]

1.3 Research goals/questions

This study was guided by several key questions, which aimed to explore the feasibility of using onboard sensory systems in prosthetic knees for fall detection.

- Is it feasible to utilize the onboard sensory system of prosthetic knees for fall detection based on publicly available data?
- Would relying solely on lower limb sensors negatively impact detection results compared to using full body sensor data?
- Could the integration of additional sensory signals from onboard sensors in prosthetic devices potentially enhance fall detection algorithms?
- How feasible is it to deploy optimized machine learning models directly onto the limited computational environments of prosthetic devices?

1.4 Ethical Considerations

In developing machine learning models for fall detection within prosthetic devices, several ethical considerations must be addressed to ensure the responsible use of technology. Ensuring the privacy of user data should always be a top priority. It is important to clearly communicate the purpose of data gathering, how the data will be used, and the participants' rights regarding data privacy.

Moreover, it is important to be clear about how the fall detection system works and how data is used. This helps ensure that users understand what the technology does and how their information is handled.

1.5 Report structure

Chapter 2 explains methods and processes used in the thesis. Chapter 3 explains how data was collected and processed to create machine learning models equipped for fall detection. Results of the machine learning models trained are introduced in Chapter 4 and a discussion on the results is in Chapter 5.

2

Background

This chapter provides context and foundation knowledge for the topics covered in this study. The chapter begins with an explanation of microprocessor-controlled knees and then introduces the inertial measurement unit. The UMAFall dataset is then introduced, leading to a discussion on feature extraction and selection. The Kalman filter is addressed next, followed by an overview of the Machine Learning methods used, specifically Support Vector Machines, Logistic Regression, and Random Forests. The chapter concludes by introducing performance evaluation metrics including Cross-validation, Precision, Recall, and F1-score.

2.1 Microprocessor-controlled knees

Microprocessor-controlled knees (MPKs) are what the name suggests, prosthetic knees controlled with a microprocessor. These cutting-edge devices offer improved control through embedded electronic systems and allow for improved stability and more natural movement for amputees. The microprocessor receives signals from sensors located throughout the prosthetic, and this can allow for real-time event detection and adaptive control.

Microprocessor knees (MPK's) can have either passive or active actuation. These two types of MPKs differ in how they control the movement of the artificial knee joint.

Passive MPKs, such as Össur's Rheo Knee® [*Össur: Rheo Knee*®], primarily provide resistance to movement. They restore eccentric muscle activity through joint resistance, providing support during stance and breaking during the swing phase of the gait cycle.

Active MPKs, like Össur's Power Knee™ [*Össur: Power Knee*™], provide active torque as well as resistance. This way they restore both eccentric and concentric activity around the knee [Creylman et al., 2016].

2.2 Inertial measurement unit

An inertial measurement unit (IMU) is an electronic device that can measure movement and orientation properties, which have proven efficient in classifying different human activity patterns [Jantawong et al., 2021]. An IMU typically contains a gyroscope, accelerometer, and sometimes magnetometer. The gyroscope reports angular rates, the accelerometer measures acceleration, and the magnetometer measures changes in the magnetic field.

The accelerometer within an IMU does not measure acceleration directly; instead, it measures "specific force" which is the apparent force that an observer would measure in a non-inertial (accelerating) frame of reference. This force can be described as $\bar{f} = \bar{a} - \bar{g}$, where $\bar{a}$ is the measured acceleration and $\bar{g}$ represents the acceleration due to gravity. Here, $\bar{f}$ represents a vector directed upwards with a magnitude of 1g when the accelerometer is at rest in the Earth's gravitational field.

While IMUs often include a magnetometer to measure changes in the magnetic field, these measurements can be heavily influenced by a range of sources such as soft or hard iron distortions, sensor nonorthogonality, and bias. These disturbances can cause the magnetometer readings to vary significantly across different environments, necessitating frequent recalibration [Tahir et al., 2019]. Due to these potential disturbances and the complexity of ensuring accurate magnetometer readings, they are often excluded from IMU-based studies, including this one.

2.3 UMAFall dataset

The dataset utilized in the initial stages of this research was derived from an experimental testbed initially developed by Jose Antonio Santoyo-Román for his MSc thesis at the University of Malaga [Casilari and A.Santoyo-Ramón, 2018]. The testbed comprised an Android smartphone and four wireless nodes or SensorTags worn by the subjects. The smartphone was kept in a trouser pocket, while the SensorTags were attached to the ankle, wrist, chest, and waist. During each experiment, the subject performed various movements and activities, with the five sensors capturing data such as accelerometer, gyroscope, and magnetometer readings.

Each recorded movement was stored in a CSV file, with the file name indicating the ID of the subject, type of movement, subtype of movement, trial number, and the date and time of the experiment. The dataset includes 746 files, capturing 12 different Activities of Daily Living (ADLs) and 3 different types of falls, all carried out in a domestic environment. The ADLs range from regular activities like walking and jogging to specific hand activities like clapping, raising hands, making a phone call, and opening a door. The falls were simulated on a mattress and categorized

into lateral, frontal, and backward falls.

2.4 Feature Extraction and Selection

Feature extraction is a way to preserve characteristics while reducing the dimensionality of the data. This is a fundamental process in supervised machine learning which not only simplifies the data collection process by identifying and excluding irrelevant variables but also acts as a form of regularization, potentially mitigating overfitting and boosting model performance.

Time series data from sensors presents unique challenges due to its temporal dependencies and potential noise. This makes feature extraction crucial for isolating meaningful patterns from raw sensor data [Agrawal and Sharma, 2022].

Feature selection is essential in supervised machine learning because not all input variables or features may contribute positively to performance. This process can act as a form of regularisation, potentially reducing overfitting and enhancing the model's performance. Additionally, if certain variables are found to be irrelevant, the data collection process can be simplified as these variables won't need to be collected [Lindholm et al., 2022].

2.5 Kalman Filter

Rudolf E. Kálmán introduced a new approach to linear filtering and prediction problems in 1960 [Kalman, 1960], which has since been widely used in applications requiring the estimation of unknown variables. This approach is now known as the Kalman filter. The Kalman filter is a recursive mathematical technique to estimate the state of a dynamic system from noisy measurements. The filter leverages information about a system's dynamics and the associated measurements, making it particularly useful in scenarios where data is subject to noise, which is common with sensor data. The filter operates by continuously predicting the system's current state and updating the prediction as new measurements are collected, thereby enhancing the accuracy of state estimates [Li et al., 2015].

The Kalman Filter operates through a prediction and an update state. Initially, the filter predicts the next state of the system and its uncertainty [Geng, 2023]:

$$\mathbf{x}_t = \mathbf{A}_{t-1}\mathbf{x}_{t-1} + \mathbf{q}_{t-1} \tag{2.1}$$

$$\mathbf{P}_t^- = \mathbf{A}_{t-1}\mathbf{P}_{t-1}\mathbf{A}_{t-1}^T + \mathbf{Q}_{t-1} \tag{2.2}$$

where $\mathbf{x}_t$ is the predicted state vector, $\mathbf{P}_t^-$ is the predicted state error covariance, $\mathbf{A}$ is the state transition matrix, $\mathbf{q}$ is the process noise (assumed zero-mean white Gaussian) and $\mathbf{Q}_{t-1}$ is the process noise covariance.

During the update phase, the filter adjusts its estimates based on new measurements:

$$\mathbf{S}_t = \mathbf{H}\mathbf{P}_t^- \mathbf{H}^T + \hat{\Sigma}_{ea} + \Sigma_{na} \tag{2.3}$$

$$\mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}^T \mathbf{S}_t^{-1} \tag{2.4}$$

$$\mathbf{x}_t = \mathbf{x}_t^- + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}\mathbf{x}_t^-) \tag{2.5}$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_t^- \tag{2.6}$$

Here, $\mathbf{S}_t$ is the innovation covariance, incorporating the predicted error covariance ($\mathbf{P}_t^-$), the measurement matrix ($\mathbf{H}$), and the covariances of the process ($\hat{\Sigma}_{ea}$) and measurement noise (Σ_{na}). These covariances help quantify the expected errors in the process and measurements, ensuring that the filter's updates are appropriately scaled to the level of uncertainty. $\mathbf{z}_t$ is the measurement vector, $\mathbf{K}_t$ is the optimal Kalman gain, and then the state estimate and the state estimate covariance are updated in equations (2.5) and (2.6).

After discussing the general mechanics of the Kalman Filter, it is important to specify its application in the context of this thesis. The Kalman filter used in this project was specifically designed by another researcher at Össur hf as part of a previous master's thesis [Geng, 2023]. This filter has been adapted to work with the prosthetic devices developed by Össur, utilizing sensor data from these devices to estimate the pitch and roll movements of the lower limb prosthetics. The adaptation involved adjusting the filter to take dataframes directly from these devices as input and returning real-time estimations of pitch and roll. This implementation is crucial as it provides the machine-learning models with dynamic features that are predictive of fall events, enhancing the models' accuracy and reliability in detecting falls.

2.6 Machine Learning methods

Machine Learning is at the heart of many modern technologies and could be described as the latest attempt to distill human knowledge and reasoning into a form suitable for constructing machines and engineering automated systems. Machine learning is focused on designing algorithms that automatically extract valuable information from data. The core concepts are data, a model, and learning [Peter et al., 2020].

This section will explain the different machine learning methods that were tried for detecting falls from labeled sensor data.

Support Vector Machines

The support vector machine is a supervised learning method introduced by [Cortes et al., 1995] in 1995. SVMs implement the following idea: The input is mapped to an n-dimensional space Z (n is the number of features we have and each feature is the value of a particular coordinate). A SVM then finds a linear decision surface that separates the coordinate points into different classes.

The support vectors are the points nearest to the optimal hyperplane. The SVM strives for a good separation between classes and a good separation is achieved when the hyperplane has the greatest distance from the closest training data points across all classes. Generally, a larger margin correlates with a lower generalization error, hence improving the classifier's performance.

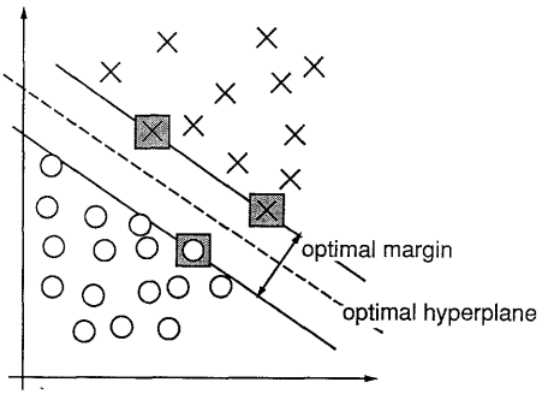


Figure 2.1 An example of a 2-dimensional space. The support vectors are marked with grey squares [Cortes et al., 1995]

Logistic regression

Logistic regression was initially utilized in biomedical studies, but its use has expanded over the past 20 years to social science research, marketing, business applications, and even genetics [Agresti, 2002].

Logistic regression uses the sigmoid function to estimate probabilities that an instance belongs to a class. The output from a logistic regression model is the logistic of a weighted sum of the input features (plus a bias term) [Geron, 2019].

The logistic is a sigmoid function that converts real-valued numbers to a value be-

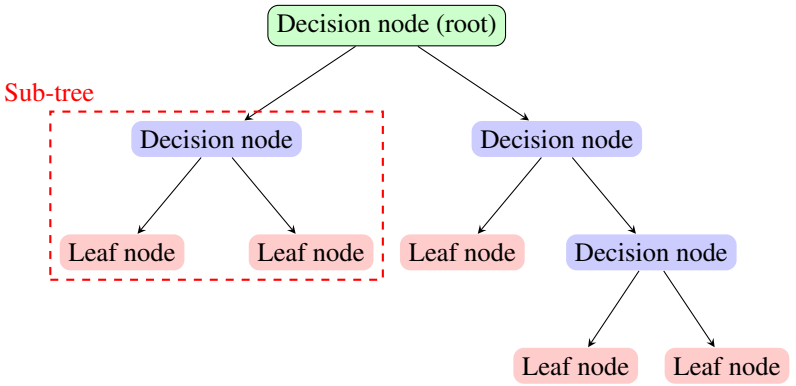


Figure 2.2 An example of a decision tree, showing the root decision node, intermediate decision nodes, and leaf nodes.

tween 0 and 1:

$$\sigma(t) = \frac{1}{1+\exp(-t)} \tag{2.7}$$

The logistic regression model outputs a function which represents the probability of a particular event occurring

$$\hat{p} = \sigma(\mathbf{x}^T \theta) \tag{2.8}$$

where $\mathbf{x}$ is the feature vector and θ is the model parameter vector.

Random Forests

Before introducing Random Forests, it's important to understand tree-structured classifiers, or more precisely, binary tree-structured classifiers which are often called decision trees. These classifiers are constructed by repeatedly splitting subsets of the feature space into two descendant subsets, starting with the entire feature space itself. Figure 2.2 provides a visual representation of a decision tree structure.

The process of predicting a class for a measurement vector in a tree-structured classifier involves traversing the tree from the root node (the entire feature space) to a terminal node (a subset of the feature space). The predicted class is given by the class label attached to that terminal node [Breiman et al., n.d.]

A Random Forest is an ensemble of Decision Trees that introduces increased randomness into the model. The classifier tries to find the best feature among a random subset of features when splitting a node in a tree, instead of searching for the best feature like decision trees do [Geron, 2019].

Since each decision tree is built from a random subset of the training dataset, some observations may be included multiple times in the sample, while others may not appear at all. But as every tree in the forest contributes to the final ensemble model, Random Forest models are known to result in a less biased model [Williams, 2011].

A drawback of random forests is that they can be computationally expensive, particularly when dealing with large datasets or a large number of trees.

2.7 Performance evaluation

Cross-validation

Cross-validation is a widely used method to assess the predictive performance of models and prevent overfitting. The most common type of cross-validation is k-fold cross-validation. The k-fold cross-validation process involves splitting the available data into k groups or "folds". The process then involves using k-1 of these groups to train the model. The model is then tested and evaluated on the remaining group. This procedure is repeated for all k possible choices for the evaluation group, each time holding out a different subset of the data [Bishop, 2006].

In situations where the data is imbalanced, meaning there is an imbalance of instances between the classes, standard k-fold cross-validation can provide misleading results because the test data may not reflect the distribution of the training data. To address this issue, a variant of k-fold cross-validation known as Stratified Cross-Validation (SCV) can be employed. In SCV, the data is not split randomly but instead, each fold is made by preserving the percentage of samples for each class. This ensures that each fold is a good representative of the overall sample distribution [Szeghalmy and Fazekas, 2023].

k-fold cross-validation results in k different performance scores, one for each iteration of training and testing. The final score is derived by averaging the k scores, providing a robust measure of model performance.

Precision

Precision, also known as Confidence in the field of data mining, evaluates the performance of predictive models by focusing on the positive predictions made by a model. Mathematically, precision is the ratio of true positive (TP) predictions to the total number of predicted positives (PP) [Powers and Ailab, n.d.] It can be presented as follows:

$$\text{Precision (confidence)} = \frac{\text{True Positives}}{\text{True positives} + \text{False positives}} \tag{2.9}$$

Recall

Another vital evaluation measure is Recall, also known as Sensitivity. Mathematically, Recall is the ratio of true positives (TP) to the total number of actual positives (RP), which includes both true positives and false negatives [Powers and Ailab, n.d.] It can be presented as follows:

$$\text{Recall (sensitivity)} = \frac{\text{True Positives}}{\text{True positives} + \text{False negatives}} \tag{2.10}$$

F1-score

The F1-score is an important evaluation metric that combines both Precision and Recall to provide a more holistic view of the model's performance. Mathematically, the F1-score is defined as the harmonic mean of Precision and Recall. It can be formulated as follows:

$$\text{F}_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.11}$$

F1-score is particularly useful for handling imbalanced datasets, which are prevalent in many real-world classification problems. By taking both Precision and Recall into account, the F1-score balances the trade-off between making correct positive predictions (Precision) and correctly identifying actual positive instances (Recall). As a result, a higher F1-score indicates not only that the positive predictions are reliable (high precision), but also that a high proportion of actual positive instances have been correctly identified (high recall) [Lindholm et al., 2022].

In the context of this project, the "positive" case can be defined in different ways depending on the focus of the analysis. For instance, in a fall detection system, one might consider a "fall" as the positive case in one scenario and a "non-fall" as the positive in another. This flexibility allows us to tailor the evaluation to reflect different priorities, such as minimizing false negatives in fall detection to ensure safety. Accordingly, F1-scores will be computed for both "fall" and "non-fall" as separate positive cases to provide a comprehensive evaluation of the model's performance across different types of classification outcomes.

3

Methodology

The purpose of the research is to determine the potential of machine learning models in accurately detecting falls based on data derived from lower leg sensors. This chapter outlines the systematic approach and steps implemented to achieve these results. Firstly, the data collection from the device is introduced. Secondly, the preprocessing steps are outlined for both UMAFall data and device data, this includes several steps such as manual labeling, adjustments to datasets, estimating pitch & roll angles, and the application of a sliding window approach.

3.1 Data collection

All simulated sensor data for indoor falls was collected inside Össur's MotionLab and the participant of the data collection process is a transfermoral (above the knee) amputee using a passive microprocessor-controlled Knee, working in Össur as a product tester.

During the data collection process, a range of movement sequences was performed, incorporating everyday activities and a couple of different falling scenarios. Everyday activities included level ground walking, uphill and downhill incline walking, ascending and descending stairs, as well as the process of sitting down on a chair and standing back up. The falling sequences also varied, with some involving the user taking a few steps forward before kneeling, others where the user turned before kneeling, and some instances where the user fell without taking any preceding steps. While the falling sequences did vary, all falls executed were forward falls. Sideways falling or backwards falling were not incorporated in these sequences. Sideways and backwards falling is deemed not to be caused by the prosthetic device, nor can the device do anything to prevent such a fall. Figure 3.1 displays a typical 'falling' sequence where the user takes a few steps before kneeling on the ground.

Throughout the test, data is collected with Össur's toolbox that connects via Bluetooth directly to the prosthetic's sensors and allows for real-time monitoring and

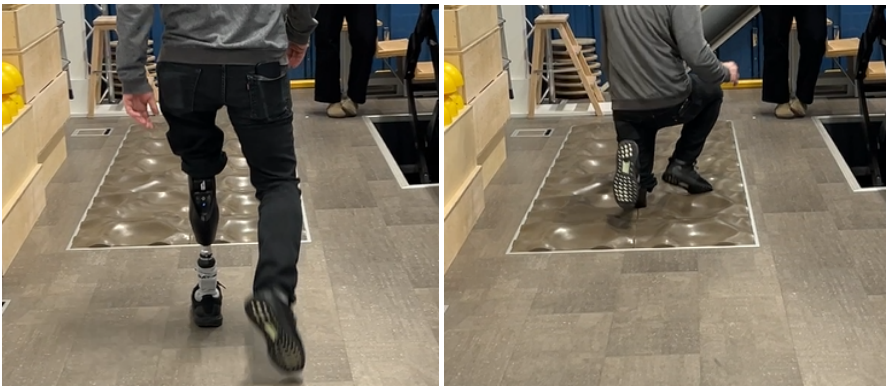


Figure 3.1 Data collection setup

data collection. The data consists of output signals from the IMU and force sensor, as well as gait phase information. The sampling rate of all measurements was 1000 Hz, corresponding to a sampling interval of 1 ms.

Össur's microprocessor knees contain IMUs as well as other sensors such as angle sensors and load cells. The IMU data as well as signals from the additional sensors hold the potential for effective fall detection. The IMU in the device used in this study is the Bosch BNO055 model [BOSCH, 2023]. This sophisticated unit combines a 3-axis accelerometer, gyroscope, and magnetometer along with orientation software and sensor fusion. It provides high-precision data in real time. The BNO055 is commonly recommended for personal health and fitness applications, indoor navigation, and other contexts requiring precise motion tracking and context awareness. Figure 3.2 illustrates the orientation of the three-dimensional axes with respect to the knee, providing a visual reference for how the IMU is positioned within the prosthetic device.

Figure 3.3 displays the raw signals collected from the IMU during level ground walking. The signals during walking exhibit consistent oscillations as normal gait is rhythmic and repetitive. After around 6 seconds in Figure 3.3, the user made a U-turn and continued walking. A slight change in the signal pattern can be observed during the U-turn, followed by the return of the consistent oscillations of normal walking. This highlights the predictability and uniformity of IMU signals during everyday movements such as walking.

Figure 3.4 then displays the raw signals collected for a falling event followed by recovery. In this sequence, the user takes two steps before kneeling forward onto his knee. The fall happens after about three seconds and is followed by a recovering

Figure 3.2 Rheo Knee 3 - The figure showcases the orientation of the three-dimensional axis with respect to the knee.

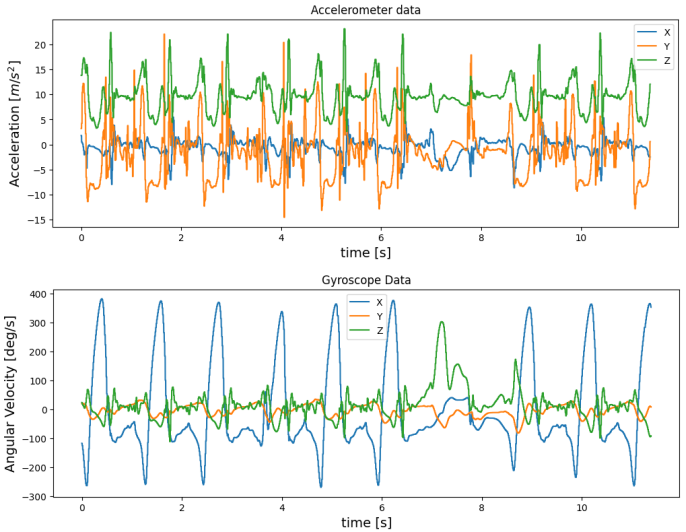


Figure 3.3 IMU signals during walking. 3-axis accelerometer (top) and 3-axis gyroscope (bottom).

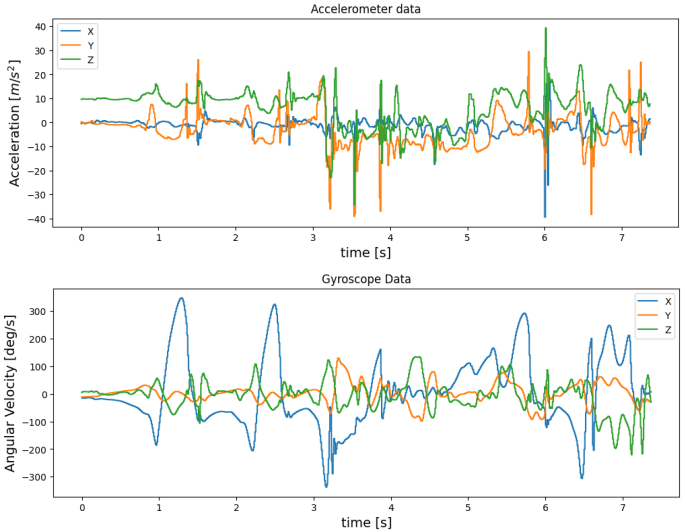


Figure 3.4 IMU signals while the user takes two steps, kneels, and stands up again. The kneeling happens after roughly 3 seconds.

phase while the user stands up again. The falling and recovery phase is characterized by abrupt changes in signal intensity and the sequence underscores the complexity and unpredictability of IMU signals during abnormal movements.

3.2 Preprocessing

Manual Labeling

The process of manual labeling was undertaken to precisely identify and categorize the fall and non-fall events in the datasets. For the UMAFall dataset, the indication of whether a file included a fall event was embedded within the filename itself. However, to ensure accuracy and to facilitate the windowing technique deployed in this research (discussed in detail later in this chapter), each file categorized as a 'fall' was visually inspected. This allowed for the identification of the specific sample number associated with the fall event.

In contrast, the device dataset required a different approach for manual labeling. Documentation was maintained during the data collection, noting the specific movements encapsulated in each data sequence. This, combined with video recordings taken during the tests, provided a comprehensive reference for accurately labeling the data. By carefully tracking each movement in the sequences, it was possible to label the data accurately, which is crucial for the reliability of analysis later on.

UMAFall data

From the UMAFall dataset, only ankle, wrist, and chest data was used. This means excluding waist data which was collected on mobile phones. The dataset was sampled at a rate of 20 Hz, and movements were monitored for 15 seconds. The orientation of the axes from the UMAFall sensors is different from the device sensors. Upon inspection of the UMAFall dataset, it was found that data from certain subjects was unreliable and showed more outliers in the data than in other parts of the data. This could be due to factors such as sensor errors or unusual movements. After eliminating sequences showing substantial faults, what is left is a dataset that includes data from sensors located on the ankle, wrist, and chest. The UMAFall set contains 465 sequences of Activities of Daily Living (ADL), and 157 fall-event sequences.

Null and NaN values were encountered in the dataset. To deal with these, they were replaced with the mean value of the respective column. Given that these values constituted approximately 1% of the data, their replacement would not significantly impact the results and this approach allowed maintaining the size of the dataset.

Combining UMAFall & Device data

Obtaining consistency and uniformity in the data when combining data sources is crucial to simplify the training process, making it more efficient and easier to manage. To establish this consistency in the data before the training process, several steps were taken.

Axis aligmnent. Firstly, the Y- and Z-axis of the UMAFall data were switched to match the orientation of the device data. This ensures that the measurements from different sources are directly comparable.

Units verification. Secondly, the units of measurement were verified to match across the data sources. Specifically, it was ensured that acceleration measurements from both sources were in meters/s^2 and that angular velocity measurements from gyroscopes were in degrees per second (deg/s) rather than radians per second (rad/s).

Windowing. Following these preprocessing steps, each time series data frame is trimmed to a 5-second window. For sequences containing a fall, the window is centered around the fall, and for non-fall sequences, the window is centered around the absolute peak acceleration point. Centering around the peak acceleration attempts to capture the most impactful segment of the data clip, which is anticipated to contain key features essential for the model's performance. The selection of a precise 5-second window is strategic in capturing data associated with a fall, without incorporating excessive, potentially irrelevant information. This is because a fall is typically estimated to occur within 1-4 seconds.

Data Downsampling. When combined with the UMAFall data for training, the device data, originally sampled at a rate of 1000Hz, was downsampled to 20Hz to match the UMAFall data. This downsampling was achieved by selecting every 50th sample from the original data. The downsampling process was applied after extracting the 5-second windows centered around the fall events or peak accelerations. This sequencing ensures that key moments, particularly those containing fall events, are preserved even though some data granularity is lost. This preprocessing step was crucial to ensure compatibility between the different data sources.

Windowing for Device Data. Similar to the procedure used with the UMAFall data, the device data was also divided into 5-second windows. If a window contained a fall event, it was arranged to center around the fall. For non-fall windows, the focus was the point of peak acceleration. After splitting the device data into 5-second windows, the dataset comprised 20 windows. Of these, 9 contained a fall event and 11 featured non-fall activities. To assess the feasibility of integrating data from multiple sources, a subset of the device data was set aside for testing, with the remainder merged with the UMAFall data to form the training set.

Sliding windows

The ultimate goal in this part of the project is to process the data in a way that allows for effectively capturing the fall event. Given that a fall typically lasts between 1 to 4 seconds, the primary objective is to accurately record not only the fall itself but also the preceding moments. This information could enable the development of a trigger mechanism that can allow for the storage of critical data from the moments before a fall.

After the manual labeling process, each sequence that contained a fall had one sample identified as a fall event. Given the sampling rate of the device data, this would suggest that the fall event happens within a millisecond, which is not accurate. The goal is to pinpoint where the fall initially happens and to provide a broader context for each fall event, which is why the sliding window approach was implemented.

The use of overlapping sliding time windows was key in increasing the number of fall events in the dataset collected from the device. As the window moves across the sequence, overlapping segments are created, each containing a portion of the fall event. This method effectively augments data, resulting in more instances of fall events for the model to learn from.

In this setup, a fall event label is applied to a window, not just a single sample in a sequence. However, a window is only labeled as a fall if the fall sample is within a specific range from the window's border.

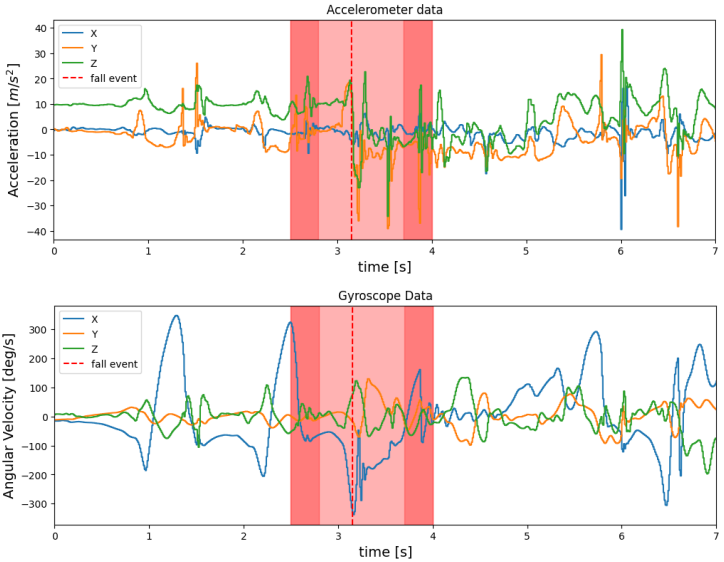


Figure 3.5 Visualization of a time window on accelerometer and gyroscope data. The window spans 1.5 seconds which is 1500 data points, with borders highlighted at 300 data points (0.3 seconds) on either side. This window is labeled as a fall window as it captures the fall event within its borders.

The sliding window technique comes with adjustable parameters, such as the window length, the step size of the window, and the required distance of a fall sample from the border for the window to be labeled as a fall event. In this step these parameters were tuned to optimize the accuracy of the fall detection algorithm.

An example of how these sliding time windows are applied can be seen in Figure 3.5, where both accelerometer and gyroscope data are displayed. The window spans 1.5 seconds or 1500 data points with a margin of 300 data points (0.3 seconds) on each side.

3.3 Feature engineering

In the process of preparing the data for the ML models, feature engineering played a critical role. Seven features on each axis (X, Y, Z) were computed for both accelerometer and gyroscope signals. These features were selected based on their potential to provide meaningful insight into the characteristics of fall events:

- Maximum

- Minimum
- Median
- Variance
- Absolute energy
- Absolute maximum
- Root mean square

Pitch & roll estimations as features

When working with device data, additional features were added to the machine-learning models in the form of pitch and roll estimations. These estimations were obtained using a part of a Kalman filter algorithm, which was previously developed by another student at Össur hf as part of his master's thesis project [Geng, 2023]. This sophisticated algorithm uses sensor fusion to estimate the attitude, velocity, and position of the lower limb prosthesis based on the data from the embedded IMU.

For this project, the Kalman filter was adapted to take a dataframe as input and return the pitch and roll estimations for a measurement sequence. This adaption made it possible to extract the pitch and roll estimations directly from the sensor data, providing valuable additional features for the machine-learning models.

The Kalman Filter algorithm and its parameters, adapted from our proprietary configurations, are crucial for accurate attitude estimation in gait analysis. Algorithm 1 details the step-by-step estimation process, utilizing the constants specified in Table 3.1. This setup ensures precise computation of pitch and roll necessary for assessing prosthetic gait dynamics.

Table 3.1 Constants and Parameters Used in the Kalman Filter Algorithm

Symbol	Description	Value
Δt	Sampling interval (s)	0.001
$\mathbf{X}_0$	Initial state vector	$[0,0,1]^T$
$\mathbf{P}_0$	Initial covariance matrix	$2.0 \times 10^{-8}\mathbf{I}_3$
$\mathbf{Q}$	Process noise covariance matrix	$2.0 \times 10^{-6}\mathbf{I}_3$
$\mathbf{R}$	Measurement noise covariance matrix	$2.0 \times 10^{-1}\mathbf{I}_3$
L	Leg length (m)	0.4
g	Gravitational acceleration (m/s^2)	9.8

Algorithm 1 Kalman Filter for Gait-based Attitude Estimation

Input: DataFrame with IMU data $\{y_{\text{gyro}}, y_{\text{acc}}\}$
Output: Arrays of pitch (β) and roll (γ) estimations
Initialize constants: Δt, buffer size, L, g
Initialize state and covariance: $\mathbf{x}_0$, $\mathbf{P}_0$
Define noise covariances: $\mathbf{Q}$, $\mathbf{R}$
for each measurement in DataFrame **do**
    if stationary condition **then**
        Estimate state with no external acceleration
        Update step using $\mathbf{H}$ and stationary assumptions
    else if stance phase **then**
        Calculate external accelerations based on leg dynamics
        Update step considering stance phase dynamics
    else
        General update step during swing phase
    end if
    Normalize $\mathbf{x}$ to prevent drift
    Calculate β and γ using $\mathbf{x}$
end for
return Arrays of β and γ

3.4 Model training and evaluation

Data Splitting

The initial phase of model training involves splitting the datasets into training and testing sets. This step is crucial to ensure that the models are trained on one subset of the data and validated on an independent subset, which helps in evaluating their generalization capabilities.

To address the issue of class imbalance, which is particularly significant in fall detection due to the rarity of fall events compared to non-fall events, stratified sampling was employed. Stratified sampling ensures that both the training and testing sets have a proportionate representation of each class.

The data splitting was implemented using the `train_test_split` function from Scikit-Learn [*scikit-learn: Machine Learning in Python*], configured as follows:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y)
```

The parameters used in the `train_test_split` function are:

- **test_size=0.3**: This parameter specifies that 30% of the data is reserved for the test set, ensuring that the majority of the data is available for training the models.
- **random_state=42**: This parameter ensures reproducibility in the results. By fixing the random state, we ensure that the same data split is achieved every time the code is run, which is important for reproducibility of the model performance metrics.
- **stratify=y**: This ensures that the train and test sets have approximately the same percentage of samples of each class. This is crucial in a scenario like fall detection where the class distribution is imbalanced, as it ensures that both sets are representative of the overall dataset.

Model Selection

The choice of ML models is a crucial step in achieving accurate and efficient fall detection. The models selected for this study were Support Vector Machines (SVM), Logistic Regression, and Random Forest, all of which were previously discussed in detail in Chapter 2.

By training and evaluating three different models, it was feasible to compare their performances and select the best model.

Hyperparameter Selection

In this study, the models were utilized with their default hyperparameters as provided by the Scikit-Learn library [*scikit-learn: Machine Learning in Python*]. The decision to use default settings was driven by the following considerations:

1. **Baseline Performance Assessment:** Using default parameters allowed for an initial assessment of each model's performance without the complexity of tuning.
2. **Project Scope and Constraints:** Given the projects focus on data collection and analysis rather than optimizing model performance, extensive hyperparameter tuning was not deemed a critical component. The primary objective was to evaluate the feasibility of different models for fall detection using basic configurations.

Support Vector Machines

The first model explored was the Support Vector Machine (SVM), which is known for its effectiveness in high-dimensional spaces and its flexibility in modeling different forms of data [Geron, 2019].

The SVM model was created using a linear kernel. Before model training, the input data was standardized using the StandardScaler function from the preprocessing module in Sklearn. The model was trained on the training data and evaluated using 5-fold cross-validation.

The trained SVM model was then used to predict the labels for the test data, which provided a final evaluation of the model's performance.

Logistic Regression

The second model utilized was Logistic Regression which is widely used for binary classification problems.

Similar to the SVM model, the input data for the Logistic Regression model was standardized before training, and 5-fold cross-validation was utilized for training, and the trained model was then used to predict labels for the test data.

Random Forest

The third model employed was the Random Forest, an ensemble learning method that operates by constructing multiple decision trees.

The Random Forest model was trained in a similar way to the previous models. The input data was standardized, and the model was trained with 5-fold cross-validation on the training dataset. Again, the trained model was then used to predict the labels for the test data.

Feature Importance

In the initial stages of model training, feature filtering was carried out using the tsfresh library [Christ et al., 2018]. This helped in excluding features that did not contribute significantly to the model's detection abilities.

However, when working with device data alone, a more focused approach was taken in extracting only a small set of features that contribute significantly to the performance of the models. After the initial training of the models, the 10 most contributing features were identified by extracting the feature importance from the models. Once the most important features were identified, new models were trained and evaluated using this refined feature set.

Pitch and Roll Estimation

In the final stages of model training and evaluation for the device data, pitch and roll angles were estimated using a Kalman filter, chosen for its efficacy in handling noisy sensor data and providing accurate real-time estimations. This approach

allowed for the dynamic adjustment of pitch and roll estimations based on the evolving sensor readings, thus reflecting more realistic movement patterns of the device users.

The pitch and roll estimations were obtained in the form of time-series vectors, aligning with the existing sensor data collected from the device. This way, the pitch and roll estimations could be seamlessly appended to the dataframe, enhancing the dataset with additional features that improve the model's context awareness and predictive capabilities.

The integration process involved adding these estimations before the feature extraction phase, ensuring that all subsequent analyses could use the enriched data effectively. The increase in dataset features from 42 to 56 allowed for a more detailed analysis and improved the depth of the model's input, helping in a more detailed understanding and detection of fall instances.

Performance Evaluation

To evaluate the performance of all three models, the cross-validation scores as well as the classification reports and confusion matrices were observed. Cross-validation was crucial as it helped to determine whether the models were at risk of overfitting, providing a robust measure of model reliability across different subsets of the dataset.

The classification report provides important metrics such as precision, recall, and F1-score for each class. Evaluating both F1-scores with 'Fall' and 'Not Fall' as positive outcomes was effective in revealing how the models performed accross the different classes. Given the known imbalance in the dataset, this was an invaluable performance metric. Comparing the two scores provided deeper insight into the model's effectiveness in accurately predicting both fall and non-fall events.

Confusion matrices further supplemented evaluation by visualizing the model's performance on the test data. These matrices detailed the number of true positives, true negatives, false positives, and false negatives. This visualization helped to pinpoint the models' strengths and weaknesses in classifying each type of event.

While striving for high accuracy is typical in model development, for the purposes of this project, achieving 100% accuracy is not critical. The primary goal here is to facilitate data collection to enhance prosthetic functionality, rather than immediate response to fall events. This perspective aligns with the project's focus on using the model primarily for gathering insights to improve device performance over time, rather than for triggering real-time alarms.

4

Results

In this chapter, the results obtained from various model evaluations are presented. Cross-validation (CV) scores are used to estimate the models' performance and robustness during the training phase, utilizing the training dataset. These scores help to determine the generalizability of the models under different data conditions. In contrast, F1-scores and confusion matrices, derived from the testing dataset, show the actual performance of the models on unseen data.

4.1 UMAFall dataset

Table 4.1 shows cross-validation scores comparing the performance of models trained and tested with full-body sensor sets versus ankle-only sensor data from the UMAFall dataset. Note: CV scores are derived during the training phase.

Table 4.1 5-fold cross-validation scores for different classification models using measurements from UMAFall's full body sensor sets compared to their ankle sensor only.

	SVM	Logistic Regression	Random Forest
Full-body dataset	100%	100%	99%
Ankle only dataset	97%	97%	96%

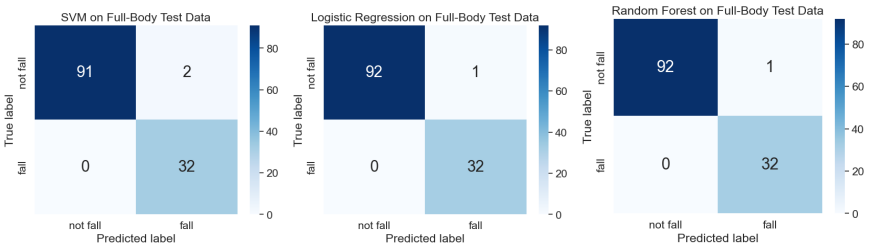


Figure 4.1 Confusion matrices for models trained on full-body data from the UMAFall dataset when tested on a separate test set from the same dataset. From left to right: Support Vector Machine (SVM), Logistic Regression, and Random Forest. Note: Reflects model performance on the testing dataset.

It can be noted from Figures 4.1 and 4.2 is the difference in the incorrect classifications. For the full-body test set, the algorithms keep misclassifying a 'not fall' sequence as 'fall' but for the ankle-only results, all algorithms misclassify a 'fall' event as 'not fall'.

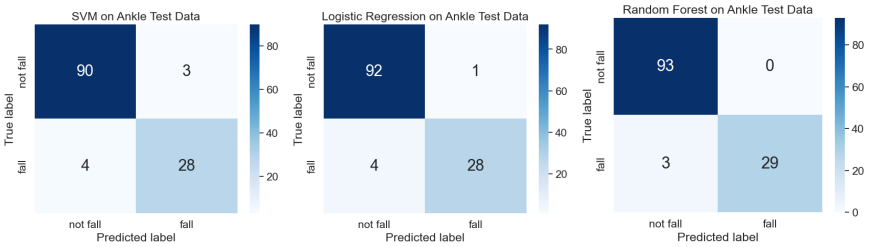


Figure 4.2 Confusion matrices for models trained on ankle-only data from the UMAFall dataset when tested on an unseen set also from the UMAFall project. From left to right: Support Vector Machine (SVM), Logistic Regression, and Random Forest.

4.2 Mixed datasets

In this part of the chapter, the results from combining the two data sources are presented. These results are summarized in Table 4.2, and visualized through confusion matrices, which provide a detailed view of the model's performance on the test sets. The first set of results corresponds to training and testing using only the ankle sensor data from the UMAFall dataset, providing a baseline for comparison.

The subsequent rows in the table illustrate the performance of the models when exposed to device data. These models were initially trained on the UMAFall dataset, which comprises 623 files, specifically utilizing the ankle sensor data. They were

then tested on 20 files from the device data. The device data was preprocessed to match the UMAFall data in terms of sampling rate and was segmented into 5-second windows that were centered around each detected fall event, but not overlapping.

The final row in the table represents a mixed training approach, combining the UMAFall (ankle sensors) and device datasets, and testing on the device data. The F1-scores in this scenario provide an understanding of how well the model generalizes across different data sources.

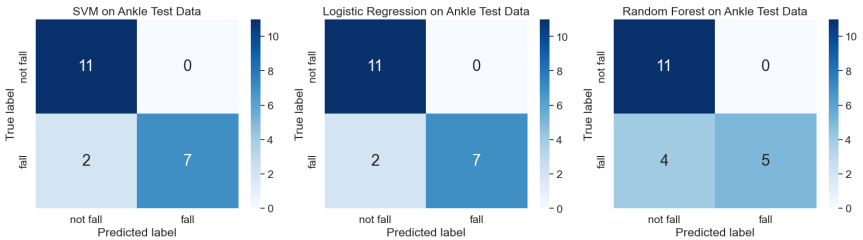
All preprocessing steps, including adjustments to window length, sampling rate, and the detailed methodology behind the alignment of data from both sources, are discussed in Chapter 3.

Table 4.2 Cross-validation (CV) scores and F1-scores for 'Not Fall' and 'Fall' classes when models were trained on different datasets. The UMAFall/UMAFall tests use ankle sensor data only, UMAFall/Device tests train on UMAFall ankle sensor data and test on device data, and Mixed/Device tests combine UMAFall and device data for training, tested on device data. F1-score measures model accuracy for each class, indicating the balance of precision and recall.

			F1-Score	
Train/Test Dataset	Model	CV Score	Not Fall	Fall
UMAFall/UMAFall	SVM	97%	96%	89%
	Logistic Regression	97%	97%	92%
	Random Forest	97%	98%	95%
UMAFall/Device	SVM	97%	92%	88%
	Logistic Regression	97%	92%	88%
	Random Forest	96%	85%	71%
Mixed/Device	SVM	97%	100%	100%
	Logistic Regression	97%	100%	100%
	Random Forest	96%	89%	67%

The confusion matrices shown in Figure 4.3 correspond to the results of the models trained on UMAFall ankle sensor data and tested on device data, as presented in the second case of Table 4.2. This test set included 11 non-fall instances and 9 fall instances, providing a balanced representation of both event types.

Figure 4.3 Confusion matrices for models trained on UMAFall ankle data when tested on device data. From left to right: Support Vector Machine (SVM), Logistic Regression, and Random Forest.



The confusion matrices in Figure 4.4 illustrate the results of training the models on a mix of UMAFall and device data. The test set for this scenario was smaller, comprising only 4 non-fall and 2 fall events.

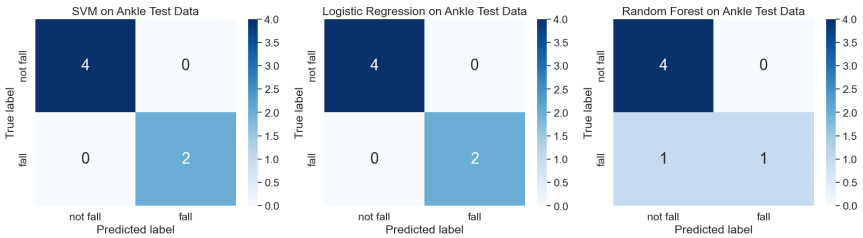


Figure 4.4 Confusion matrices for models trained with data from mixed sources when tested on device data. From left to right: Support Vector Machine (SVM), Logistic Regression, and Random Forest.

4.3 Device data

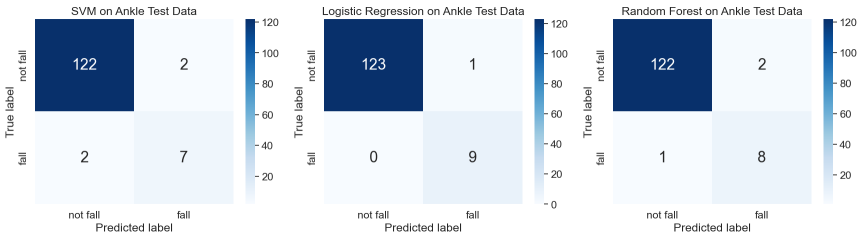
This section presents the outcomes of models trained and tested using only device data. Training models with device data alone involved the use of the sliding window method introduced in previous chapters, with specific parameters of a window size of 1500, step size of 250, and border size of 300. The application of these parameters along with stratified sampling during the division into training and testing sets resulted in a training set comprising 288 non-fall events and 22 fall events. The testing set consisted of 124 non-fall events and 9 fall events.

Table 4.3 and the confusion matrices in Figure 4.5 present the result of the cross-validation scores and F1-scores for the baseline models trained on 42 features extracted from the data windows.

Table 4.3 CV scores and F1-scores for the baseline models trained and tested on the device data with 42 features.

		F1-Score	
Model	CV Score	Not Fall	Fall
SVM	94%	98%	78%
Logistic Regression	95%	100%	95%
Random Forest	97%	99%	84%

Figure 4.5 Confusion matrices for the baseline models trained and tested on the device data. From left to right: Support Vector Machine (SVM), Logistic Regression, and Random Forest.



The top 10 features from the baseline models were identified and new models were created using only the top 10 features, a process often referred to as feature selection. The following table, 4.4 and confusion matrices in Figure 4.6 present the results of this round of model creation and evaluation.

Table 4.4 CV scores and F1-scores for the models trained with the top 10 features on the device data.

		F1-Score	
Model	CV Score	Not Fall	Fall
SVM	97%	99%	84%
Logistic Regression	96%	98%	75%
Random Forest	98%	98%	74%

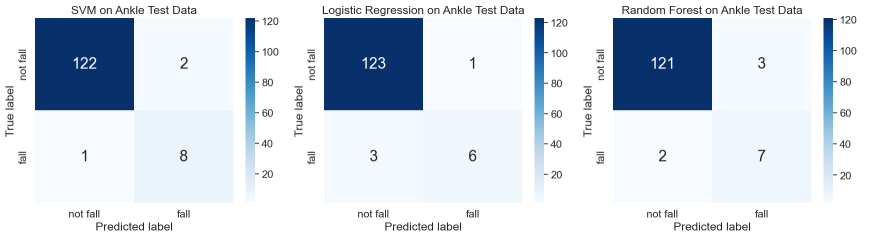


Figure 4.6 Confusion matrices for the models trained with the top 10 features on the device data. From left to right: Support Vector Machine (SVM), Logistic Regression, and Random Forest.

As a final step of model training and evaluation, pitch and roll estimations from a Kalman filter were added to the dataset, increasing the number of features from 42 to 56.

New models were trained with this expanded feature set, and the results are presented in Table 4.5 and Figure 4.7. Finally, feature selection was applied to this 56-feature set to identify the top 10 features. Again, new models were trained with this optimized feature set and results are displayed in Table 4.6 and Figure 4.8

Table 4.5 CV scores and F1-scores for the models trained on the 56-feature set.

		F1-Score	
Model	CV Score	Not Fall	Fall
SVM	96%	99%	89%
Logistic Regression	97%	100%	95%
Random Forest	97%	100%	100%

Already, an improvement can be observed in the F1-scores for the 'Fall' category, as compared to earlier results where models were trained without the addition of pitch and roll estimations.

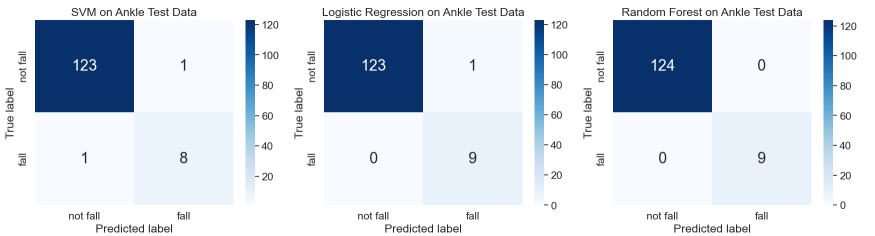


Figure 4.7 Confusion matrices for the models trained on the 56-feature set. From left to right: Support Vector Machine (SVM), Logistic Regression, and Random Forest.

Table 4.6 CV scores and F1-scores for the models trained on the top 10 features from the 56-feature set.

		F1-Score	
Model	CV Score	Not Fall	Fall
SVM	97%	99%	89%
Logistic Regression	98%	100%	94%
Random Forest	97%	99%	89%

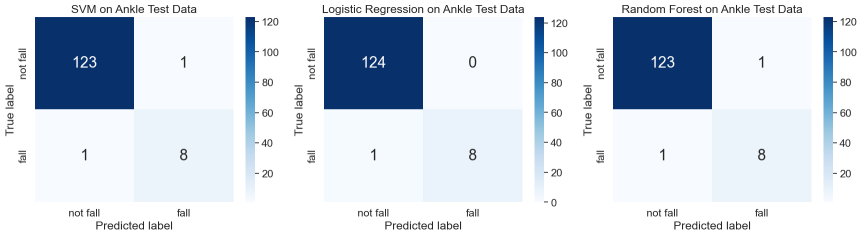


Figure 4.8 Confusion matrices for the models trained on the top 10 features from the 56-feature set. From left to right: Support Vector Machine (SVM), Logistic Regression, and Random Forest.

In choosing a model for deployment in devices with limited memory capacities, the model sizes have to be evaluated as well as the performance. Table 4.7 presents the sizes of each model when trained with 56 features compared to 10 features.

Table 4.7 Model sizes with different numbers of features

Model	**56 Features [KB]**	**10 Features [KB]**
SVM	15	4
LR	5	2
RF	158	150

5

Discussion

In this chapter, the results presented in Chapter 4 will be discussed and examined in relation to the goals and methods of this study. The discussion starts with an evaluation of the performance of the machine learning models across different datasets: UMAFall data, mixed data, and device data.

Following model evaluation, the limitations of this study are explored, and then a discussion on further research that could build upon the findings of this study. Finally, a conclusion of this study is presented.

5.1 Model Evaluation

The performance of the models varied depending on the dataset used for training and the features included in the model. Given that the primary use case for this project is to facilitate data collection aimed at future analysis and enhancements of the prosthetic device, achieving perfect accuracy is not paramount. This approach contrasts with applications where fall detection models trigger immediate alarms or emergency responses, where higher accuracy would be imperative.

UMAFall Data

Initially, models trained solely on the UMAFall dataset, without the inclusion of device data, achieved high accuracy with minimal adjustments. Several factors could explain this but this dataset was collected with fall detection in mind, making it quite optimal for this purpose.

When comparing results from models trained on the full-body sensor set to those trained on data from ankle sensors only, a slight decrease in cross-validation (CV) accuracy is observed. This could be explained by the reduced size of the training set as there are significantly fewer features to train, and less diverse information available in the ankle-only training set.

However, when the models trained on UMAFall data were tested on the device data, the accuracy dropped considerably, particularly, the F1-scores suggest a drop in the detection of fall events. This suggests that the models struggled to generalize from the UMAFall data to the device data, despite the application of downsampling the device data, along with other preprocessing steps to make the device data more compatible with the UMAFall data. This drop in accuracy highlights the challenges of applying models trained on one type of data to a different type of data, even after substantial preprocessing.

Mixed Data

Interestingly, when the models were trained on a mixed dataset of UMAFall and device data, the performance seemed to improve, with F1-scores racing up to 100% for both fall and non-fall events. This indicates that the models were able to learn useful patterns from the combination of the two datasets, and suggests that combining data from different sources could be a promising approach for improving fall detection in prosthetic devices.

However, it's important to consider the context of these results. The test set used in this particular round of evaluation was relatively small, including only 4 non-fall and 2 fall events. The small size of this test set most likely contributed to the high accuracy scores, as it may not fully represent the complexity and variability of real-world data. Therefore, while the results are promising, further validation with larger and more diverse test sets would be beneficial.

Device Data

For the device data, a sliding window technique was employed to create the training and testing sets. While this method allowed for an increase in both fall and non-fall sequences, the datasets remained notably unbalanced. The training set consisted of 288 non-fall sequences and 22 fall sequences, while the test set comprised 124 non-fall sequences and 9 fall sequences.

The class imbalance is manifested in the consistently lower F1-scores for fall events compared to non-fall events. Addressing this imbalance, possibly through techniques such as oversampling the minority class or undersampling the majority class, could enhance the model's predictive performance, particularly in accurately detecting fall events.

Pitch & Roll Estimations

After adding pitch and roll estimations to the dataset, the F1-scores for fall instances finally increased, particularly when using the full set of 56 features. The significance of features extracted from pitch and roll estimations became evident through

subsequent analyses. During feature performance evaluation, pitch and roll estimations consistently ranked among the top-performing features. This observation underscores their relevance and impact in enhancing the model's ability to detect falls effectively.

Incorporating these estimations not only expanded the dataset's feature set but also improved the predictive power of the machine learning models. By providing additional kinematic context, these features offer a richer, more detailed representation of user movements, which is crucial for accurate fall detection.

Model Sizes

The comparison of model sizes, as detailed in Table 4.7, reveals significant reductions in memory usage when the number of features is reduced. As microprocessor devices typically have limited computational resources and storage capacities, this is a critical factor to keep in mind. The Random Forest model, for instance, shows a minimal reduction in size, which suggests that while effective, it may not be the best choice for on-device deployment due to its relatively large size. In contrast, the SVM and Logistic Regression models demonstrate substantial size reductions with feature filtering, highlighting their potential suitability for embedded systems in medical devices where space and processing power are at a premium. These insights will guide the selection of the most appropriate models for deployment in prosthetic knees, ensuring that the device can operate reliably in real-world conditions without exceeding its hardware limitations.

5.2 Limitations

Data collection

A significant limitation of this study stems from the possibilities for data collection. The data used in this research is derived from simulated falls that often resemble kneeling more than actual falls. These simulations may not fully capture the diversity and unpredictability of real-world fall events. This could potentially impact the generalizability of the models trained on this data set. Real-world falls can be influenced by a variety of factors not present in the simulated fall events in this round of data collection. Factors that can influence real-falls include environmental conditions, the health of individuals, unexpected circumstances, and the surface of the ground.

Participant Diversity

The study might have been constrained by the limited availability of participants for data collection. Specifically, there was only one prosthetic knee user available for testing. This lack of diversity in the participant pool might limit the model's ability

to generalize to a broader population of prosthetic knee users. Each individual has a unique gait pattern and movement characteristics, as well as reactions to a potential fall event.

Sensor Accuracy

A potential limitation lies in the accuracy of the sensor measurements. While the sensors used in this study provided valuable data for detecting falls, they are subject to potential inaccuracies and drift over time, as any sensor does. These factors could introduce noisy data, which could impact the performance of the fall detection models, especially if the model is to be continuously monitoring sensor signals in real-time fall detection.

5.3 Future Work

The research presented in this thesis sets the stage for several future improvements and deeper investigation. The current fall detection system using IMU sensor data from prosthetic knees shows potential but also faces real-world application challenges. To enhance the robustness and applicability of the findings, the key areas identified for future work have been listed:

Obtaining Real-Life Fall Data

For a diverse dataset, the goal is to move beyond simulated data and incorporate data from real-life fall events. This could involve employing a fall detection model to monitor raw signals continuously in a user's everyday life. The model would then activate a data collection trigger if a fall is detected, ensuring that the sensor data leading up to the fall is stored. Confirming these events with the user ensures that the data represents true fall scenarios, which will provide valuable data to refine the detection algorithms.

Expanding User Diversity

To improve the generalizability of the fall detection system, future work will aim to include more users. By integrating the triggers system for data collection, it becomes feasible to gather data from real-world falls of a diverse group of users. Starting with the initial participant and gradually expanding to more users will help in developing a more robust model that can accommodate the variability in fall patterns.

Addressing Sensor Drift and Noise

Drift and noise are known problems of IMU sensors. Future research involves focusing on methods to reduce noise and compensate for the drift of sensor signals.

This could include improvements on both hardware and software, but better sensor data quality will improve the reliability and performance of the fall detection algorithms, especially for continuous long-term monitoring.

5.4 Conclusion

The exploration of machine learning for fall detection using IMU sensor data from prosthetic knees gives several critical insights and potential advancements in the field of prosthetic technology. This study has demonstrated the feasibility and effectiveness of Support Vector Machines, Logistic Regression, and Random Forests in distinguishing between fall and non-fall events with a high degree of accuracy.

An important objective was to create an efficient, real-time applicable solution tailored specifically for individuals using prosthetic devices. By focusing on simple, directly computable features extracted directly from raw IMU data, a system was developed that aligns closely with the real-time computational capabilities of prosthetic technology.

Through rigorous testing and validation, it was demonstrated that careful selection of features improved the accuracy and efficiency of the fall detection model. The process of selecting highly contributing features allowed the system to focus on computational resources, reducing processing power.

Ultimately, this research contributes to the ongoing efforts to enhance the safety and autonomy of prosthetic users, promising not just to improve their quality of life but also to provide a foundation for further innovation in integrated healthcare solutions.

Bibliography

Agrawal, S. and D. K. Sharma (2022). "Feature extraction and selection techniques for time series data classification: a comparative analysis". In: *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*. GLA University, Mathura, India.

Agresti, A. (2002). *Categorical Data Analysis*. Wiley, pp. 165–211. ISBN: 0-471-36093-7.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

BOSCH (2023). *Smart sensor: bno055*. URL: https://www.bosch-sensortec.com/products/smart-sensors/bno055/.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone (n.d.). *Classification and regression trees*.

Casilari, E. and J. A.Santoyo-Ramón (2018). "UMAFall: Fall Detection Dataset (Universidad de Malaga)". DOI: 10.6084/m9.figshare.4214283.v7. URL: https://figshare.com/articles/dataset/UMA_ADL_FALL_Dataset_zip/4214283.

Christ, M., N. Braun, J. Neuffer, and A. W. Kempa-Liehr (2018). "Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package)". *Neurocomputing* **307**, pp. 72–77. URL: https://www.sciencedirect.com/science/article/pii/S0925231218304843.

Cortes, C., V. Vapnik, and L. Saitta (1995). *Support-vector networks*.

Creylman, V., I. Knippels, P. Janssen, E. Biesbrouck, K. Lechler, and L. Peeraer (2016). "Assessment of transfemoral amputees using a passive microprocessor-controlled knee versus an active powered microprocessor-controlled knee for level walking". *BioMedical Engineering Online* **15**. ISSN: 1475925X. DOI: 10.1186/s12938-016-0287-6.

Geng, H. (2023). *Optimization of Sensor Data Processing Methods for Gait Tracking*. MA thesis. Aalto University.

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. O'Reilly Media, Inc. ISBN: 1492032646.

Jantawong, P., N. Hnoohom, A. Jitpattanakul, and S. Mekruksavanich (2021). "A lightweight deep learning network for sensor-based human activity recognition using imu sensors of a low-power wearable device". In: Institute of Electrical and Electronics Engineers Inc., pp. 459–463. ISBN: 9781665411974. DOI: 10.1109/ICSEC53205.2021.9684631.

Kalman, R. E. (1960). "A new approach to linear filtering and prediction problems". *Journal of Basic Engineering*.

Li, Q., R. Li, K. Ji, and W. Dai (2015). "Kalman filter and its application". In: *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pp. 74–77. DOI: 10.1109/ICINIS.2015.35.

Lin, C. L., W. C. Chiu, F. H. Chen, Y. H. Ho, T. C. Chu, and P. H. Hsieh (2020). "Fall monitoring for the elderly using wearable inertial measurement sensors on eyeglasses". *IEEE Sensors Letters* **4** (6). ISSN: 24751472. DOI: 10.1109/LSENS.2020.2996746.

Lindholm, A., N. Wahlström, F. Lindsten, and T. B. Schön (2022). *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press. URL: https://smlbook.org.

Nho, Y. H., J. G. Lim, and D. S. Kwon (2020). "Cluster-analysis-based user-adaptive fall detection using fusion of heart rate sensor and accelerometer in a wearable device". *IEEE Access* **8**, pp. 40389–40401. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2969453.

Össur: Power Knee™. URL: https://www.ossur.com/en-us/prosthetics/knees/power-knee.

Össur: Rheo Knee®. URL: https://www.ossur.com/en-us/prosthetics/knees/rheo-knee.

Peter, M., D. A. Aldo, F. Cheng, and S. Ong (2020). *Mathematics for machine learning*. URL: https://mml-book.com..

Powers, D. M. W. and Ailab (n.d.). *Evaluation: from precision, recall and f-measure to roc, informedness, markedness correlation*. URL: https://doi.org/10.48550/arXiv.2010.16061.

scikit-learn: Machine Learning in Python. URL: http://scikit-learn.org/.

Shawen, N., L. Lonini, C. K. Mummidisetty, M. V. Albert, K. Kording, and A. Jayaraman (2017). "Fall detection in individuals with lower limb amputations using mobile phones: machine learning enhances robustness for real-world applications". *JMIR mHealth and uHealth* **5** (10). ISSN: 22915222. DOI: 10.2196/mhealth.8201.

Szeghalmy, S. and A. Fazekas (2023). "A comparative study of the use of stratified cross-validation and distribution-balanced stratified cross-validation in imbalanced learning". *Sensors* **23** (4). ISSN: 14248220. DOI: 10.3390/s23042333.

Tahir, M., A. Moazzam, and K. Ali (2019). "A stochastic optimization approach to magnetometer calibration with gradient estimates using simultaneous perturbations". *IEEE Transactions on Instrumentation and Measurement* **68** (10), pp. 4152–4161. ISSN: 15579662. DOI: 10.1109/TIM.2018.2885624.

Tang, J., J. Xu, T. Tan, Z. Wang, Y. Zhou, and S. Jiang (n.d.). *Synthetic imu datasets and protocols can simplify fall detection experiments and optimize sensor configuration*. URL: http://ieeexplore.ieee.org.

Tanwar, R., N. Nandal, M. Zamani, and A. A. Manaf (2022). *Pathway of trends and technologies in fall detection: a systematic review*. DOI: 10 . 3390 / healthcare10010172.

Usmani, S., A. Saboor, M. Haris, M. A. Khan, and H. Park (2021). "Latest research trends in fall detection and prevention using machine learning: a systematic review". *Sensors* **21**:15. DOI: 10.3390/s21155134. URL: https://www.mdpi.com/1424-8220/21/15/5134.

Williams, G. (2011). *Data Mining with Rattle and R*. Springer New York. DOI: 10.1007/978-1-4419-9890-3.

Ziegler-Graham, K., E. J. MacKenzie, P. L. Ephraim, T. G. Travison, and R. Brookmeyer (2008). "Estimating the prevalence of limb loss in the united states: 2005 to 2050". *Archives of Physical Medicine and Rehabilitation* **89** (3), pp. 422–429. ISSN: 00039993. DOI: 10.1016/j.apmr.2007.11.005.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
May 2024

Document Number
TFRT-6236

Author(s)
Heiðrún Dís Magnúsdóttir

Supervisor
Stefán Páll Sigurþórsson, Össur, Sweden
Bo Bernhardsson, Dept. of Automatic Control, Lund University, Sweden
Kristian Soltesz, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Development and Evaluation of a Machine-Learning Based Fall Detection System for Prosthetic Knees

Abstract

This thesis explores the feasibility of integrating a fall detection system into microprocessor-controlled prosthetic knees using onboard sensors, with a focus on optimizing machine learning models for real-time operational efficiency within the limited computational capacities of such devices. Initial investigations utilized the public UMAFall dataset to gain insights into fall detection methodologies and preprocessing techniques. This study also examined the potential for combining the UMAFall dataset with device-specific data to enhance model robustness and performance.

Several machine learning models, including Support Vector Machines (SVM), Logistic Regression (LR), and Random Forests (RF), were evaluated for their ability to accurately detect falls and for their suitability in terms of computational footprint when deployed in a prosthetic device environment. The models were initially trained with 42 features, which increased to 56 after incorporating pitch and roll estimations into the device-specific dataset. This study further experimented with reducing the feature set to 10 core features to examine the impact on model size and efficiency.

Results indicate that feature reduction significantly decreases model size while maintaining high accuracy, with SVM and LR models showing the most substantial reduction in size, making them ideal candidates for on-device implementation. The Random Forest model, although effective in fall detection, demonstrated a less significant reduction in size, posing challenges for its practical deployment in prosthetic knees with strict hardware limitations.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-46

Recipient's notes

Security classification

http://www.control.lth.se/publications/