UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE FÍSICA

Development of an advanced environment adaptive algorithm to enable control of lower limb orthoses and exoskeletons

Rodrigo Miguel Francisco

Mestrado em Engenharia Física

Dissertação orientada por:
Prof. Dra. Guiomar Gaspar de Andrade Evans
Eng. Nikolaus Bätge

2025

Acknowledgment

I would like to begin by expressing my deepest gratitude to my coordinators, Prof. Dr. Guiomar Evans and Eng. Nikolaus Bätge, for their unwavering support and guidance throughout this project. Their expertise and thoughtful feedback helped shape both my work and my personal development, and I am truly thankful for the time they spent helping me refine my ideas.

I am also immensely grateful to my family, whose belief in my abilities gave me the courage to persist through difficult moments. Their encouragement was a constant source of motivation. My friends deserve special mention as well, not only for their hands-on help with data collection and testing, but also for offering creative ideas that enriched the AI models tuning. Their presence made this journey far more enjoyable and rewarding.

Finally, I would like to thank the wonderful team at Elysium Industries for the opportunity to collaborate on advancing the future of leg orthoses and exoskeletons. Their openness to new ideas and willingness to share insights were truly inspiring, and I hope our combined efforts will continue to drive meaningful innovations in assistive technology.

Abstract

Activity recognition plays a pivotal role in the evolution of wearable technologies and assistive devices, enabling them to adapt to user needs and improve mobility. Traditional methods often rely on multiple sensors or complex setups, limiting their practicality for everyday use. This dissertation introduces an approach to activity recognition utilizing Inertial Measurement Units (IMUs), specifically designed for seamless integration with an adaptive leg orthosis.

The research employs a Raspberry Pi based system to classify activities such as walking, stair climbing, stair descending, ramp ascending, ramp descending, and standing in real-time using advanced artificial intelligence techniques. By addressing common challenges in existing solutions, such as real-time processing limitations, power inefficiency, and integration difficulties, this study provides a streamlined and cost-effective alternative. The orthosis in development, created in collaboration with Elysium Industries pretends to leverage fluidic muscles as a control mechanism, replacing traditional motors to enhance responsiveness, adaptability, and energy efficiency.

Testing was conducted under varied conditions to validate the system's accuracy and reliability, achieving classification performance comparable to multi sensor setups. Ultimately, this research emphasizes the feasibility of accurate activity recognition and control implementation using a single sensor integrated with AI. The proof of concept demonstrates the potential for developing more efficient and adaptable orthoses, which could be extended to other wearable assistive technologies, thereby enhancing their practicality and usability in real world applications.

Keywords: Activity recognition, Inertial Measurement Unit, Leg orthosis, Fluidic muscles

Resumo

O reconhecimento de atividades físicas tem assumido um papel fundamental na evolução de tecnologias portáteis e dispositivos de assistência, permitindo-lhes adaptar-se às necessidades dos utilizadores e melhorar a sua mobilidade diária. Métodos tradicionais de reconhecimento de atividade, contudo, recorrem frequentemente a múltiplos sensores ou configurações complexas, tornando-se dispendiosos e pouco práticos para uso contínuo. Nesta dissertação, propõe-se um sistema de reconhecimento de atividades baseado em Unidades de Medição Inercial (IMUs), concebido para ser integrado de forma transparente num aparelho ortopédico adaptativo para membros inferiores.

A motivação principal para este trabalho deriva da necessidade de criar soluções mais leves, eficientes e responsivas para utilizadores com limitações de mobilidade. A maioria dos aparelhos ortopédicos ou exoesqueletos existentes recorre a motores elétricos de grande dimensão, o que aumenta não só o consumo energético mas também o peso e o ruído mecânico. Para contornar estas limitações, em colaboração com a Elysium Industries, foi concebido um sistema que substitui motores convencionais por músculos fluídicos (*fluidic muscles*). Estes atuadores pneumáticos apresentam uma alta relação potência-peso, aproximando-se do funcionamento de músculos biológicos.

No decurso do trabalho, construiu-se um modelo teórico para verificar se os músculos fluídicos conseguem gerar, em cada fase do ciclo de marcha, o momento articular necessário de forma a manter a trajectória pretendida da articulação abrangendo situações de caminhada em piso plano, subida e descida de escadas e locomoção em rampas com várias inclinações. O modelo considera a rigidez, a pré-carga e o comprimento nominal de cada músculo, bem como a sua pressão de operação a pressão interna de ar, aqui no intervalo de 3 a 8 bar. As simulações indicaram que uma combinação de dois músculos fluídicos, um com diâmetro maior, de 20 mm, e outro mais pequeno, de 10 mm podem abranger a maior parte das exigências de momento no joelho, mantendo uma margem de segurança e evitando sobrecargas.

Para aferir a capacidade de reconhecimento de atividades, recolheram-se dados de IMUs colocadas em diferentes pontos da perna (coxa e canela, ou apenas canela) de participantes saudáveis. A recolha de dados foi realizada com recurso a uma placa Raspberry Pi 4, à qual foram ligados vários sensores IMU, através de um multiplexador (MUS), utilizando o protocolo de comunicação I^2C (Inter-Integrated Circuit). Foram realizadas duas abordagens principais de recolha de dados: quatro IMUs, uma em cada perna (coxa e canela direita e esquerda), e com apenas uma IMU colocada na canela direita. A segunda abordagem, apesar de fornecer menos informação, permitiu um aumento significativo da taxa de amostragem (de cerca de 100 Hz para 200 Hz), pois eliminou a latência introduzida pela comutação do MUS.

Os dados de aceleração e velocidade angular obtidos foram posteriormente processados, segmentando-se em janelas de 25 ou 50 amostras. Para cada segmento, aplicaram-se técnicas de

limpeza de dados e, de seguida, foram exploradas duas linhas de modelação computacional. Um método de aprendizagem automática convencional (*Random Forest*) e algoritmos de aprendizagem profunda (redes neuronais convolucionais (CNN), redes neuronais do tipo *Long Short-Term Memory* (LSTM) e um modelo híbrido CNN–LSTM.

Numa fase inicial, testou-se o método *Random Forest* como modelo base, analisando-se diferentes configurações de segmentação (25 e 50 amostras) e número de IMUs (1 e 4). Verificou-se que, apesar de o uso de quatro IMUs e janelas maiores proporcionar uma maior exatidão, a diferença não era tão significativa a ponto de justificar a complexidade extra em termos de processamento e componentes físicos. Este resultado abriu caminho para a adoção de uma abordagem simplificada de uma única IMU, sem sacrificar drasticamente o desempenho dos modelos.

Posteriormente, implementaram-se modelos de aprendizagem profunda em PyTorch, uma biblioteca *open-source* de aprendizagem automática em Python, desenvolvida pelo Meta AI Research e optimizada para execução em GPU (Unidade de Processamento Gráfico). Implementaram-se tanto CNNs como redes LSTM, bem como um modelo híbrido CNN–LSTM, que conjuga a extracção de padrões locais (CNN) com a modelação temporal de sequências (LSTM). Os resultados mostraram que todos os modelos de aprendizagem profunda superaram o *Random Forest* em termos de exatidão de classificação global, sobretudo quando se utilizaram segmentos de 50 amostras. O modelo híbrido CNN–LSTM atingiu os melhores resultados, aproximando-se ou ultrapassando 95% de exatidão em algumas classes, demonstrando uma capacidade notável de distinguir entre atividades como subida de escadas e subida de rampas, que apresentam padrões de movimento semelhantes.

Para validar a aplicabilidade em tempo real, realizou-se um teste adicional com o melhor modelo (CNN–LSTM) em dois sujeitos: um cujos dados de locomoção já constavam do conjunto de dados de treino e outro completamente novo. Durante este teste, o sistema processava continuamente as leituras da IMU e, por um mecanismo que exige três previsões consecutivas idênticas, gerava um sinal de controlo que poderia ser usado para ativar ou desativar os músculos fluídicos. Observou-se que, embora existissem alguns erros de classificação pontuais, a maioria das transições entre atividades foi corretamente detetada. O sistema também introduziu uma categoria de atividade desconhecida para casos em que a confiança de classificação não atingia um certo limite, evitando ações potencialmente incorretas e priorizando a segurança.

Este estudo confirma a viabilidade de integrar reconhecimento de atividade baseado em inteligência artificial (IA) e músculos fluídicos num único aparelho ortopédico para os membros inferiores. O uso de uma única IMU reduz custos, simplifica a montagem e diminui o consumo de energia, tornando a solução mais atrativa para uso prolongado.

Em termos de perspetivas futuras, sugere-se a migração para microcontroladores de baixo consumo ou circuitos integrados dedicados de inferência neural, de modo a dispensar o Raspberry Pi e tornar o dispositivo ainda mais compacto. Planeia-se realizar testes práticos com os músculos fluídicos para confirmar as previsões teóricas discutidas, sendo possível obter uma visão mais clara do desempenho real dos músculos fluídicos e avaliar de forma consistente a viabilidade global da ortótese.

Em suma, esta dissertação demonstra a possibilidade de desenvolver um sistema de reconhecimento de atividades em tempo real, recorrendo apenas a uma IMU, que se integra num protótipo de ortótese de perna movida por músculos fluídicos. A abordagem proposta alia baixo custo, reduzida complexidade dos componentes físicos e lógicos, e uma ótima exatidão de classificação, constituindo um passo promissor para o avanço de dispositivos de assistência na área

da reabilitação e melhoria da mobilidade humana.

Palavras chave: Reconhecimento de atividades, Unidade de Medição Inercial, Ortótese, Músculos fluídicos

Table of Contents

List of Figures

List of Acronyms

AI Artificial Intelligence

CNN Convolutional Neural Network

CSV Comma-Separated Values

CUDA Compute Unified Device Architecture

DFT Discrete Fourier Transform

DOF Degree(s) of Freedom

GPU Graphics Processing Unit

I2C Inter-Integrated Circuit

IMU Inertial Measurement Unit

LED Light-Emitting Diode

LSTM Long Short–Term Memory network

RF Random Forest

SPI Serial Peripheral Interface

List of Tables

Chapter 1

Introduction

1.1 Motivation and Problem Statement

Wearable technology and assistive devices have advanced rapidly, creating new opportunities to improve the quality of life for individuals with mobility impairments. A crucial enabling technology is human activity recognition, which allows devices like prosthetic legs or orthoses to detect what the user is doing and respond appropriately. However, many traditional activity recognition approaches rely on multiple sensors or complex setups that, while effective in controlled settings, can be expensive, power-hungry, and impractical for everyday use. This gap between laboratory systems and real-world wearable solutions motivates the need for a more streamlined approach.

Recent research has therefore focused on reducing the number and size of sensors needed without sacrificing accuracy (N. Vu et al., 2022). Inertial Measurement Units (IMUs) have emerged as an attractive option in this context due to their compact size, affordability, and ability to capture motion data from accelerometers and gyroscopes. The challenge remains that even with a single IMU, the system must process data in real time under strict constraints: it should consume low power, introduce minimal latency, and integrate seamlessly with assistive devices such as leg orthoses.

This research proposes the development of a real-time activity recognition system using a minimal IMU setup, designed specifically for integration into an adaptable leg orthosis. The goal is to deliver accurate and reliable classification of locomotion modes using only the essential sensor data overcoming the bulky and costly systems. At the same time, the system must provide timely and precise output (e.g., detected activity signals) to the orthosis control mechanism so that assistance can be provided instantly as the user activity changes. In addition to the sensing aspect, this project explores an innovative actuation approach: using pneumatic artificial muscles (fluidic muscle) instead of traditional electric motors in the orthosis. Fluidic muscles contract and expand in a manner analogous to biological muscles, potentially offering a more responsive and energy-efficient actuation method for assistive devices (Ohta et al., 2018). By integrating an intelligent IMU-based recognizer with fluidic muscle actuation, the research aims to demonstrate the possibility of this novel, lightweight system that adapts to the user movements in real-time.

1.1.1 What Is a Leg Orthosis?

An orthosis (plural: orthoses) is an externally applied device designed to support or correct musculoskeletal function or to alleviate pain. In the context of lower-limb orthoses, these brace-like structures are fitted to the leg to control joint motion, redistribute load, and improve wearer stability during activities such as walking, stair climbing or standing. By modulating joint kinematics and kinetics, an orthosis can reduce muscular demand and protect vulnerable joints.

1.1.2 About Elysium Industries

This project is carried out in collaboration with Elysium Industries, a research-driven company specializing in next generation assistive devices. Elysium Industries develops lower-limb orthoses and exoskeletons that integrate novel actuators with real-time control algorithms. Their mission is to bring lightweight, energy-efficient and highly responsive gait assistance to clinical and everyday environments.

1.2 Objectives

The main objectives of this research are as follows:

1. **Evaluate the use of fluidic muscles as a possible actuation mechanism in the orthosis:** Investigate whether pneumatic fluidic muscle actuators can effectively replace traditional motor-driven actuation in the assistive device. This involves simulating the integration of fluidic muscles into the orthosis and assessing their possible performance in providing the necessary support for different activities. A key factor is the ability to deliver sufficient force at the knee joint during demanding activities.

2. **Develop a real-time AI activity recognition system with minimal sensors:** Implement and evaluate an AI-based method for real-time classification of human locomotion activities using data from as few inertial sensors as possible (ideally a single IMU). The method relies on the simulation and adaptation of existing deep learning algorithms, selected for their proven performance in activity recognition tasks. The targeted activities include level walking, standing, stair ascent, stair descent, ramp ascent, and ramp descent, corresponding to common modes an orthotic device should handle. The system must be optimized for rapid response and reliability, enabling the possible integration with a leg orthosis control loop.

By achieving these objectives, the expected outcomes include a validated prototype system and new insights into actuator selection for wearable assistive devices. Specifically, the research aims to demonstrate a working proof-of-concept that reliably recognizes user activities in real time and coordinates an orthosis possible control action (via fluidic muscle actuation) in a timely manner. Successful results will indicate that a simpler hardware configuration can still provide high accuracy and that fluidic muscles are a viable alternative to motors in terms of efficiency for this application. Hopefully this outcomes would pave the way for further studies and for more lightweight, cost-effective, and user-friendly orthotic systems in the future.

1.3 Scope of the Study

This study covers the design, implementation, and testing of an IMU-based activity recognition system within the context of a leg orthosis project provided by Elysium Industries. The work is largely focused on the software and algorithm development required to interpret sensor data and control the device in real-time. A Raspberry Pi single-board computer is used as the processing unit for the prototype. It collects data from the IMU sensor(s), runs the activity classification algorithms, and outputs a possible control signal based on the recognized activity. The prototype will be tested under certain conditions to verify that it can correctly and consistently identify the defined locomotion modes and promptly trigger the appropriate orthosis support actions.

It is important to clarify the limitations of the scope as well. While the system is developed with an eventual physical orthosis in mind, this dissertation does not involve the full integration of the activity recognition system into a commercial or clinical orthotic device. The interaction with the leg orthosis is demonstrated in a simulated or prototyped environment, meaning that the control signals from the recognition algorithm are tested on a model (or a simplified hardware setup) rather than on a final product. The focus is on proving the concept and evaluating performance metrics such as accuracy, latency, and reliability in recognizing activities. Additionally, the study concentrates on a specific set of activities and a particular orthosis design, it does not attempt to generalize to all possible movements or assistive devices. These boundaries keep the research scope manageable, allowing for the development of a functional activity recognition prototype and a thorough evaluation of fluidic muscle feasibility within the available time and resources. This initial work lays the groundwork for further development toward a market-ready product.

This dissertation is organized into four chapters to present the work in a structured and logical manner. Chapter 2 provides a review of the state-of-the-art in activity recognition using inertial sensors, with particular emphasis on AI algorithms and their application in assistive devices. Chapter 3 outlines the adopted methodology, covering sensor setup, data acquisition, algorithm selection, and the implementation of the system on a Raspberry Pi. Chapter 4 presents the results of the experimental evaluation, including performance metrics such as classification accuracy, latency, and system responsiveness. Finally, Chapter 5 summarizes the main findings, discusses the system's limitations, and proposes directions for future work toward integration into a complete orthotic solution.

Chapter 2

Literature Review

2.1 Activity Recognition

Activity recognition (AR) is the process of inferring human actions or movements from sensor data, and it forms the backbone of intelligent wearable systems. Over the past few decades, AR research has evolved through improvements in both sensing technology and data analysis techniques (Anguita et al., 2013; Huang et al., 2011). In this section, it is reviewed the historical development of the field and the advances in machine learning that have enabled modern AR systems.

2.1.1 Historical Context and Evolution

The concept of automatically recognizing human activities dates back several decades. Early efforts in the 1980s leveraged basic motion sensors such as accelerometers and gyroscopes to capture human movement data. These initial systems were often confined to specialized motion laboratories, the sensors and data acquisition hardware were large, expensive, and power-intensive, which made them impractical for everyday use (Huang et al., 2011).

In the 1990s, classical machine learning algorithms began to influence AR techniques. Researchers introduced methods such as decision trees, k-nearest neighbors, and support vector machines (SVMs) to classify sensor signals into distinct activities. Even with relatively low-power processors of the time, algorithms such as SVMs demonstrated the ability to distinguish basic activities (e.g., walking versus running) by analyzing features extracted from accelerometer data (Huang et al., 2011). However, due to limited computational capacity, these systems typically handled only a few activities with moderate accuracy, and real-time classification was very challenging.

The early 2000s brought significant technological shifts. The advent of sensor miniaturization, largely driven by advances in Microelectromechanical Systems allowed for the production of small, low-power accelerometers and gyroscopes. This led to the widespread adoption of IMU sensors in wearable devices, enabling data collection in real-world settings beyond controlled laboratories (N. Vu et al., 2022). With improvements in wireless communication and mobile computing, wearable AR systems started to be deployed in natural environments, introducing new challenges such as sensor noise and variability in human movement.

2.1.2 Advances in Machine Learning

As sensing hardware improved, parallel advances in machine learning greatly enhanced the capabilities of AR systems. Early studies often relied on manual feature engineering to extract statistical and frequency-domain features from sensor signals. These handcrafted features were then used in classifiers like SVMs and Random Forests (RFs), achieving respectable recognition rates for structured activities (Huang et al., 2011; Anguita et al., 2013).

However, the rise of deep learning in the 2010s transformed the landscape. Deep neural networks, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) networks, began to automatically learn hierarchical features from raw sensor data, eliminating the need for extensive manual feature design. CNNs have been particularly effective in extracting spatial patterns from time-series data, making them suitable for recognizing periodic activities such as walking or running (Ha et al., 2016). In contrast, LSTMs excel at modeling the temporal dependencies inherent in sequential sensor data, which is crucial for accurately identifying activities that involve a clear temporal structure (Wang et al., 2018).

Hybrid models that combine the strengths of CNNs and LSTMs have also emerged. These architectures first extract local features with CNN layers and then capture long-term dependencies with LSTM layers, achieving state-of-the-art performance in many AR benchmarks (Ariza-Colpas et al., 2022). Such hybrid approaches provide robust and scalable solutions for complex real-world scenarios, validating the methodology applied in this dissertation.

2.2 Machine Learning Models for Activity Classification Using IMUs

Given the focus of this dissertation on IMU-based recognition, it is essential to examine how previous studies have utilized IMU data with various machine learning models. A review of studies employing both traditional machine learning methods and deep learning architectures is presented, followed by a comparative analysis that highlights the strengths, limitations, and relevance of these approaches to the current work.

2.2.1 Traditional Machine Learning Approaches

Early research in human activity recognition relied on manual feature extraction from IMU signals followed by classical classifiers such as Support Vector Machines (SVMs), decision trees, and Random Forests (RFs). For example, (Anguita et al., 2013) extracted a 561-dimensional feature vector from smartphone accelerometer data in the UCI HAR dataset (using fixed sliding windows of approximately 2.56 seconds) and achieved approximately 96% accuracy with an SVM classifier. Similarly, studies using the WISDM dataset reported accuracies in the range of 88% to 91% for subject-independent evaluations, with models often reaching up to 98% in subject-specific settings (Weiss et al., 2011). Although these models benefit from interpretability and low computational cost, the requirement for manual feature engineering typically involves using fixed-length windows. These windows, if too short, may not capture enough temporal context, while longer windows introduce delays that negatively impact real-time performance.

2.2.2 Deep Learning Approaches

The rise of deep learning has revolutionized activity recognition by enabling automatic feature learning directly from raw sensor data. Convolutional Neural Networks (CNNs) have proven effective at extracting local spatial patterns from IMU signals. For instance, Ha et al., 2016 applied a deep CNN to raw accelerometer data with a window length typically ranging from 50 to 200 samples (corresponding to 50 ms to 200 ms at a 1 kHz sampling rate), achieving up to 95% accuracy on standard datasets. In parallel, Long Short-Term Memory (LSTM) networks have been successfully employed to model the temporal dynamics of sequential data. Ordóñez et al., 2016 introduced a hybrid DeepConvLSTM model that combined convolutional layers with LSTM units, reaching an F1-score of approximately 0.91 on the challenging Opportunity dataset. More recently, Xu et al., 2019 proposed a hybrid architecture integrating Inception modules and GRU layers, achieving accuracies between 93% and 96% across multiple datasets. Additionally, the study by H. T. T. Vu et al., 2022 compared classical machine learning and deep learning-based methods for locomotion mode recognition using a single IMU placed on the lower shank. Their experiments, which used segmentation windows as low as 50 samples, indicated that CNN and LSTM models outperformed traditional approaches in both accuracy and real-time feasibility. However, deep models often require larger datasets and greater computational resources, although recent studies have demonstrated that lightweight networks can be optimized for real-time, on-device inference (Peppas et al., 2020; Shakerian et al., 2023).

2.2.3 Comparative Analysis and Discussion

Table 2.1 summarizes key aspects of representative studies in the literature. In addition to dataset characteristics, sensor placement, and model architecture.

The table reveals that traditional machine learning models, such as those used in Anguita et al., 2013 and Weiss et al., 2011, benefit from low inference latency and interpretability but depend on manually engineered features extracted from relatively long sliding windows (typically around 2–3 seconds). Such long windows can capture adequate statistical properties but may introduce delays that are unsuitable for rapid control transitions in prosthetic applications.

In contrast, deep learning approaches demonstrate superior performance by automatically learning features from raw sensor data, even when using shorter window lengths (e.g., 50–200 samples, corresponding to 50–200 ms at high sampling rates). For example, the deep CNN presented in Ha et al., 2016 and the hybrid DeepConvLSTM in Ordóñez et al., 2016 achieve high accuracy while effectively modeling temporal dependencies. The study by H. T. T. Vu et al., 2022 is particularly relevant to the current dissertation, it compares multiple models using data from a single IMU on the lower shank and reports that CNN and LSTM models can achieve accuracies above 98%, thereby supporting the feasibility of a minimal sensor setup for real-time locomotion mode recognition in prosthetic devices.

Furthermore, recent works focused on lightweight deep models for real-time applications, such as Peppas et al., 2020 and Shakerian et al., 2023, illustrate that reducing the segmentation window to as little as 50 samples can still yield competitive accuracy while reducing computational latency. This is critical for applications where rapid detection of locomotion mode transitions is essential to adjust prosthetic control parameters in real time.

Table 2.1: Comparative Analysis of IMU-Based Activity Recognition Studies

Study	Dataset	Sensor Placement	Input/Features	Window Length	Model	Performance
Anguita et al., 2013	UCI HAR (30 subjects, 6 activities)	Smartphone	561 engineered features	2.56 sec	SVM	~96% accuracy (controlled lab settings)
Weiss et al., 2011	WISDM (36 subjects, 6 activities)	Smartphone	Statistical features (mean, variance, etc.)	2–3 sec	Random Forest	Up to 91% (subject-independent)
Ha et al., 2016	UCI HAR and others	Smartphone	Raw data	50–200 samples (500–2000 ms)	CNN	~95% accuracy
Ordóñez et al., 2016	Opportunity dataset (17 activities)	Multiple body-worn IMUs	Raw data	Varies	ConvLSTM (CNN + LSTM)	F1-score of ~0.91
Xu et al., 2019	Opportunity, PAMAP2, Smartphone datasets	Multiple IMU setups	Raw data	Varies	InnoHAR (GRU)	Up to ~96% accuracy
Peppas et al., 2020	WISDM	Smartphone	Raw data	50–100 samples (500–1000 ms)	CNN	94.2% accuracy on WISDM
Shakerian et al., 2023	Custom dataset (6 activities)	Single chest-mounted IMU	Raw data	50 samples (500 ms)	CNN	90.4% overall accuracy
H. T. T. Vu et al., 2022	Single IMU data (4 activities)	Single lower shank IMU	Raw data	50 samples (500 ms)	Various	CNN: 99.60% , LSTM: 98.68%

Overall, the literature suggests that while classical models offer the advantage of simplicity and low computational overhead, they are limited in their ability to capture complex temporal dynamics, particularly when short segmentation windows are required for real-time performance. Deep learning models, with their capacity for automatic feature extraction and efficient temporal modeling, provide a more promising approach for the goals of this dissertation. The insights from these studies, including the importance of segmentation window length, underscore the relevance of exploring lightweight CNN and LSTM architectures for robust, real-time IMU-based activity recognition in prosthetic control.

2.2.4 Models Adopted in This Dissertation

Four supervised classifiers were implemented, chosen to cover complementary modelling principles and to provide a clear speed accuracy trade off for on-device inference. Architectural hyperparameters and training details are given in Chapter 3, here, a small theoretical foundation of each model used in this work is made.

Random Forest (RF): A Random Forest is a large group of decision trees, each a model that makes a prediction by asking a sequence of simple yes/no questions about the input features, until it reaches a decision that all vote on the final answer. To make each tree different, we give every tree a slightly different copy of the training data and let it look at only a random subset of the input features when it splits. Because the trees are diverse, their combined vote is usually more accurate and less prone to overfitting than any single tree (Breiman, 2001).

Convolutional Neural Network (CNN): A Convolutional Neural Network is a layered model that learns to recognise small, meaningful patterns in the raw IMU signal by sliding a set of trainable filters over the data and combining their outputs. Each filter produces a feature map that highlights where its learned pattern occurs, stacking multiple convolutional layers allows the network to build up from simple patterns (e.g. sudden changes in acceleration) to more complex signatures (e.g. the rhythm of a walking step) while keeping the number of parameters manageable through weight sharing (Bevilacqua et al., 2019).

Long Short-Term Memory Network (LSTM): An LSTM is a type of recurrent neural network that reads the IMU data one sample at a time while carrying along a hidden memory. At each step it uses three small "gates", one to decide what new information to write into memory, one to decide what old information to forget, and one to decide what to output. This way it can remember important patterns over many time steps. This ability to selectively store and update information makes LSTMs especially good at detecting transitions in movement, such as the shift from walking to stopping (Ghojogh et al., 2023).

Hybrid CNN–LSTM: A Hybrid CNN–LSTM first applies convolutional layers to the IMU data to extract local movement patterns, then feeds the resulting sequence of pattern summaries into an LSTM that tracks how these patterns evolve over time.

2.3 Fluidic Muscles in Assistive Devices

Fluidic muscles, or pneumatic artificial muscles (PAMs), have emerged as a promising alternative to traditional electric motors in assistive devices due to their lightweight, compliant, and muscle-like characteristics. These actuators operate by inflating a flexible membrane, which contracts to generate force in a manner analogous to biological muscles.

Recent research has focused on integrating fluidic muscles into wearable devices to achieve smoother and more responsive motion. For example, Ohta et al., 2018 investigated the use of fluidic muscles in a multi-joint exoskeleton, demonstrating that their compliant nature can lead to more natural and adaptive movement.

In summary, the literature on fluidic muscles in assistive devices suggests that they offer significant advantages in terms of weight, compliance, and energy efficiency. These findings align with the methodology of this dissertation, where the integration of an IMU-based AR system with fluidic muscle actuation is proposed.

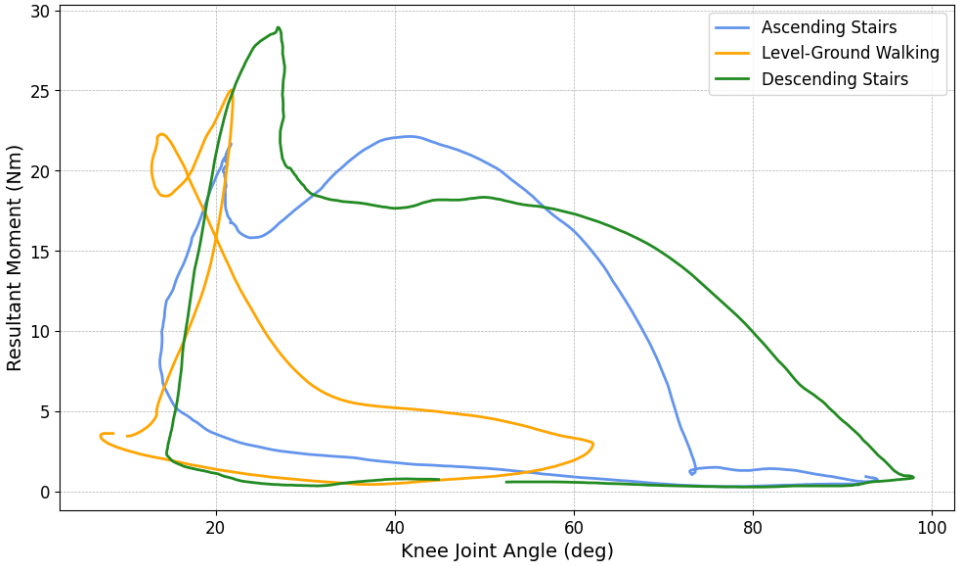
Chapter 3

Methodology

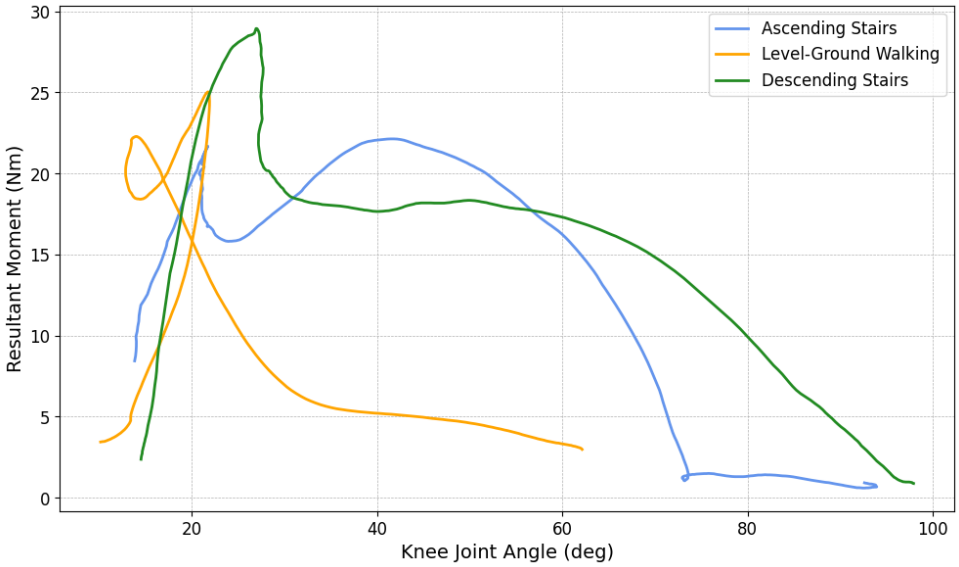
3.1 Optimal Fluidic Muscle Characteristics

To assist Elysium in determining the optimal fluidic muscle characteristics for further testing, an initial study was conducted using the theoretical orthosis model. This model employs two predefined springs, each with its own properties, such as stiffness values, preload values, and spring lengths, to calculate the resultant moment in Newton meters at the knee joint for various flexion angles. As the company provided only an executable script, the validated model it implements was adopted, re-deriving its internal calculations would be overly time consuming and beyond the scope of this work. Additionally, Elysium provided data from previous studies, which included moments measured at various knee joint angles during activities such as ascending stairs, descending stairs and level-ground walking for a healthy subject with a weight of 75 kg (Figure 3.1a). In order to analyse the phases of the gait cycle that demand greater knee joint moments and, consequently, higher physical effort, when orthotic support is most critical, the swing phase was excluded from the analysis (Figure 3.1b). The gait cycle refers to the sequence of movements from initial contact of one foot to the next contact of the same foot. It is commonly divided into two main phases: the stance phase, during which the foot is in contact with the ground, and the swing phase, during which the foot is lifted and moves forward. The knee moment refers to the torque produced at the knee joint, it reflects the mechanical demands placed on the joint during movement. For the purposes of this study, only those phases of the gait cycle involving greater mechanical demand, specifically those with higher knee moments, were considered relevant, as these are the moments when orthotic assistance provides the greatest benefit.

It is worth noting that in 3.1), as well as in later figures where joint moment is plotted directly against knee angle, some curves may appear to loop back or cross over themselves. This visual effect arises because the original measurements evolve in three dimensions (moment, angle, and time). During the gait cycle, the knee passes through the same angle at different instants while producing different moment values. When the time dimension is omitted and the data are projected onto a two-dimensional plot, these separate phases collapse onto the same plane, creating the impression of a knot or self-intersection. In a time-resolved representation, the curves would follow distinct trajectories and would not intersect.



(a) Knee moment over the full gait cycle for three activities.

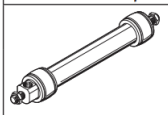


(b) Knee moment during stance (non-swing) phase only, for three activities.

Figure 3.1: Resultant knee joint moment data for each joint angle, provided by Elysium.

The initial task focused on investigating the dynamic behavior of fluidic muscles available from Festo fluidic muscles. The force generated by these fluidic muscles varies with internal diameter, nominal length and operating pressure. Figure 3.2 presents manufacturer specifications for multiple configurations, including maximum permissible contraction, pretension, operating pressure range and lifting force.

As an example, the behavior of a fluidic muscle with an inner diameter of 10 mm is presented in Figure 3.3. This figure illustrates the permissible force F [N] as a function of contraction h [%] relative to the nominal length, under different operating pressures. The exact mathematical model and equation used to generate these curves were not provided by Festo, the manufacturer of the muscles, despite direct request. Having access to them would have greatly simplified the work and removed the need for the approximations adopted.

Function	Version	Inside ⌀ [mm]	Nominal length [mm]	Lifting force [N]
Single-acting, pulling	Fluidic muscle with press-fitted connection			
		5	30 ... 1000	0 ... 140
		10	40 ... 9000	0 ... 630
		20	60 ... 9000	0 ... 1500
		40	120 ... 9000	0 ... 6000

Inside ⌀ [mm]	Max. permissible pretensioning	Max. permissible contraction	Operating pressure [bar]	→ Page/Internet
Fluidic muscle with press-fitted connection				
5	1% of nominal length	20% of nominal length	0 ... 6	10
10	3% of nominal length	25% of nominal length	0 ... 8	
20	4% of nominal length	25% of nominal length	0 ... 6	
40	5% of nominal length	25% of nominal length	0 ... 6	

Figure 3.2: Datasheet overview of Festo fluidic muscle characteristics, showing permissible contraction, pretensioning, operating pressures and lifting forces for various diameters and lengths. Source: Festo fluidic muscles.

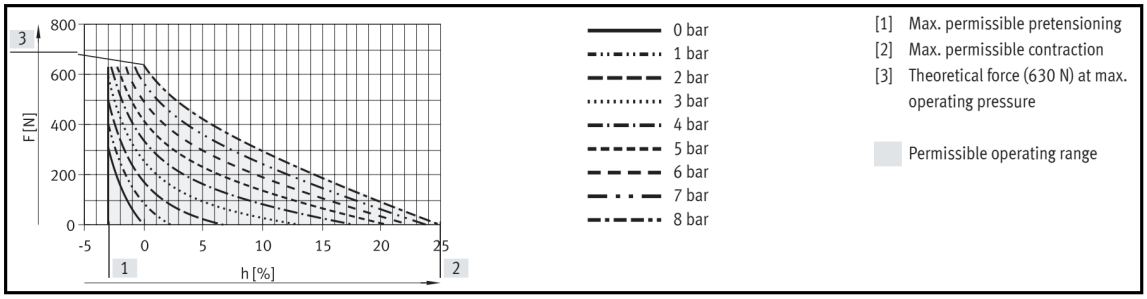


Figure 3.3: Example behavior of a fluidic muscle with a 10 mm inner diameter. Source: Festo fluidic muscles.

To identify the optimal physical characteristics for further testing, the behavior of fluidic muscles with diameters of 5 mm, 10 mm and 20 mm was evaluated across a range of operating pressures. Based on the data provided by Festo, a script was developed to model these muscles by approximating their behavior as that of a conventional spring. This approach enables the values to be directly used as inputs in the provided Elysium model, which requires parameters such as spring constants and spring lengths. Elysium provided the executable script that receives the input parameters and returns the corresponding results, without any underlying mathematical documentation. For this reason, the validated mathematical model already implemented in the script was adopted, as dissecting and fully understanding the math in detail would have been excessively time consuming.

This script determines the spring stiffness (k) and the intercept (b) for each fluidic muscle were determined using a linear regression, applied to the data provided (Figure 3.3). The behavior of each fluidic muscle was represented by the following linear relationship:

$$F = -k \cdot x + b \tag{1}$$

where:

- F is the force ([N]);
- k is the spring constant ([N/%], when x is expressed in percentage);

- x is the displacement or contraction (h [%]);
- b is the intercept, representing the force generated by the fluidic muscle when it is stretched to its nominal length ($x = 0$) under the given operating pressure.

Elysium's integrated black-box model was executed using the outputs of the spring-approximation script as inputs, namely muscle diameter, length, operating pressure, spring constant k and equilibrium length (the length at which $F = 0$). The model computes knee-joint moment profiles across the full spectrum of flexion angles by enforcing all fluidic muscle constraints and generating the corresponding moment curve.

Each computed profile was then evaluated against predefined minimum and maximum moment thresholds. Configurations that produced peak moments outside these limits or failed to generate sufficient force were discarded. The remaining scenarios were overlaid on the target moment profile (Figure 3.1b); Section 4.1 provides a detailed account of this selection process.

With the optimal fluidic muscle parameters identified (done in Section 4.1), the study proceeded to design and construct a dedicated hardware platform for movement data acquisition in augmented reality applications (see Section 3.2).

3.2 Hardware

The hardware was selected and assembled to ensure an easy integration of all necessary components for real-time data acquisition and processing. This section is divided into two subsections: the first provides an overview of the hardware components used (3.2.1), and the second outlines the assembly process (3.2.2), including the setup and integration of these components.

The system utilizes the SparkFun Qwiic ecosystem, which employs 4-pin JST-SH (Figure 3.4) connectors to simplify the connection between the development board and the sensors. This plug-and-play approach minimizes possible wiring issues and accelerates the prototyping process.

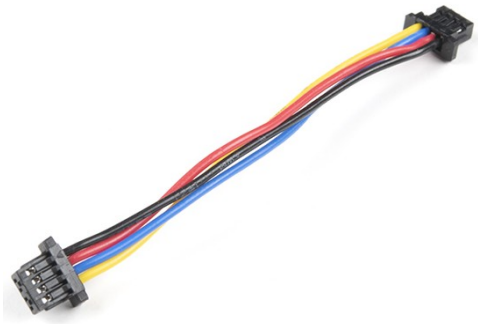


Figure 3.4: 4-pin JST-SH Qwiic connector. Source: SparkFun Qwiic.

3.2.1 Components

3.2.1.1 IMUs

The IMUs selected for this study are the ISM330DHC units from Adafruit (Figure 3.5). These 6-DOF sensors (3-axis linear acceleration and 3-axis angular velocity) were chosen for their cost-effectiveness, robust Python library support, and active community. IMU key specifications are summarized in Table 3.1.

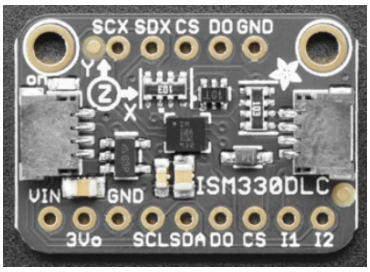


Figure 3.5: The ISM330DHC IMU sensor. Source: Adafruit LSM6DSOX and ISM330DHC Guide.

Table 3.1: Key specifications of the ISM330DHC IMU sensor

Parameter	**Specification**
Dimensions	25.6 mm × 17.8 mm × 4.6 mm
Accelerometer Range	±2/±4/±8/±16 g
Gyroscope Range	±125/±250/±500/±1000/±2000/±4000 dps
Accelerometer Update Rate	1.6 Hz to 6.7 kHz
Gyroscope Update Rate	12.5 Hz to 6.7 kHz
Communication Interfaces	SPI, I²C
Plug-and-Play (STEMMA QT)	4-pin JST-SH connector
I²C Address (modifiable)	Default 0x6A (changeable to 0x6B)

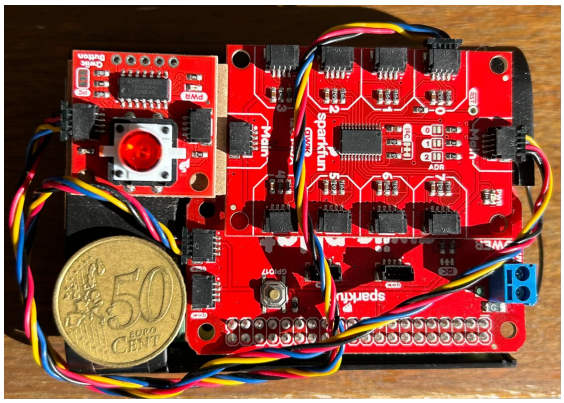
3.2.1.2 Raspberry Pi and Peripheral Components

The Raspberry Pi 4 Model B was chosen as the single-board computer for this project. The flexible programming options, and well-documented resources facilitated the integration for real-time data acquisition, processing, and possible control. To enhance portability and usability, the Raspberry Pi was housed in a casing, and a heatsink along with a cooling fan were installed to ensure proper performance under prolonged use.

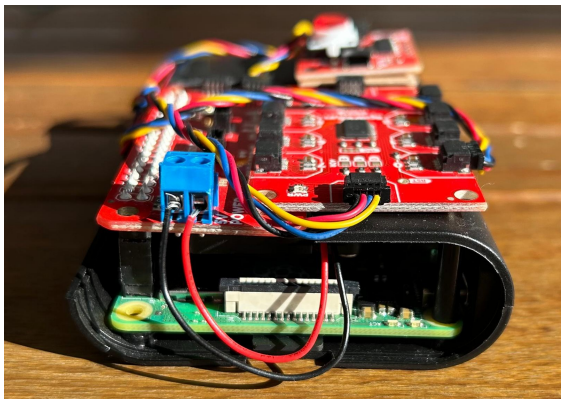
A Qwiic pHAT was added to the Raspberry Pi as an interface board, enabling communication with Qwiic devices via the Inter-Integrated Circuit (I²C) bus (GND, 3.3 V, SDA and SCL) on the Raspberry Pi. To address multiple sensor connections, a Qwiic MUX (multiplexer) was included. This is particularly necessary because the ISM330DHC IMUs have the same default I²C address, meaning that without a MUX, only one sensor could be connected to the I²C bus at a time. The Qwiic MUX allows us to connect multiple sensors by switching between different channels, enabling simultaneous data acquisition from all IMUs.

To improve functionality and usability, a Qwiic Button was integrated into the setup, allowing direct system interaction without the need for an external device. The button's built-in LED provides visual feedback, enabling users to monitor system status without relying on a screen or additional peripherals.

All of the mentioned components were integrated into a single, compact hardware device. Figure 3.6 illustrates the complete assembly, featuring the Raspberry Pi 4 Model B, Qwiic pHAT, Qwiic MUX, Qwiic Button and additional components, along with a 50 cent euro coin for size comparison.



(a) Top view of the Raspberry Pi and peripheral components setup.



(b) Side view of the Raspberry Pi and peripheral components setup.

Figure 3.6: Fully assembled hardware setup along with a 50 cent euro coin for size comparison.

3.2.1.3 Battery and Non-Electronic Components for Wearability

A compact, portable 30,000 mAh external battery was employed to power the system. This battery was chosen to support extended use, ensuring that the Raspberry Pi 4 Model B and connected IMUs (along with other peripherals) can operate continuously for long durations without needing frequent recharges. With a full charge, the system is expected to run continuously for approximately 18 hours. Additional non-electronic components, such as Velcro straps, a belt, and a belt-mounted pocket, were also integrated to improve wearability by securely mounting the battery, Raspberry Pi, and peripheral components, making the entire system easily portable.

3.2.2 Assembly Process

The assembly process was carried out as follows:

1. **Raspberry Pi and Peripheral Components:** Attaching the Qwiic pHAT to the Raspberry Pi's GPIO header, enabling communication between the Raspberry Pi and the Qwiic-enabled devices. The Qwiic MUX was then connected to the Qwiic pHAT to manage the connections to the IMUs.

2. **Connecting the IMUs:** Four ISM330DHC IMUs were connected to the Qwiic MUX using Qwiic I^2C cables.

3. **Sensor Placement on the Subject:** The IMUs were positioned on the subject's legs according to the study's design specifications. The velcro straps were used to attach the sensors to the designated locations. Figure 3.7 shows the placement of the IMUs on the subject, the IMU axes (labeled in the image) indicate the orientation of each IMU.

Once the hardware was assembled, a dataset was needed to train and test our future machine learning models (Section 3.3).

Figure 3.7: Placement of the IMUs on the subject's legs using velcro straps.

3.3 Data Collection and Processing

3.3.1 The Data

After an extensive search for a free, publicly available dataset, the most suitable option identified was the dataset by Camargo et al. This dataset includes locomotion data collected in various contexts, such as different stair heights, ramp inclinations, speeds on level-ground, and treadmill locomotion. However, due to differences in sensor placement and the need for full implementation in Python, it was decided to create a custom dataset tailored specifically to the objectives of this research.

The dataset used in this study comprises IMU-derived acceleration and gyroscope data collected from five male participants. The subjects were physically fit, aged between 20 and 23 years, with heights ranging from 171 cm to 179 cm and knee heights varying from 47 cm to 54 cm. Each participant performed six activities: level-ground walking, stair ascent, stair descent, ramp ascent, ramp descent, and standing with roughly equal samples for each. (Note: In the classification models, these six activities are labeled as `levelground`, `up_stairs`, `down_stairs`, `ramp_up`, `ramp_down`, and `standing`, matching the nomenclature used in subsequent chapters.)

Data was collected in three distinct environments: staircases, flat surfaces, and ramps. Specifically, measurements on staircases were obtained from five structures, two straight and three curved, each with varying step heights. For ramp environments, data was acquired from three ramps with inclinations ranging approximately between 7° and 12°. Finally, for level-ground walking, measurements were taken on surfaces with no inclination.

3.3.2 Data Collection with Multiple IMUs

To capture movement data, four IMUs were utilized, each positioned on a specific body segment: the right shank, right thigh, left shank, and left thigh, as shown in Figure 3.7. To optimize data acquisition speed, the I^2C address of two of the four IMUs were modified to 0x6B by bridging their solder jumpers. This configuration allowed simultaneous data collection from two IMUs, significantly reducing read latency.

3.3.2.1 System Initialization and Calibration

To ensure proper communication and data collection from the multiple IMUs, each sensor was assigned a unique I^2C address and linked to a specific MUX channel, configured as follows:

- **Right Shank**: MUX channel 1 (I^2C address 0x6A);
- **Right Thigh**: MUX channel 2 (I^2C address 0x6B);
- **Left Shank**: MUX channel 4 (I^2C address 0x6A);
- **Left Thigh**: MUX channel 5 (I^2C address 0x6B).

The MUX is organized into two banks of four channels each: bank A (channels 0-3) and bank B (channels 4-7). To keep all sensors for each leg grouped together, bank A was dedicated to the right leg and bank B to the left leg. Channel 0 (bank A) was reserved for the Qwiic Button, and channels 3, 6, and 7 remained unused. By assigning each leg's sensors to a single, contiguous bank of channels, the wiring layout becomes more intuitive.

During the calibration phase, which lasted approximately three seconds, the light-emitting diode (LED) on the button blinked to provide visual feedback to the user. During this interval, each Inertial Measurement Unit (IMU) was sampled N times per axis. Let $a_{\alpha,i}$ and $\omega_{\alpha,i}$ denote the ith raw accelerometer and gyroscope measurements along axis $\alpha \in \{x, y, z\}$. The offsets were computed as the arithmetic means:

$$\bar{a}_\alpha = \frac{1}{N} \sum_{i=1}^{N} a_{\alpha,i} \tag{2}$$

$$\bar{\omega}_\alpha = \frac{1}{N} \sum_{i=1}^{N} \omega_{\alpha,i} \tag{3}$$

These offsets were then applied in real time to correct subsequent raw readings:

$$a_{\alpha,\text{corr}} = a_{\alpha,\text{raw}} - \bar{a}_\alpha \tag{4}$$

$$\omega_{\alpha,\text{corr}} = \omega_{\alpha,\text{raw}} - \bar{\omega}_\alpha \tag{5}$$

The MUX sequentially enabled and disabled the corresponding channels for each sensor, and the corrected data were logged for further analysis, thereby minimizing errors caused by sensor drift or environmental factors. It is important to note that during calibration, the subject remained stationary on level-ground.

3.3.2.2 Data Logging Procedure for the Multiple IMU Setup

Following calibration, data logging was initiated via a button press. The system leveraged the unique I^2C addresses of the IMUs to concurrently acquire data from both the shank and thigh of a given leg. Communication with all four IMUs was managed by a MUX, which sequentially switched channels to poll each sensor. However, the MUX introduced a switching delay of approximately 0.004 seconds per channel, limiting the maximum sampling rate to 100 Hz.

For each reading, the following procedures were executed:

- A precise timestamp and sensor identification were recorded;
- Accelerometer and gyroscope data were simultaneously acquired from the shank and thigh;
- Calibration offsets, computed during initialization, were applied to the raw measurements;

- The corrected acceleration (X, Y, Z) and gyroscope (X, Y, Z) data, along with sensor identification, were recorded to a comma-separated values (CSV) file with corresponding timestamps.

The Qwiic Button's LED remained illuminated during logging, serving as a visual indicator of active data collection. Data logging continued until a subsequent button press terminated the session and closed the CSV file. This configuration effectively managed four IMUs on a single I²C bus using two distinct I²C addresses.

3.3.3 Data Collection with a Single IMU

3.3.3.1 System Initialization and Calibration

For single IMU data collection, the device was mounted on the right shank. This decision was made because the shank generally experiences less complex, high amplitude movements compared to the thigh, providing a more stable and representative signal for capturing leg dynamics.

Employing a single IMU in this configuration eliminates the need for MUX channel switching, allowing the sensor to operate at an increased sampling rate of 200 Hz, which provides finer temporal resolution. During initialization, the IMU was calibrated by collecting 600 stationary samples over a 3 second interval, from which average offsets for both accelerometer and gyroscope readings across all axes were computed. The Qwiic Button provided visual feedback via LED blinking during calibration.

3.3.3.2 Data Logging Procedure for the Single IMU Setup

After calibration, data logging commenced with a button press, for each sample:

- A precise timestamp was recorded;
- Accelerometer and gyroscope data were directly retrieved from the IMU;
- Calibration offsets were applied to correct the raw data;
- The adjusted acceleration (X, Y, Z) and gyroscope (X, Y, Z) data, along with the timestamp, were recorded in a CSV file.

As with the multi-IMU configuration, the LED remained illuminated throughout the data logging phase, indicating active data acquisition. Logging persisted until a subsequent button press safely terminated data collection and closed the CSV file.

3.3.4 Data Processing

The IMU data was processed in several steps to extract, clean, segment, and visualize the sensor readings. These steps prepared the data for use in machine learning models.

3.3.4.1 Data Extraction and Organization

The raw sensor data was first read from the CSV files using custom Python script. These scripts read the data sequentially, separating the accelerometer and gyroscope readings by sensor and by the X, Y, and Z axes. In addition, the scripts computed the average sampling rate by

analyzing the time intervals between consecutive data points. For the single IMU configuration, the average sampling interval was approximately 5 ms (corresponding to a sampling rate of about 200 Hz), and for the multi-IMU configuration, the average sampling rate was around 100 Hz. These results confirmed that the sensor data were being acquired reliably within the expected performance parameters.

3.3.4.2 Visualization of Sensor Data

After data extraction, the sensor data was plotted to provide an overall view of the signals and to detect any anomalies. Time series plots were generated for the accelerometer and gyroscope readings along the X, Y, and Z axes for each IMU. An observation from these plots was that the accelerometer data on the X-axis and the gyroscope data on the Z-axis were notably more consistent compared to other axes. This increased consistency is likely due to the orientation of the IMU during data collection the X-axis of the accelerometer and the Z-axis of the gyroscope tend to align with the dominant direction of motion during walking, capturing the periodic and less noisy aspects of the leg movement. Figure 3.8 shows a sample plot from the right leg IMUs, where the acceleration at the X-axis remains near zero when the foot is planted and exhibits positive and negative deflections as the leg lifts and lowers, while the Z-axis gyroscope values peak during the swing phase with angular velocities markedly larger on the shank than on the thigh, signals that reliably indicate step timing, cadence, gait symmetry, and correct sensor placement.

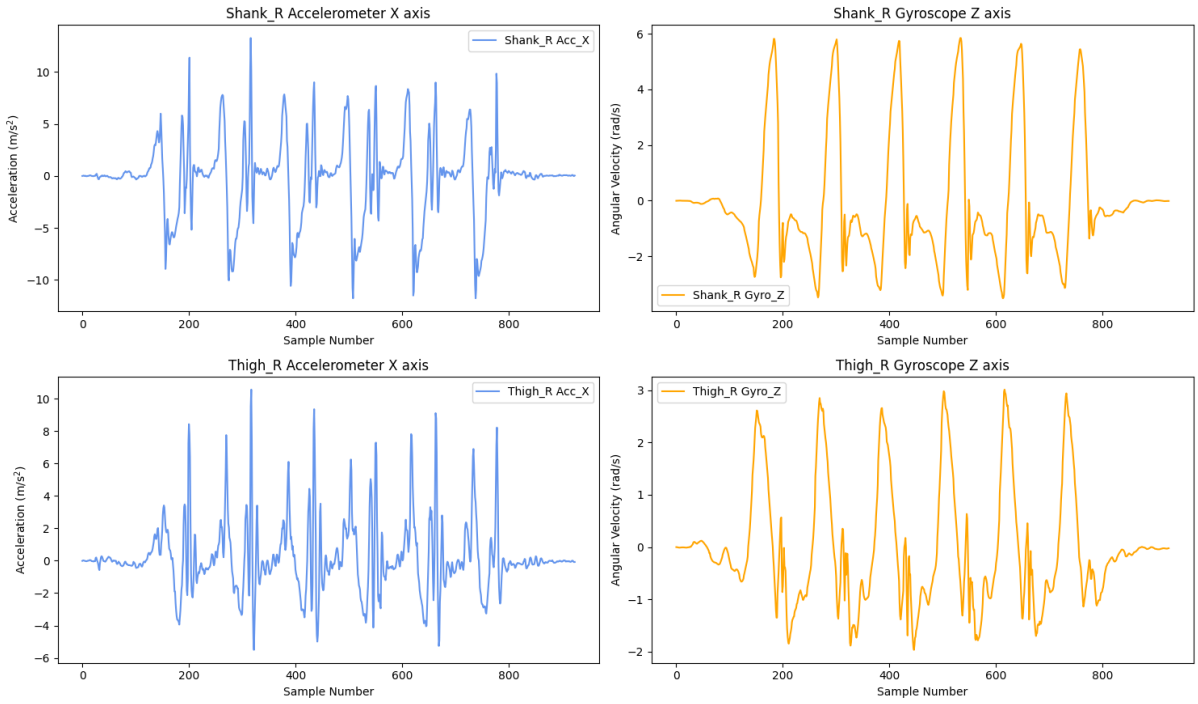


Figure 3.8: A sample plot of the accelerometer (X-axis) and gyroscope (Z-axis) data from the right leg's IMUs (shank and thigh).

3.3.4.3 Data Segmentation

To prepare the data for machine learning models, the continuous sensor data was divided into smaller segments. Two segment sizes were used, 25 and 50 data points which correspond to

time windows of approximately 0.125 s and 0.25 s for the single IMU configuration (operating at 200 Hz), and 0.25 s and 0.5 s for the multi IMU configuration (operating at 100 Hz). Each segment contains sequential accelerometer and gyroscope readings across the X, Y, and Z axes. The segmented data was then organized and stored in a dictionary structure, as illustrated in Figure 3.9.

```
{
    'activity_1': {
        'subject_A': {'Acc': {'X': [...], 'Y': [...], 'Z': [...]},
                      'Gyro': {'X': [...], 'Y': [...], 'Z': [...]}},
        ...,
        'subject_E': {'Acc': {'X': [...], 'Y': [...], 'Z': [...]},
                      'Gyro': {'X': [...], 'Y': [...], 'Z': [...]} }
    },
    ...,
    'activity_6': {
        'subject_A': {'Acc': {'X': [...], 'Y': [...], 'Z': [...]},
                      'Gyro': {'X': [...], 'Y': [...], 'Z': [...]}},
        ...,
        'subject_E': {'Acc': {'X': [...], 'Y': [...], 'Z': [...]},
                      'Gyro': {'X': [...], 'Y': [...], 'Z': [...]} }
    }
}
```

Figure 3.9: Dictionary structure that stores the segmented accelerometer and gyroscope data for each activity and subject.

Finally, the segments from the first two seconds and the last two seconds of each recording were removed, as these parts often contained irrelevant data related to the start or end of the activity such as the initial and final acceleration when the subject starts/stops an activity.

3.3.4.4 Handling Outliers and Lost Data Points

The IMU data acquisition system was designed to address scenarios where the IMU encountered issues during data collection. In such cases, zero values were recorded for all axes (X, Y, and Z) at the corresponding timestamp in the CSV file, representing missed or lost readings. To mitigate the effects of these data gaps, linear interpolation was applied to estimate the missing values, ensuring continuity and integrity of the dataset.

The missing data points were estimated using the following linear interpolation formula:

$$y = y_1 + \frac{(x - x_1)(y_2 - y_1)}{x_2 - x_1} \tag{6}$$

where:

- y: The estimated value for the missing data point in a specific axis (X, Y, or Z);
- x: The timestamp of the missing data point;

- x_1, x_2: The timestamps of the data points immediately before and after the missing point;
- y_1, y_2: The values of the data points at x_1 and x_2 for the corresponding axis (X, Y, or Z).

Additionally, each segment was also analyzed to detect and remove outliers using the *z-score method*. The z-score for a given data point y is calculated as:

$$z = \frac{y - \mu}{\sigma} \tag{7}$$

where:

- z: The z-score of the data point;
- y: The data point being evaluated (acceleration or angular velocity value);
- μ: The mean of the data within the segment:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} y_i \tag{8}$$

- σ: The standard deviation of the data within the segment:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \mu)^2} \tag{9}$$

- N: The total number of data points in the segment.

A data point was flagged as an outlier if its absolute z-score exceeded a chosen threshold ($|z| > 3$), indicating a significant deviation from the normal range of values. This threshold was chosen a priori based on the *three-sigma rule* in statistics, which states that approximately 99.7% of data points in a normal distribution fall within three standard deviations of the mean. Thus, any point with a z-score greater than 3 is highly unlikely to belong to the main distribution and can be confidently considered an anomaly.

Once detected, the outliers were removed and replaced using the same linear interpolation method described above, thereby maintaining continuity in the dataset.

This two-step process, handling missing data and detecting and removing outliers ensured that the dataset was clean and more robust, making it better for further machine learning applications.

3.4 Machine Learning Models Development

3.4.1 Conventional Machine Learning

As an initial step in exploring the dataset it was first implemented a Random Forest classifier as a baseline conventional approach, to gauge how well simple models perform on the dataset. This model was chosen based on its robustness and wide applicability to handle high-dimensional data with minimal hyperparameter tuning requirements.

The dataset used did not include ramp data. This was because the ramp data was collected after the initial data retrieval period, and the early tests focused on activities such as walking,

stair ascending, stair descending, and standing. Consequently, the baseline evaluation with the RF classifier reflects performance on this reduced set of activities.

The development and testing of the RF model followed a structured process using the scikit-learn library, as detailed below.

3.4.1.1 Feature Selection and Data Preparation

Both segmentation strategies were evaluated using data collected from a single IMU as well as combined data from four IMUs.

For each segment, statistical features were extracted to provide a summary of the data. The extracted features included time-domain features such as the mean (8), standard deviation (9), minimum value, maximum value, and median computed as:

$$m = \begin{cases} y_{(N+1)/2}, & \text{if } N \text{ is odd} \\ \frac{y_{(N/2)} + y_{(N/2+1)}}{2}, & \text{if } N \text{ is even} \end{cases} \tag{10}$$

where $y_{(i)}$ denotes the i-th value in the sorted segment, and N is the total number of data points in the segment.

In addition, a frequency domain feature was incorporated to capture the periodic and spectral characteristics of the IMU signals. The spectral energy of a signal segment is computed as:

$$E = \sum_{k=0}^{K-1} |X(k)|^2 \cdot \Delta f \tag{11}$$

where:

- E: the spectral energy of the segment;
- $X(k)$: the k-th coefficient of the Discrete Fourier Transform (DFT);
- $\Delta f = \frac{f_s}{N}$: the frequency resolution, where f_s is the sampling rate and N is the total number of points in the segment;
- K: the total number of frequency bins.

The spectral energy, along with time domain statistical features (mean, standard deviation, minimum, maximum, median), were consolidated into a single feature vector for each IMU. This process transformed each segment of the original data (25 or 50 data points) into a more compact and descriptive representation. A new dictionary structure was then created, where the extracted features were used to replace the raw segment data for each IMU enabling the RF model to effectively learn and classify different activities based on these extracted features.

3.4.1.2 Model Development and Testing

A `RandomForestClassifier` from scikit-learn was used to construct a classification model, chosen for its robustness and ability to handle high-dimensional data. The dataset was split into training and test sets using an 80/20 ratio to ensure that model performance was evaluated on unseen data.

Hyperparameter tuning, the process of systematically searching for the best model settings, was performed via grid search with 5-fold cross-validation to guard against overfitting given our

dataset size. The search space was defined based on both my own preliminary experiments and insights from the literature included:

- the number of trees in the forest (`n_estimators`);
- the maximum depth of each tree (`max_depth`);
- the minimum number of samples required to split an internal node (`min_samples_split`);
- the minimum number of samples required to form a leaf node (`min_samples_leaf`).

Model selection was guided by the weighted F1-score, defined as the harmonic mean of precision and recall, weighted by the number of true instances in each class, to balance false positives and false negatives when activity classes are imbalanced. The hyperparameter combination yielding the highest weighted F1-score was chosen for final evaluation.

3.4.1.3 Testing Across Different Configurations

Four RF models were created to assess how segment size and sensor configuration affect classification performance:

- segments of 25 samples with a single IMU;
- segments of 25 samples with four IMUs;
- segments of 50 samples with a single IMU;
- segments of 50 samples with four IMUs.

Each model followed the same training and evaluation process, using accuracy, precision, recall, and F1-score. Confusion matrices are square tables that summarize the number of correct and incorrect predictions by showing counts of true versus predicted classes for each activity. These were generated to provide insight into classification performance for each activity class. This setup enabled a direct comparison of the impact of different segment sizes (25 vs. 50) and sensor counts (1 vs. 4) on classification accuracy.

3.4.1.4 Best Hyperparameters for Each Configuration

Bootstrap sampling was enabled for all models (each tree is trained on a random sample drawn with replacement from the training set). Table 3.2 presents the optimal hyperparameters found for each configuration after grid search, note that `max_depth = None` indicates no fixed depth limit was applied to the decision trees. The final outcomes and a detailed analysis of these models are provided in Section 4.2.

3.4.2 Deep Learning Algorithms

Both the CNN and LSTM networks were implemented in PyTorch for flexible model development. Unlike TensorFlow, an open-source machine learning framework developed by Google, uses static computation graphs, meaning that the network architecture and data flow must be defined and compiled into a fixed graph before any data can be processed. In contrast, PyTorch

Table 3.2: Best hyperparameters for each configuration

Configuration	n_estimators	max_depth	min_samples_split	min_samples_leaf
1 IMU, Segment Size 25	200	20	5	2
1 IMU, Segment Size 50	300	20	5	2
4 IMUs, Segment Size 25	300	None	5	1
4 IMUs, Segment Size 50	100	None	5	1

dynamically constructs computation graphs at runtime, greatly simplifying debugging and experimentation. Additionally, GPU acceleration via the `torch.cuda` module significantly reduced training time. However, even when using the full dataset including all activities, in contrast to the reduced set used for the conventional RF model, a single grid search over a predefined range of hyperparameter values on an Nvidia RTX 3080 GPU with approximately 6144 CUDA (Compute Unified Device Architecture) cores required between four and six hours. This highlights the substantial computational demands of deep learning, even for relatively small datasets.

Due to these prolonged training times and based on insights from the conventional RF baseline, the decision was made to proceed with a single IMU setup. This choice simplifies the overall system by removing the need for multiplexer operations and enabling a higher sampling rate. Although the RF results (see Chapter 4) indicated that a multi IMU approach could yield higher accuracy, prioritizing real-time speed and hardware simplicity was deemed more critical. It was anticipated that the deep learning models would largely compensate for the lower sensor count.

3.4.2.1 Data Preparation

The deep learning models are designed to accept inputs as tensors. A tensor is a multidimensional array that efficiently organizes data for processing by neural networks. The input tensor is structured as:

$$(B, T, 6)$$

where:

- B represents the batch size, which is the number of samples processed simultaneously. This hyperparameter is chosen based on the available computational resources and the performance of the model during optimization;
- T denotes the number of time steps in each sample, with $T \in \{25, 50\}$. Either 25 or 50 data points per segment;
- 6 is the number of features per time step, corresponding to the three axis measurements of both the accelerometer and the gyroscope.

3.4.2.2 Hyperparameter Selection

A comprehensive grid search was performed for each model to identify the optimal hyperparameter configuration. The following outlines the search space and the number of unique combinations explored for each architecture:

CNN Model (64 combinations):

- **Number of Filters per Convolutional Layer:** $[32, 64]$ and $[64, 128]$;
- **Kernel Sizes:** 3 and 5;
- **Strides:** 1 and 2;
- **Dropout Rates:** 0.3 and 0.4;
- **Learning Rates:** 0.001 and 0.0001 (using an adaptive reduce on plateau schedule);
- **Batch Sizes:** 32 and 64.

LSTM Model (72 combinations):

- **Hidden Sizes:** 128, 256, and 512 units per layer;
- **Number of Layers:** 2 and 3;
- **Dropout Rates:** 0.3 and 0.4;
- **Learning Rates:** 0.001 and 0.0001 (using an adaptive reduce on plateau schedule);
- **Batch Sizes:** 16, 32, and 64.

Hybrid CNN-LSTM Model (128 combinations):

- **CNN Parameters:**
  - **Number of Filters per Convolutional Layer:** $[32, 64]$ and $[64, 128]$;
  - **Kernel Sizes:** 3 and 5;
  - **Dropout Rates (CNN):** 0.3 and 0.4.
- **LSTM Parameters:**
  - **Hidden Sizes:** 128 and 256;
  - **Number of Layers:** 2 and 3;
  - **Dropout Rates (LSTM):** 0.3 and 0.4.
- **Learning Rates:** 0.001 and 0.0001

3.4.2.3 Models Evaluation

For each hyperparameter combination tested across the CNN, LSTM, and Hybrid models, the model was trained over multiple epochs with early stopping was used to prevent overfitting in each training run, based on validation loss. A 10-fold cross-validation strategy was employed to ensure a robust evaluation and to reduce the variance caused by dataset partitioning. For each fold, the following steps were performed:

- The dataset was partitioned into training and validation subsets;

- The selected model (CNN, LSTM, or Hybrid) was trained on the training set;
- The modelperformance was then evaluated on the validation set;
- Metrics such as accuracy, loss, precision, recall, and F1-score were recorded.

The best model for each architecture was selected based on average validation metrics across folds, providing an unbiased assessment of the modelgeneralization capability.

All deep learning models were trained using the categorical cross-entropy loss, which measures how well the predicted probabilities match the true class. The loss becomes lower when the model assigns high probability to the correct activity and higher when predictions are wrong or uncertain. For a batch of N samples and C classes, the loss is defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log p_{ic},$$

where y_{ic} is the true label and p_{ic} the predicted probability for class c. While accuracy reflects the proportion of correctly classified samples, the loss is more sensitive to prediction confidence and small misclassifications. For this reason, the validation loss was used for early stopping.

3.4.2.4 Convolutional Neural Network Architecture

Based on the hyperparameter selection detailed earlier, the final CNN model for the single IMU configuration was constructed with a fixed architecture. The grid search was repeated for each segment size (25 and 50), evaluating several hyperparameters. The final model employs the optimal configuration as follows:

1. **Convolutional Layers:** The network accepts the input tensor (as described previously) and processes it through two 1D convolutional layers to extract local features from the time series data;
2. **Global Average Pooling (GAP):** The feature maps produced by the convolutional layers are aggregated using a Global Average Pooling layer, which computes the average of each feature map, reducing the spatial dimensions while preserving the essential information;
3. **Fully Connected Layers and Softmax:** The pooled features are then passed to a fully connected layer with 120 neurons and a Rectified Linear Unit (ReLU) activation, which simply replaces negative values with zero (see Equation 12). Finally, an output layer maps these features to the number of classes and applies the Softmax function to generate class probabilities (see Equation 13).

$$\mathrm{ReLU}(x) = \max(0, x) \tag{12}$$

$$\mathrm{Softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{13}$$

where:

- x: the activation input to the ReLU function;

- z_i: the logit (un-normalized score) for class i;
- $\sum_j e^{z_j}$: the normalization constant, where the index j runs over all output classes, ensuring the resulting probabilities sum to one and form a valid distribution.

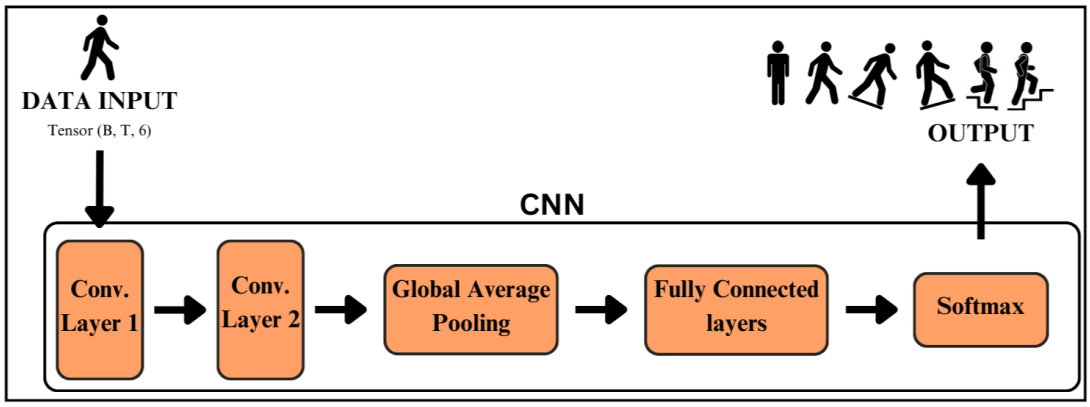


Figure 3.10: Block diagram of the CNN architecture for the single IMU configuration.

Figure 3.10 illustrates the block diagram of the final CNN architecture. The optimal hyperparameters, as determined by the grid search, are summarized in Table 3.3. Detailed performance results, including accuracy, loss and additional plots, are presented and discussed in Section 4.2.

Table 3.3: Optimal hyperparameters for the CNN model

Parameter	Segment Size 25	Segment Size 50
Learning Rate	0.001	0.001
Batch Size	32	64
Kernel Size	3	3
Stride	1	1
Dropout Rate	0.4	0.4
Filters (Layer 1)	32	32
Filters (Layer 2)	64	64

3.4.2.5 Long Short Term Memory Architecture

Following the previously mentioned hyperparameter selection process for each segment size (25 and 50), the final LSTM model is constructed as follows:

1. **LSTM Layers:** The model accepts the input tensor and processes it through a stacked LSTM network. In this many to one configuration, only the final hidden state at the last time step is used to represent the entire sequence;

2. **Fully Connected Layers and Softmax:** The final hidden state is then passed to a fully connected layer with 120 neurons and ReLU activation (Equation 12). Finally, an output layer maps these extracted features to the number of classes, using a *Softmax* activation (Equation 13) to produce probabilistic predictions.

Figure 3.11 illustrates the final LSTM architecture. The optimal hyperparameters, as determined by grid search, are summarized in Table 3.4. Model performance metrics are discussed in Section 4.2.

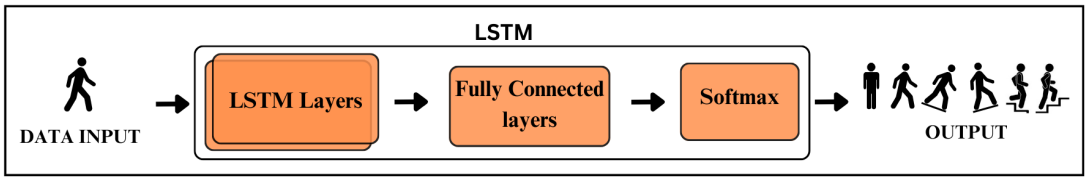


Figure 3.11: Block diagram of the LSTM architecture for the single IMU configuration.

Table 3.4: Optimal hyperparameters for the LSTM model

Parameter	Segment Size 25	Segment Size 50
Hidden Size	512	256
Number of Layers	2	2
Dropout Rate	0.4	0.3
Learning Rate	0.001	0.001
Batch Size	32	32

3.4.2.6 Hybrid CNN-LSTM Architecture

The hybrid architecture combines both the Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to leverage the strengths of both spatial and temporal feature extraction. In this model, the CNN component acts as an automatic feature extractor converting raw sensor sequences into a higher level sequence of features, and the LSTM component captures temporal dependencies from the sequential feature maps generated by the CNN. Once again the grid search was repeated for each segment size (25 and 50) and evaluated several hyperparameters. The final optimal hybrid model configuration is as follows:

1. **CNN Component:**
   The input tensor is processed through the CNN module, which consists of:
   - 1D convolutional layers to extract spatial features;
   - A dropout layer is applied after the convolutional layers to mitigate overfitting.

   The output feature maps are then flattened and organized as a sequence to be fed into the LSTM component;

2. **LSTM Component:**
   The sequential data from the CNN is processed through a stacked LSTM network to capture temporal dependencies;

3. **Fully Connected Layers:**
   The final hidden state from the LSTM is passed through a fully connected layer with 120 neurons and ReLU activation (Equation 12). A dropout layer is then applied, and finally, the output layer maps the features to a number of neurons equal to the number of classes, once again the *Softmax* (Equation 13) generates the probabilistic predictions.

Figure 3.12 shows the block diagram of the hybrid CNN-LSTM architecture. The optimal hyperparameters, as determined by the grid search, are summarized in Table 3.5. Interestingly, the hybrid model favored a larger number of filters (128 and 256) than the stand alone CNN did.

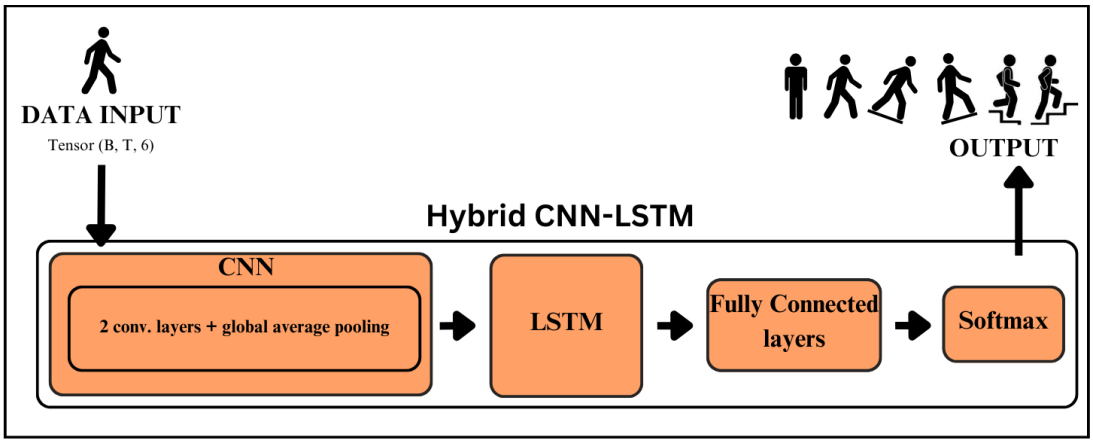


Figure 3.12: Block diagram of the hybrid CNN-LSTM architecture.

This could be because the hybrid architecture can utilize a richer spatial representation before the LSTM stage. Model performance metrics are discussed in Section 4.2.

Table 3.5: Optimal hyperparameters for the Hybrid CNN-LSTM model

Parameter	**Segment Size 25**	**Segment Size 50**
Learning Rate	0.001	0.001
Batch Size	32	32
CNN Component:		
Kernel Size	3	3
Stride	1	1
Dropout (CNN)	0.4	0.4
Filters (Layer 1)	128	128
Filters (Layer 2)	256	256
LSTM Component:		
Hidden Size	256	128
Number of Layers	2	2
Dropout (LSTM)	0.3	0.3

Chapter 4

Results and Discussion

4.1 Fluidic Muscle Selection

In this study, many two fluidic muscle combinations were tested to see how well they could produce the knee joint moments needed for walking, ascending stairs, and descending stairs. All possible combinations of diameters (10 mm and 20 mm) and lengths (80 mm to 170 mm in steps of 10 mm) were explored, and simulations were run to generate moment curves for each combination. Each curve was checked against maximum force, length, and moment constraints (10 Nm to 30 Nm), and only those sets with more than five valid curves were selected for further analysis. The selected sets were then evaluated for three different activities individually, focusing on how well the simulated moment matched the target moment at specific knee angles for each activity. The smallest permissible difference that satisfied all constraints was 7 Nm, although increasing the difference to 10 Nm allowed more valid curve scenarios. This approach made it possible to quickly compare different muscle sizes and pressure ranges to find a setup that stays within design limits while still covering the required moment range for the knee joint. Using one larger diameter fluidic muscle (H) and one smaller diameter fluidic muscle (U) yielded the best results. Specifically, the chosen configuration was:

- Main fluidic muscle (H): diameter 20 mm, length 170 mm;
- Secondary fluidic muscle (U): diameter 10 mm, length 170 mm.

Both muscles operate effectively at pressures between 3 bar and 8 bar, creating enough moment to support knee movement without exceeding safe limits. The larger diameter muscle (H) provides most of the force needed for knee assistance, while the smaller diameter muscle (U) refines the assistance, allowing adaptation to different knee angles. Longer lengths offer a wider range of work and higher overall moment values, which is why 170 mm was chosen. Adjusting the pressure in each muscle shapes the overall moment to match the demands of walking, stair ascent, or stair descent.

Figure 4.1 compares the selected fluidic muscle configuration against the ideal target moment profiles (colored lines). Both the reddish lines and the black lines show the achievable knee moment limits at different pressures. In particular, the red curves highlight the maximum moment attainable for each activity under our previously defined constraints, thereby indicating feasible pressure values for walking, stair ascent, and stair descent. As can be observed, the chosen muscle diameters and stroke lengths nearly span the full required moment range for all

three activities. For simplicity, only discrete pressure values were tested, pressure is inherently continuous, so many intermediate combinations remain untested and could further extend the achievable moment range. The visible discrepancies between our model curves and the ideal target profiles stem from our simplification of each fluidic muscle at a fixed pressure as a linear spring. Nonetheless, despite these modeling simplifications, the selected muscle combination meets the key moment requirements and remains within safe operating pressures.

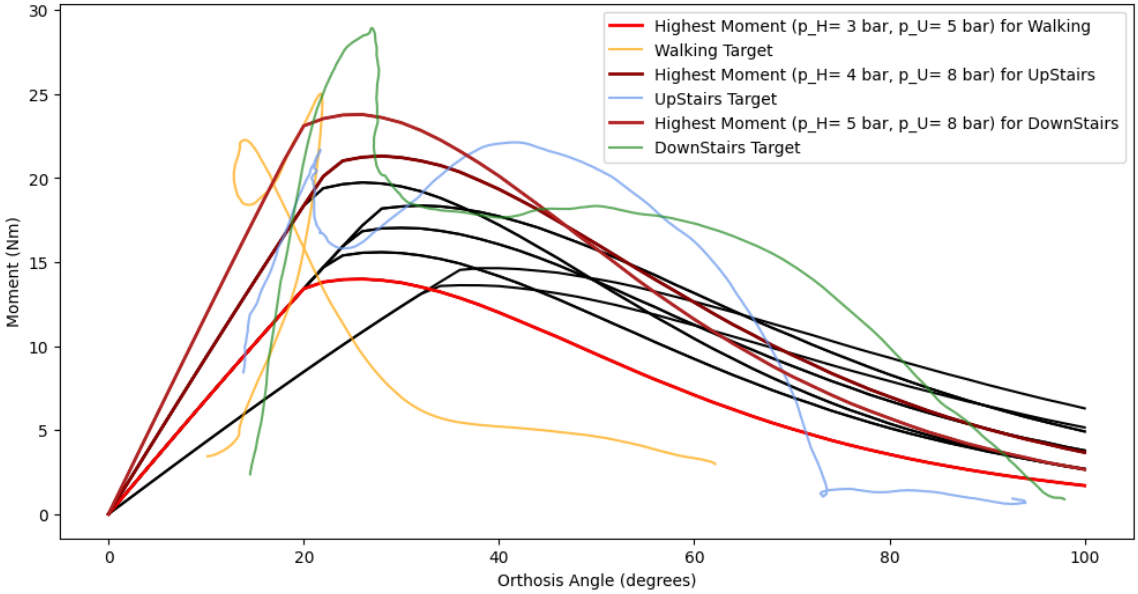


Figure 4.1: Knee joint moment data at various knee flexion angles using the chosen fluidic muscle combination (H: diameter 20 mm, length 170 mm; U: diameter 10 mm, length 170 mm). The black lines show the model knee moment output at different pressures. The red lines show the highest possible moment at each activity. The colored curves are the target moment profiles for walking, ascending stairs, and descending stairs.

4.2 Performance Analysis of the Machine Learning models

4.2.1 Conventional Machine Learning

The performance of the RF classifiers was evaluated across four configurations, varying the segment size (25 vs. 50 data points) and the number of IMUs (1 vs. 4).

To ensure robustness, 5-fold cross-validation was applied. Each fold involved training on four subsets and testing on the remaining subset, ensuring that the model was evaluated on multiple data splits. The final metrics and confusion matrices represent the averaged results across all validation folds. Figures 4.2 through 4.5 illustrate the normalized confusion matrices for the four configurations.

By analyzing the Figures it is possible to verify that The most common misclassifications occur in the classification of down_stairs and up_stairs, both of which are frequently confused with levelground. This suggests that these activities share similar signal patterns, making differentiation more challenging. In contrast, standing is consistently classified with perfect or near perfect accuracy in all configurations, as indicated by its high True Positive (TP) rate. Additionally, the classification of levelground improves as the number of IMUs and segment size increase, leading to higher TP values.

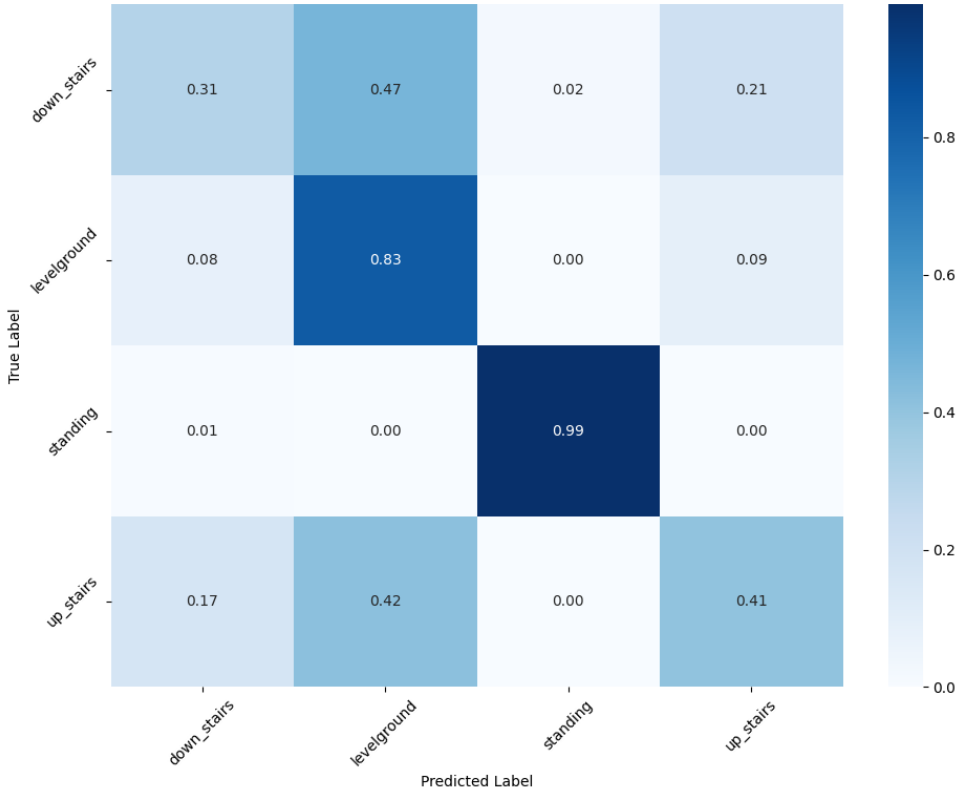


Figure 4.2: Normalized confusion matrix for 1 IMU with segment size 25 (5-fold averaged).

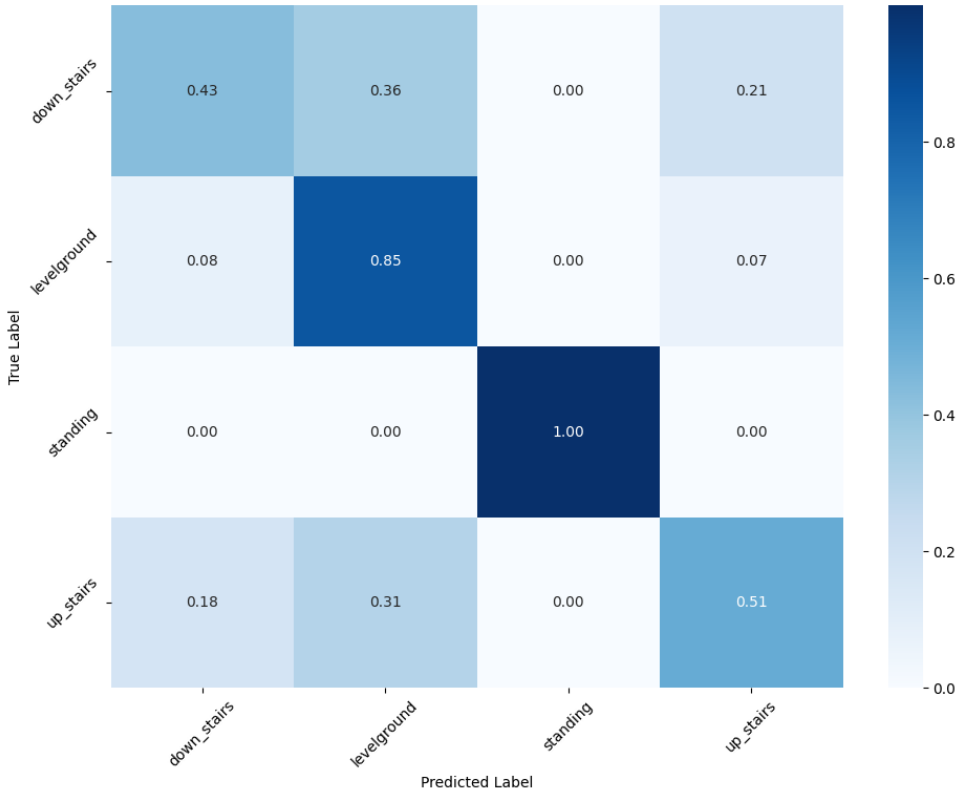


Figure 4.3: Normalized confusion matrix for 1 IMU with segment size 50 (5-fold averaged).

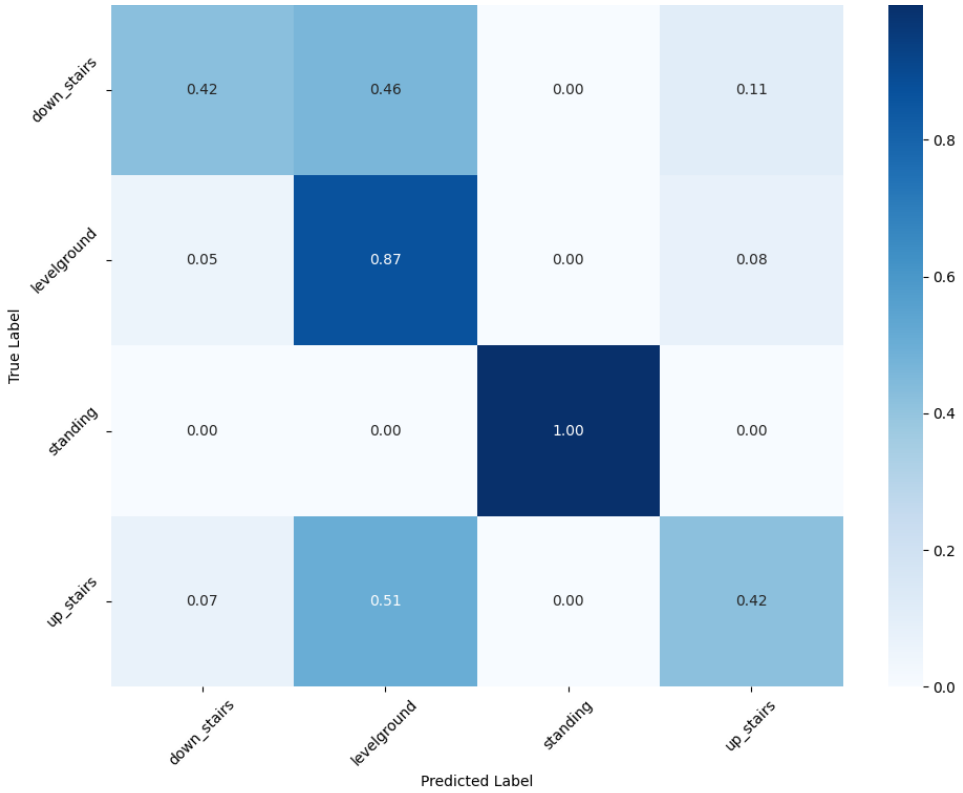


Figure 4.4: Normalized confusion matrix for 4 IMUs with segment size 25 (5-fold averaged).

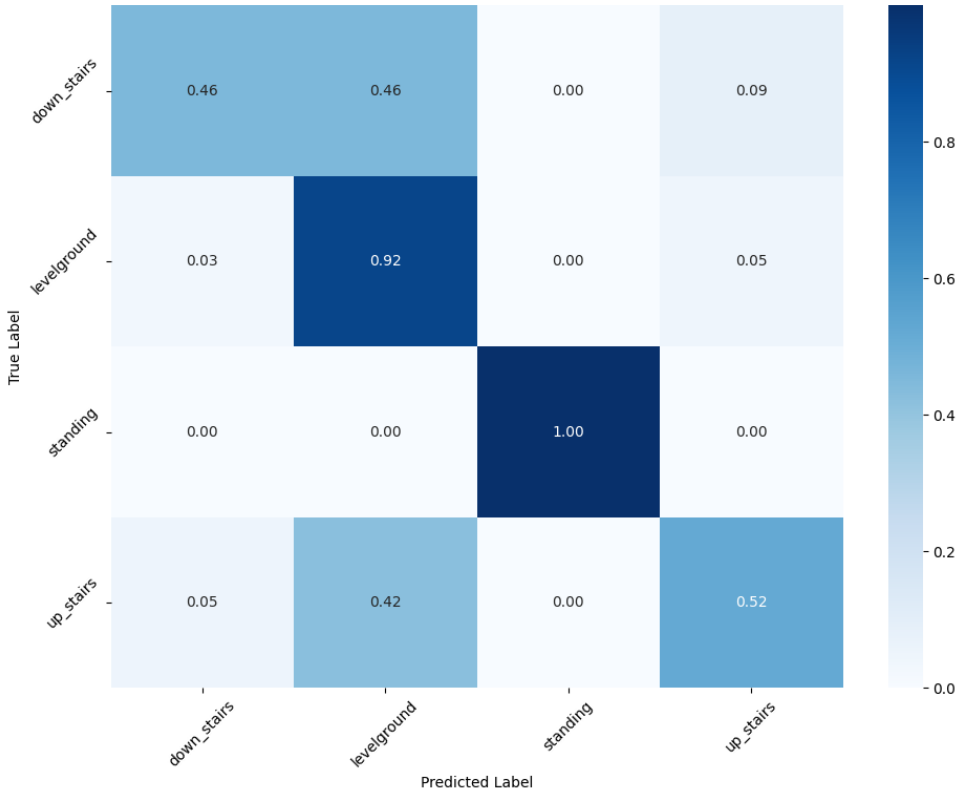


Figure 4.5: Normalized confusion matrix for 4 IMUs with segment size 50 (5-fold averaged).

4.2.1.1 Discussion of Results

Table 4.1 presents the mean accuracy and F1-score for each configuration, obtained by averaging the results across the five folds.

Table 4.1: Performance metrics of RF classifiers (5-fold cross-validation)

Config #	IMUs	Seg. Size	Sampling Time (s)	Accuracy (%)	F1 (%)
1	1	25	0.125	65	63
2	1	50	0.250	73	72
3	4	25	0.250	70	69
4	4	50	0.500	81	80

The results confirm that, as expected, increasing the number of IMUs and the segment size improves classification accuracy. The best performing configuration, used four IMUs and 50 data points per segment, it achieves the highest accuracy (81%) and F1-score (80%).

Interestingly, the single IMU configuration with a 50 sample segment size emerges as a particularly compelling alternative. As shown in the normalized confusion matrices, this was the only setup in where, for every activity, the number of correctly classified instances (True Positives) was always greater than any individual False Positive count from misclassified activities. It offers a reasonable accuracy (73%) while maintaining a lower hardware requirement, making it a practical choice for real-time applications. Consequently, the subsequent models will be developed exclusively using this single IMU data.

4.2.2 Deep Learning Algorithms

For each model (CNN, LSTM, and Hybrid CNN–LSTM), the number of training epochs was determined by monitoring both the training and validation loss curves. The training loss consistently decreased, indicating that the models were learning meaningful patterns from the data. In contrast, the validation loss typically decreased at first and then stabilised or began to rise, signalling the onset of overfitting.

Because the loss is sensitive not only to whether a prediction is correct but also to the confidence of that prediction, the validation loss was used as the criterion for early stopping. Training was halted when the validation loss no longer improved, ensuring that the models did not memorise the training data and retained good generalisation to unseen samples.

For completeness, the loss and accuracy curves, confusion matrices, and final test accuracies for each model are presented in the following subsections. A comparative discussion of the deep learning results is provided in Subsection 4.2.2.4.

4.2.2.1 CNN Model Results

Segment Size 25:

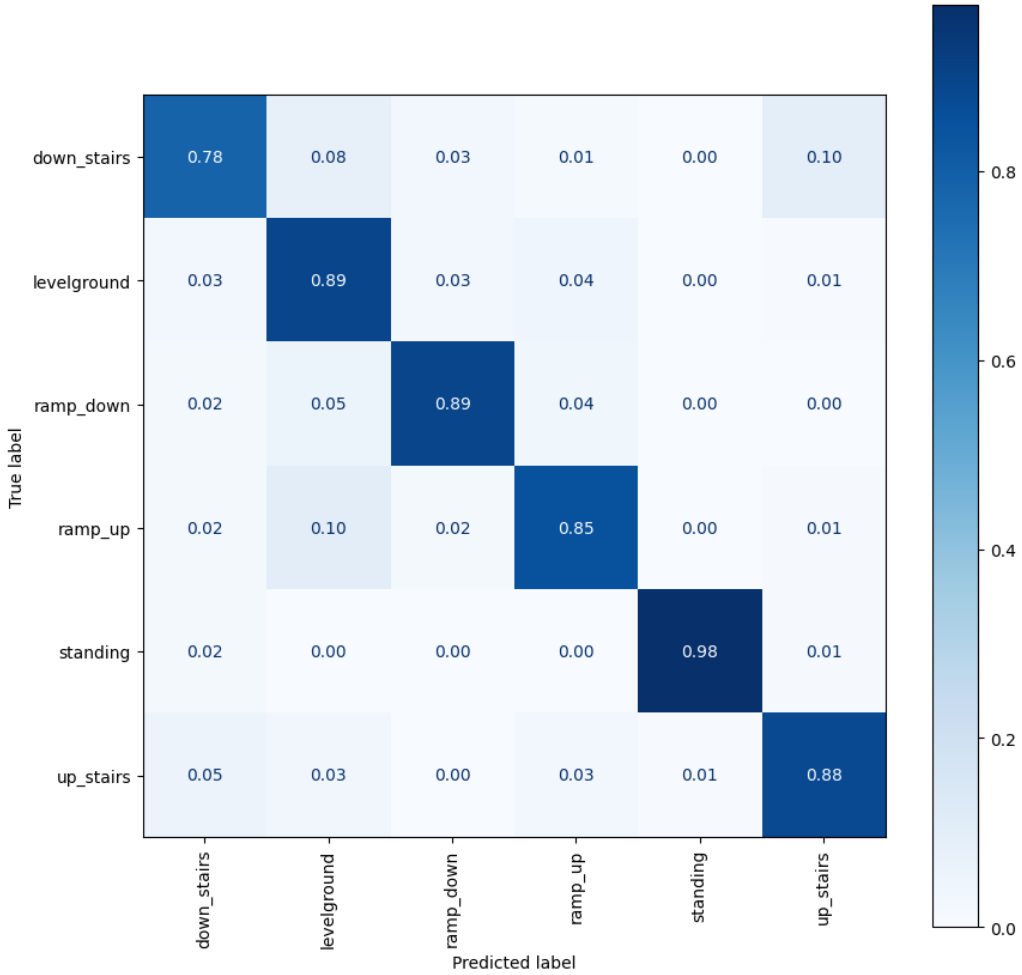


Figure 4.6: Normalized confusion matrix for the CNN model (segment size 25).

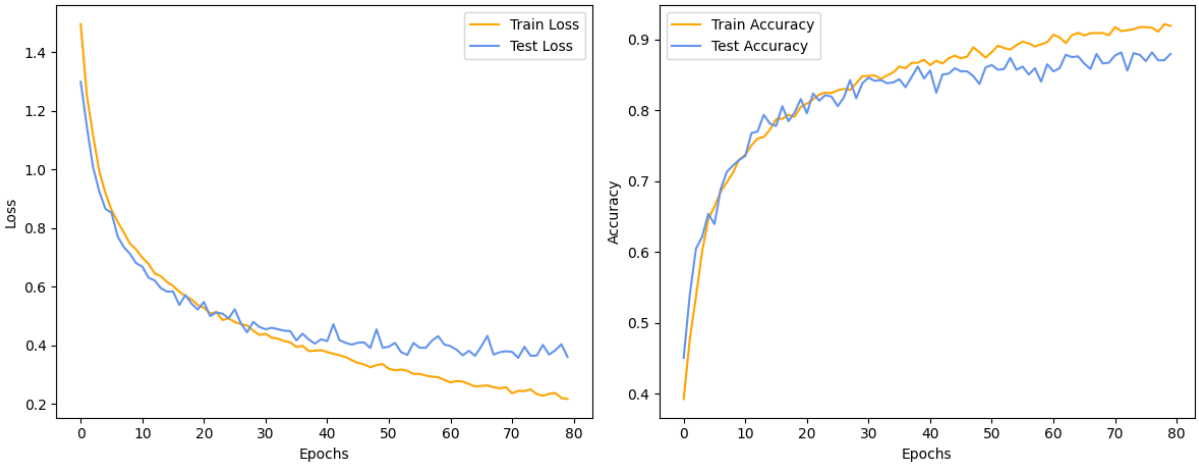


Figure 4.7: Loss and accuracy curves for the CNN model (segment size 25).

Segment Size 50:

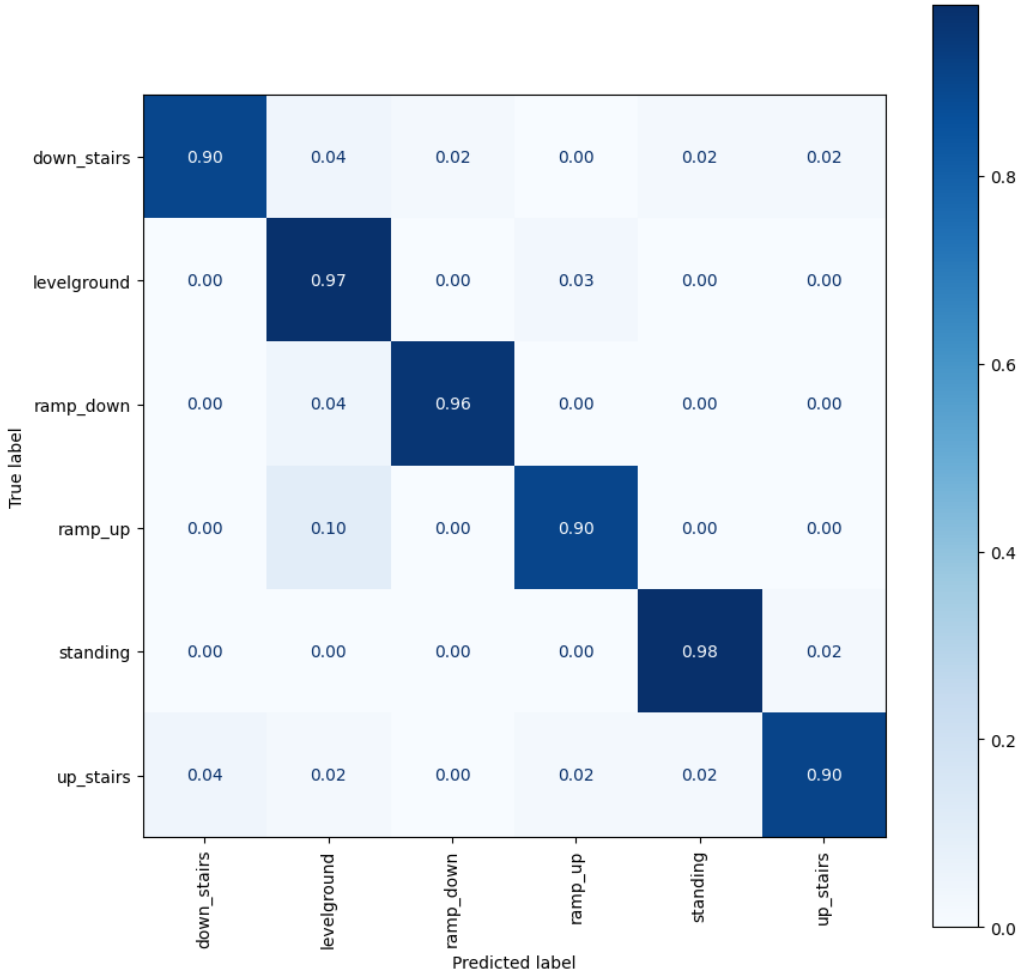


Figure 4.8: Normalized confusion matrix for the CNN model (Segment Size 50).

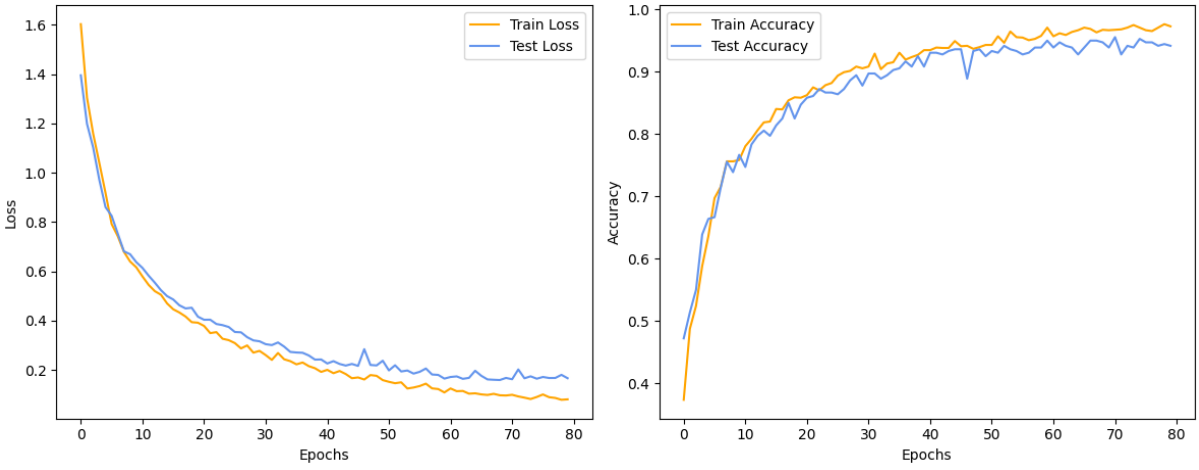


Figure 4.9: Loss and accuracy curves for the CNN model (segment size 50).

Table 4.2: Performance metrics (segment size 25)

Metric	Value
Accuracy	86.95%
Loss	0.3610

Table 4.3: Performance metrics (segment size 50)

Metric	Value
Accuracy	94.44%
Loss	0.1802

As summarized in Tables 4.2 and 4.3, the CNN model achieves an accuracy of 86.95% for a segment size of 25 and 94.44% for a segment size of 50, accompanied by a notable decrease in loss. Overall, the CNN demonstrates robust performance and clearly benefits from larger segment sizes, with the most notable improvement observed in down_stairs classification accuracy.

4.2.2.2 LSTM Model Results

Segment Size 25:

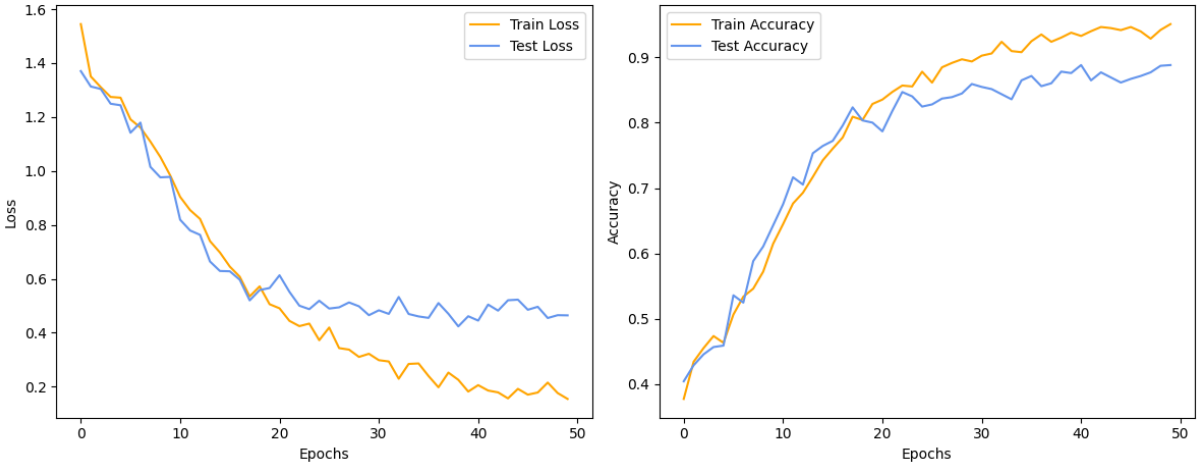


Figure 4.10: Loss and accuracy curves for the LSTM model (segment size 25).

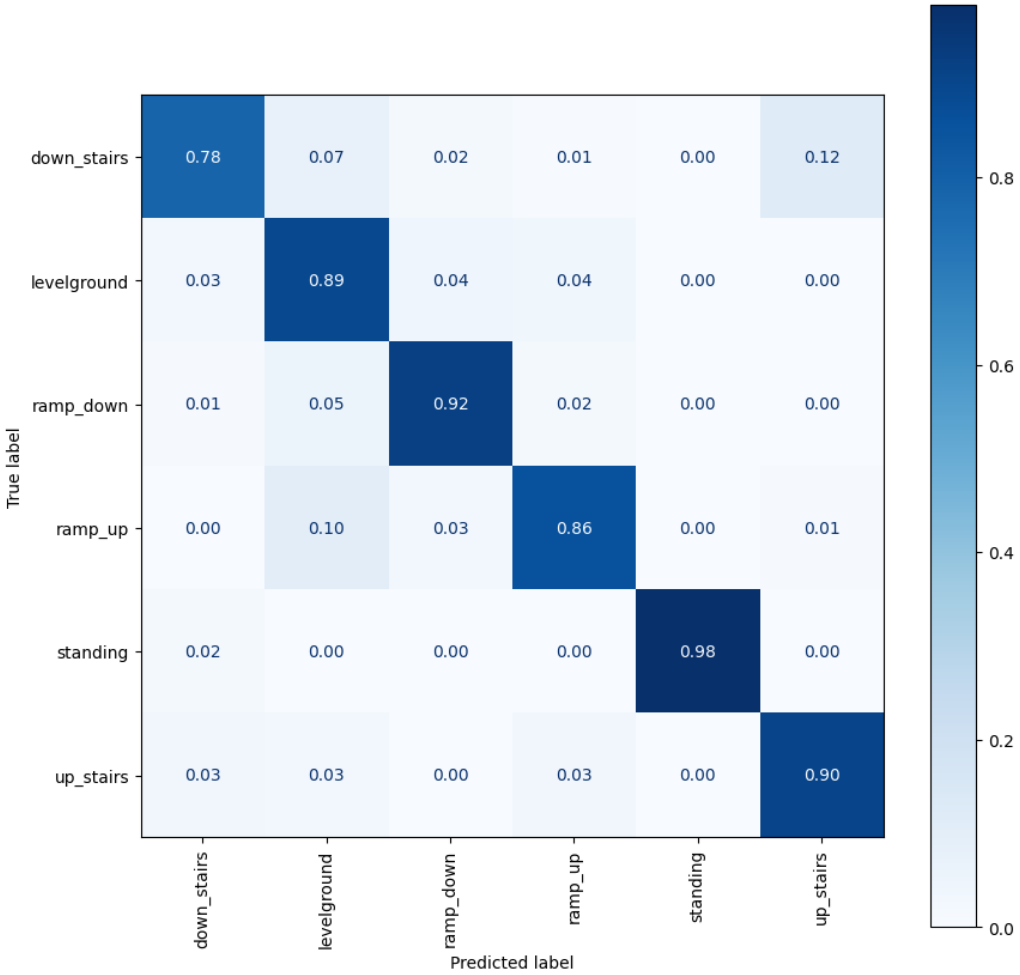


Figure 4.11: Normalized confusion matrix for the LSTM model (segment size 25).

Segment Size 50:

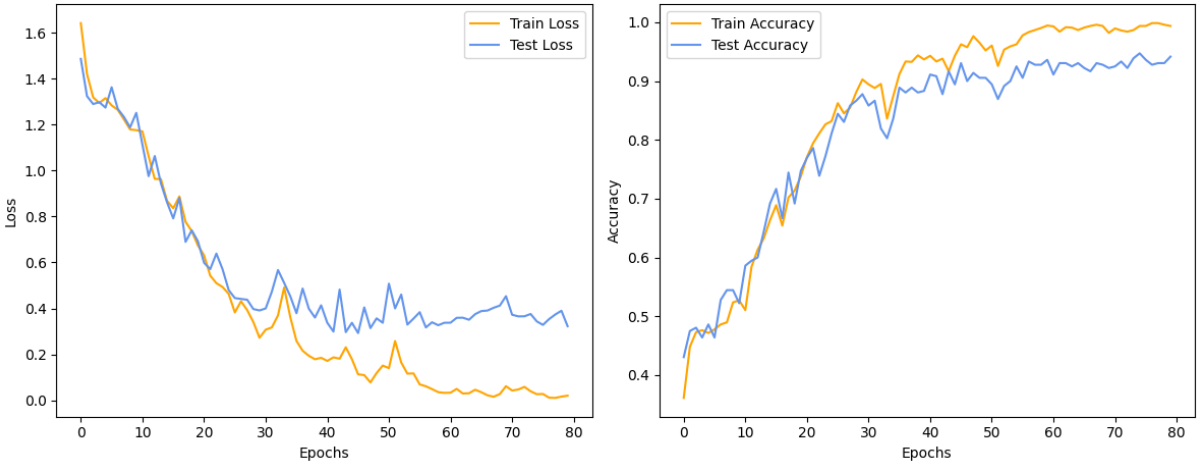


Figure 4.12: Loss and accuracy curves for the LSTM model (segment size 50).

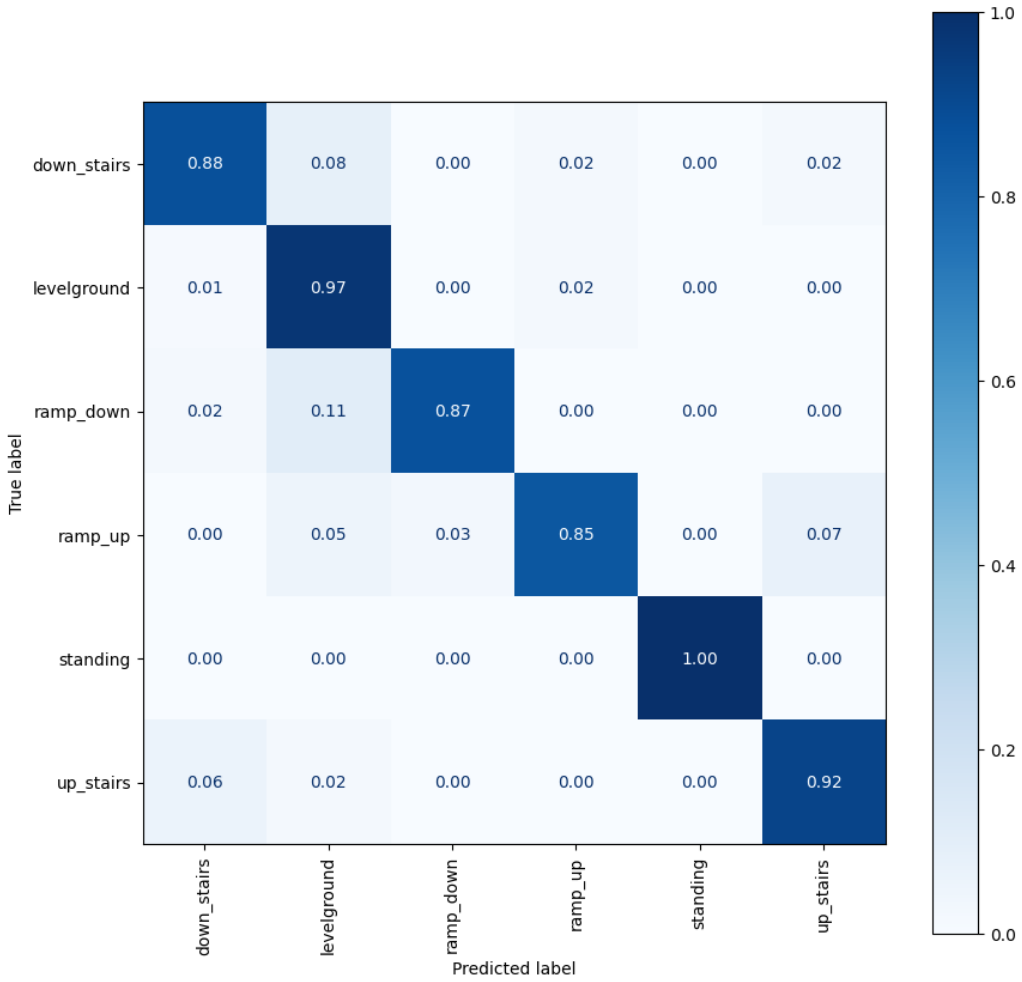


Figure 4.13: Normalized confusion matrix for the LSTM model (segment size 50).

For a direct comparison of the test set performance metrics between the two segment sizes, Tables 4.4 and 4.5 are presented side by side.

Table 4.4: Performance metrics (segment size 25)

Metric	Value
Accuracy	88.84%
Loss	0.4648

Table 4.5: Performance metrics (segment size 50)

Metric	Value
Accuracy	93.06%
Loss	0.3733

Overall, the LSTM model demonstrates robust performance for both segment sizes. To comment that, although the 50 sample configuration yields an improvement in classification accuracy, a few other activities experience slight accuracy drops an effect not observed in the CNN when increasing segment size. This phenomenon is relatively common in LSTM models, when the segment size increases, the broader temporal view can highlight distinctions in certain activities (such as down_stairs) but occasionally introduce minor confusion in others, particularly if they share overlapping motion patterns or rely on shorter, repetitive sequences. In other words, while the larger window can strengthen performance for some classes, it may dilute or complicate the signal for others, resulting in slight drops in accuracy.

4.2.2.3 Hybrid CNN-LSTM Model Results

Based on the performance of the standalone CNN and LSTM models, the hybrid CNN–LSTM architecture was evaluated exclusively with a 50 sample segment size to pursue higher accuracy.

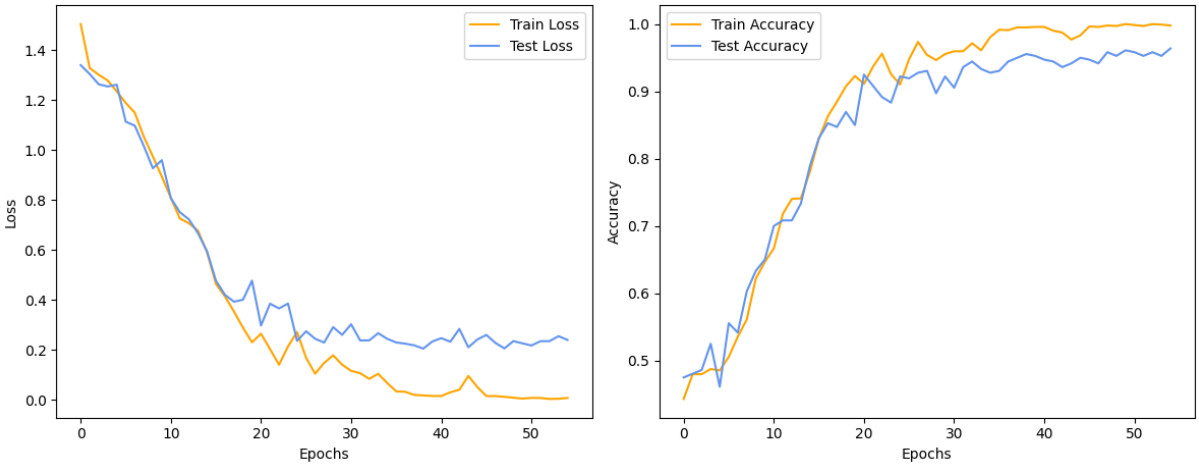


Figure 4.14: Loss and accuracy curves for the Hybrid CNN-LSTM model (segment size 50).

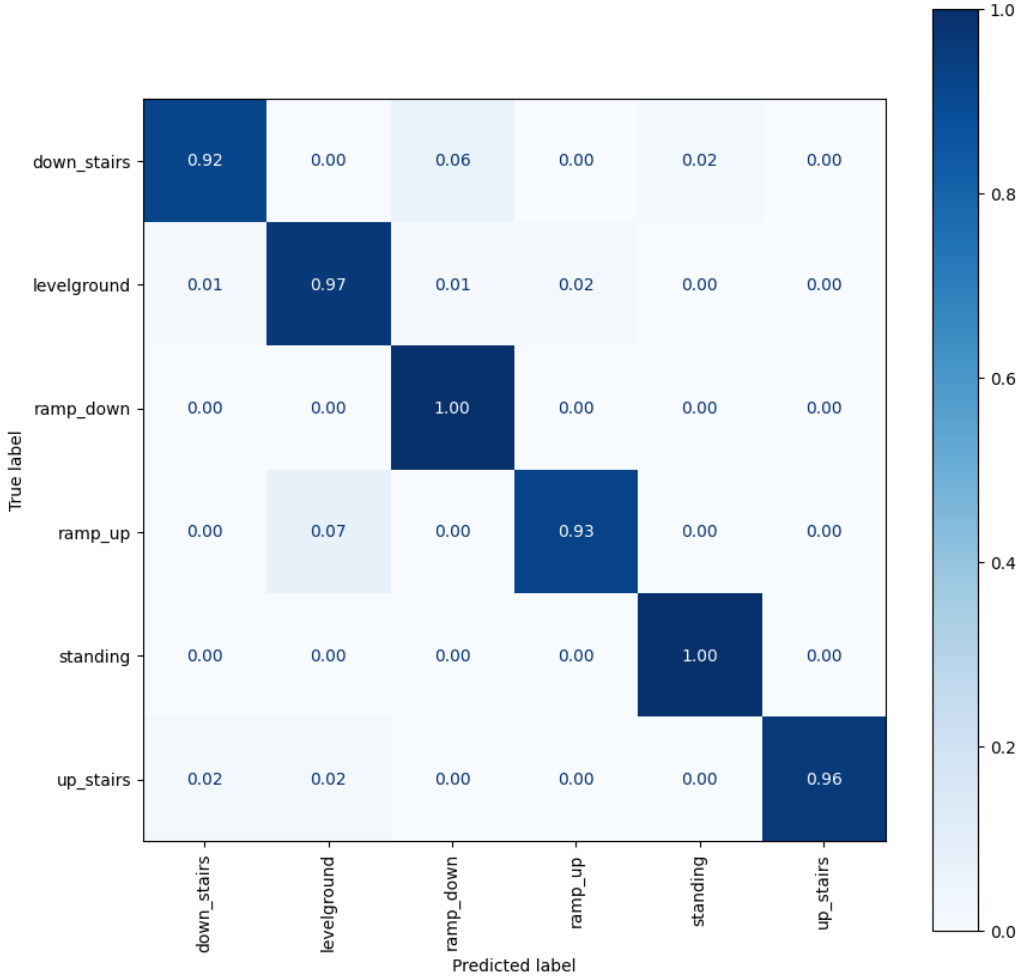


Figure 4.15: Normalized confusion matrix for the Hybrid CNN-LSTM model (segment size 50).

Table 4.6 summarizes the test set performance metrics for the hybrid model:

Table 4.6: Performance metrics (Hybrid CNN-LSTM Model, segment size 50)

Metric	Value
Accuracy	96.39%
Loss	0.2398

Overall, the hybrid CNN–LSTM model demonstrates excellent performance by combining the strengths of both CNN and LSTM architectures, achieving perfect accuracy on ramp_down segments a result not observed in any of the standalone models.

4.2.2.4 Result Comparison and Discussion

Table 4.7 summarizes the final accuracies achieved by the deep learning models under different segment sizes. All proposed methods outperform the baseline RF in both accuracy and stability, indicating the effectiveness of deep architectures in capturing complex features from sensor data.

Table 4.7: Overall Comparison of Deep Learning Models

Model	Segment Size	Accuracy (%)
CNN	25	86.95
CNN	50	94.44
LSTM	25	88.84
LSTM	50	93.06
Hybrid CNN-LSTM	50	96.39

From the loss and accuracy curves (Figures 4.7 to 4.9 for CNN, Figures 4.10 to 4.12 for LSTM, and Figure 4.14 for the Hybrid model), it is evident that increasing the segment size from 25 to 50 improves classification performance. Larger segments provide more contextual information, enabling the networks to learn more robust spatiotemporal patterns. For example, the CNN improves from 86.95% to 94.44% accuracy, while the LSTM goes from 88.84% to 93.06%. To mention once again that for the LSTM model sometimes larger window can both increase and decrease classification accuracy for some individual classes as explained previously.

By integrating strengths from both the CNN and the LSTM, the Hybrid CNN-LSTM achieves the highest overall accuracy of 96.39% taking as input segment sizes of 50 samples.

Examining the curves also shows that both training and validation losses steadily decrease before leveling off, suggesting effective learning with minimal overfitting. The corresponding accuracy curves ascend smoothly, with validation accuracy closely tracking the training accuracy. However, the LSTM model generally exhibits a slightly larger gap between training and validation performance compared to the CNN, indicating that the recurrent structure may be more prone to overfitting. The Hybrid CNN-LSTM model falls somewhere in between, thus maintaining a balanced gap. Overall, these trends suggest that early stopping effectively prevents excessive memorization while allowing each model to learn robust patterns from the data.

The confusion matrices (e.g., Figures 4.6 and 4.8 for CNN, Figures 4.11 and 4.13 for LSTM, and Figure 4.15 for the Hybrid model) confirm that most activities are correctly identified. How-

ever, there is occasional confusion among activities that exhibit similar motion characteristics (such as ramp_up and up_stairs, or down_stairs and ramp_down). Notably, these misclassifications diminish with larger segment sizes, indicating that a broader temporal context helps distinguish subtle movement differences. Comparing the standalone models confusion matrices with that of the hybrid CNN–LSTM confirms that classification accuracy improved for every activity. This demonstrates that the hybrid approach not only achieved higher overall accuracy but also provided stronger individual class performance, making it the preferred choice for further testing.

This way the Hybrid CNN-LSTM confirmes the advantage of combining convolutional and recurrent layers to exploit spatial and temporal information simultaneously.

4.3 Real-Time Testing for a Possible Control

Building on the superior performance of the hybrid CNN-LSTM model described above, this section evaluates the systems real-time processing capabilities. Specifically, the best-performing hybrid model was deployed and tested on two individuals, hereafter referred to as Subject A and Subject B. Subject A's walking data was part of the training set, whereas Subject B (whose height, knee height, and age are similar to those of the training subjects) was entirely new to the system. Testing was conducted in an environment that was not used during the training data collection phase.

To enhance reliability, the proposed possible control signal is updated only after observing three consecutive identical predictions. This approach helps mitigate transient misclassifications due to sensor noise or short transitions between activities. Additionally, a confidence threshold was implemented so that ambiguous cases are classified as "unknown activity", reducing the risk of issuing incorrect control commands. Concretely, the model provides a confidence score for each prediction, and these scores are compared against a set threshold. If no single activity exceeds this threshold, the system defaults to unknown activity, thereby prioritizing safety over potentially inaccurate classifications.

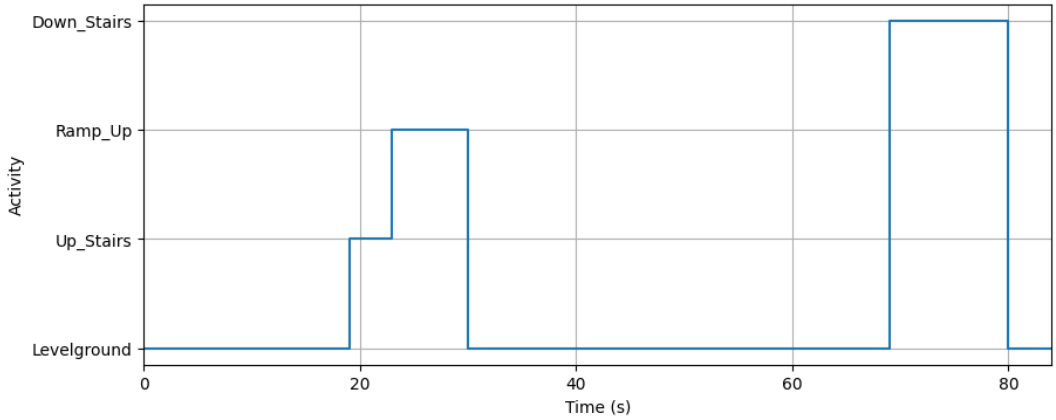
Figures 4.16 and 4.17 present the results for Subject A and Subject B, respectively. Each figure contains two stacked plots. The top plot shows the actual activity being performed, providing a clear baseline for comparison. The bottom plot illustrates the AI model classification outputs (in red) and the resulting possible control signals (in blue). In appendix, figure A.1 includes a QR code linking to a video demonstration of how these model outputs could drive the control system in real-time. It is highly recommended to watch the video for a clearer understanding of the system performance.

During the first 19 seconds of each test, the ground where the trials were conducted had a slight inclination of about 7 degrees. Because the modelramp_down category was trained on slopes ranging roughly from 7 to 12 degrees, this borderline angle acted as a transition zone between levelground (0 degrees) and ramp_down.

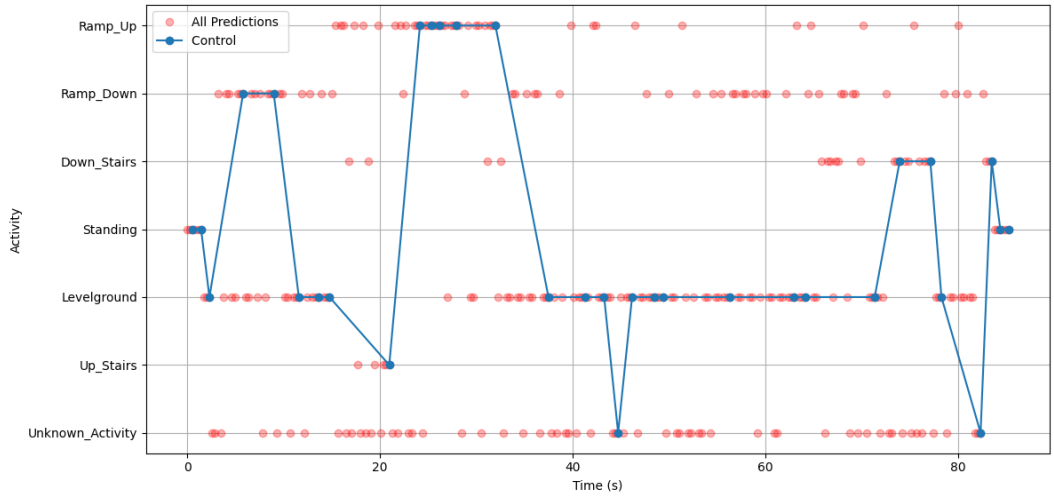
Thus, the overall circuit consisted of an initial levelground/ramp_down section, followed by an up_stairs, ramp_up, a return to levelground, a down_stairs segment, and finally a small levelground stage. Both subjects performed this circuit at slightly different paces, resulting in a small variance of a few seconds in the timing of each activity transition.

Subject A (Figure 4.16) begins with accurate model predictions, correctly identifying both levelground and ramp_down. After this initial section, the subject transitions to upstairs, where

most predictions favor ramp_up. This outcome is understandable given that the stairs are neither particularly tall nor steep, as seen in the shared video. Right after there is a ramp_up section wich is correctly identified. Once the ramp becomes less inclined, the system correctly starts predicting levelground. Although occasional ramp_up or ramp_down guesses occur during this phase, they never appear three times in a row, so the possible control output remains set to levelground. The following segment includes a staircase interspersed with short flat sections. Around the 67 second mark, the model briefly predicts down_stairs, but not consistently enough to trigger a possible control change. The classification then returns to levelground when the subject steps onto a flat area, only to shift back to down_stairs as the subject continues descending. Eventually, another level segment is detected, matching the subject's actual activity. Near the end of the circuit, a series of three consecutive down_stairs predictions appear just before the subject stops (standing), which triggered an incorrect control action. A likely explanation is related to how the subject ended the movement, since the model had previously produced three consecutive unknown_activity guesses right before switching to down_stairs and levelground was the biggest prediction right before.



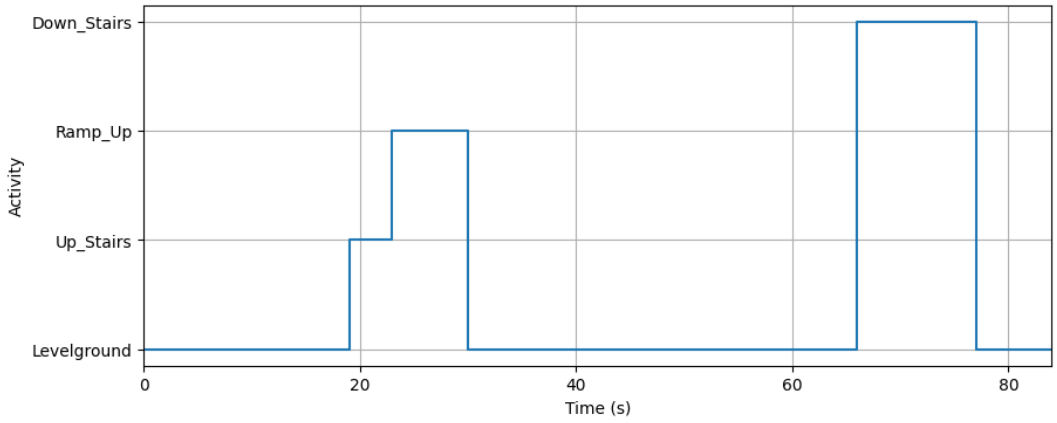
(a) Activity over time for subject A



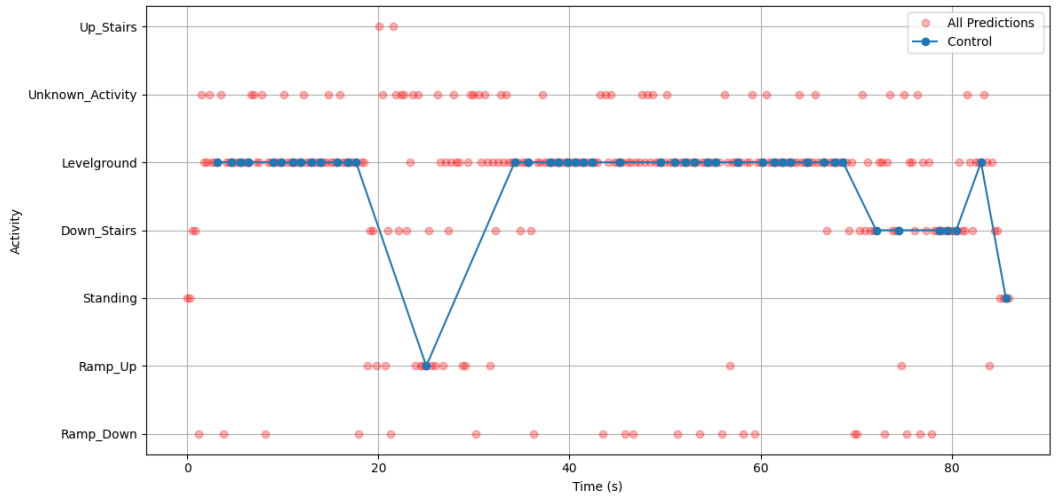
(b) Final control map and predictions for subject A

Figure 4.16: Results for subject A. The top plot shows recognized activities over time, while the bottom plot depicts the possible control decisions based on the model predictions.

For Subject B (Figure 4.17), the initial levelground phase was recognized successfully, whereas the subsequent up_stairs section showed frequent misclassifications with multiple unknown_activity guesses. During ramp_up, the predictions were generally accurate, and the control transitioned to levelground as the incline decreased. The extended levelground period that followed was well identified, and the control remained stable. Once the subject began descending the stairs, the model alternated between down_stairs and levelground predictions, although the control ultimately settled on down_stairs. Finally, a brief levelground segment was correctly recognized, concluding with a standing classification.



(a) Activity Over Time for Subject B



(b) Final Control Map and Predictions for Subject B

Figure 4.17: Results for Subject B. The top plot shows recognized activities over time, while the bottom plot depicts the final control decisions based on the model predictions.

Subject A's predictions were slightly more accurate overall, likely because some of Subject A's walking data was included in the training set, a finding consistent with the potential overfitting indicated by the loss and accuracy curves. Subject B, despite being new to the system, still achieved largely correct classifications. In both scenarios, the model had difficulties with classifying the up_stairs segment of the test, reflecting the fact that the physical staircase used in testing is relatively different from those used for training, thus resembling a ramp more than a typical staircase.

4. RESULTS AND DISCUSSION

Overall, these real-time trials demonstrate that the proposed hybrid CNN-LSTM model adapts well to modest variations in both terrain and user activity, being able to deliver control signals that align closely with the system's intended behavior. While occasional misclassifications occur, the three consecutive predictions rule and confidence threshold help minimize erroneous control outputs. Combined with the system's proven accuracy in offline tests, these results underscore the modelpotential for deployment in assistive leg orthoses and other wearable devices.

Chapter 5

Conclusions, Limitations and Future Work

This chapter summarises the main achievements of the dissertation, revisits the insights gained from the modelling of fluidic muscles and the machine learning framework, and outlines directions for future research and system development.

5.1 Overall Conclusion

This dissertation presented an end-to-end proof of concept for real-time locomotion mode recognition using a single IMU and deep learning models, with the long-term goal of enabling adaptive control in lower-limb orthoses. The work covered three main components: (i) modelling and selecting fluidic muscles capable of producing knee joint moments, (ii) developing a wearable data acquisition system based on a compact, single-sensor IMU configuration, and (iii) designing and evaluating machine learning models capable of classifying six locomotion modes with high accuracy.

A custom dataset was collected from multiple subjects across realistic walking environments. Conventional machine learning methods served as a baseline, but deep learning models significantly outperformed them. In particular, the hybrid CNN-LSTM architecture achieved a final accuracy of 96.39% using short 50-sample windows, confirming that high-quality activity recognition can be achieved with minimal sensing hardware. Real-time experiments further demonstrated that the model generalises well to unseen subjects, producing stable predictions suitable for a potential closed-loop control pipeline.

The fluidic muscle modelling results indicated that suitably chosen pneumatic actuators can theoretically produce knee joint moments comparable to those required during walking, stair ascent, and ramp locomotion. Although based on linear approximations, these simulations support the feasibility of fluidic muscles as an alternative actuation mechanism to electric motors for lightweight, compliant wearable orthoses.

Together, these findings establish a solid foundation for the development of compact, accurate, and adaptive orthotic systems that combine minimal sensing with intelligent control strategies.

5.2 Reflections on Fluidic Muscle Modelling

Fluidic muscle behaviour was approximated using linear regression, enabling integration into a simplified knee joint moment model. This approach provided initial insights into optimal actuator lengths, diameters, and operating pressures. Although this linear approximation neglects the nonlinearities inherent to pneumatic systems, it offered a practical starting point for assessing feasibility.

Further validation through physical prototyping is essential. Real contraction curves, dynamic force responses, and pressure–time characteristics may differ substantially from the simplified model. Nonetheless, the simulations performed in this dissertation demonstrate that fluidic muscles have strong potential as a compliant and lightweight actuation method for orthotic devices.

5.3 Limitations and Opportunities for Model Improvement

While the achieved results are highly promising, several limitations should be acknowledged. First, certain locomotion classes with subtle biomechanical differences (e.g., stair ascent vs. ramp ascent) remain challenging to distinguish, reflecting both natural gait variability and dataset size.

Different users display unique gait patterns, meaning that a single global model may not yield optimal predictions for all individuals. Personalised models or subject-specific fine tuning could substantially improve performance. Moreover, although the hybrid CNN-LSTM model yielded excellent accuracy, emerging sequence modelling architectures, such as Transformers or attention-based networks may further enhance robustness and interpretability.

Finally, published accuracy values in related work must be interpreted cautiously. Differences in dataset preparation, segmentation windows, and experimental design make comparisons non-trivial. Nonetheless, this dissertation demonstrates that meaningful and reliable activity recognition is achievable with minimal sensor input when models are carefully designed and validated.

5.4 Future Work

Building on the outcomes of this dissertation, several research directions are recommended:

1. **Expanded Dataset Collection:** Acquire larger and more diverse datasets, including additional subjects, speeds, and environmental conditions, to improve generalisation and robustness.

2. **Advanced AI Architectures:** Explore next-generation models such as Transformers or attention-based networks, which may better capture long-range temporal dependencies and subtle gait variations.

3. **Long-Term Device Testing:** Deploy the trained model on passive orthotic devices for extended real-world testing to study drift, noise accumulation, and user adaptation.

4. **Fluidic Muscle Prototyping:** Construct a physical prototype integrating the selected fluidic muscles to validate simulated contraction forces and evaluate real-time actuation performance.

5. **Closed-Loop Control Integration:** Develop a real-time feedback algorithm that adjusts fluidic muscle pressure based on predicted user activity, enabling true adaptive assistance.

These research directions aim to transform the presented proof-of-concept into a practical, reliable, and user-friendly assistive technology for everyday use.

Appendix A

Supplementary Materials

A.1 Aditional Images

Figure A.1: QR code linking to the demonstration video showcasing the possible real-time control during movement.

Bibliography

Anguita, Davide et al. (2013). "A Public Domain Dataset for Human Activity Recognition Using Smartphones". In: *21st International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Springer, pp. 437–442.

Ariza-Colpas, Enrique et al. (2022). "Hybrid CNN-LSTM models for robust human activity recognition". In: *Pattern Recognition Letters* 157, pp. 20–27.

Bevilacqua, Antonio et al. (2019). "Human activity recognition with convolutional neural networks". In: *arXiv preprint arXiv:1906.01935*. URL: https://arxiv.org/abs/1906.01935.

Breiman, Leo (2001). "Random forests". In: *Machine Learning* 45.1, pp. 5–32. DOI: 10.1023/A:1010933404324. URL: https://link.springer.com/article/10.1023/A%3A1010933404324.

Ghojogh, Benyamin and Ali Ghodsi (2023). *Recurrent Neural Networks and Long Short-Term Memory Networks: Tutorial and Survey*. arXiv preprint arXiv:2304.11461. A comprehensive tutorial-style survey on RNNs and LSTMs. URL: https://arxiv.org/abs/2304.11461.

Ha, Seungjin and Jang-Ho Choi (2016). "Convolutional neural networks for human activity recognition using mobile sensors". In: *Sensors* 16.11, p. 1861.

Huang, He et al. (2011). "Continuous locomotion-mode identification for prosthetic legs based on neuromuscular-mechanical fusion". In: *IEEE Transactions on Biomedical Engineering* 58.10, pp. 2867–2875. DOI: 10.1109/TBME.2011.2161671.

Ohta, Masayoshi et al. (2018). "Pneumatic Artificial Muscles for Wearable Exoskeletons: Design and Control". In: *Advanced Robotics* 32.10, pp. 506–516.

Ordóñez, Francisco Javier and Daniel Roggen (2016). "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition". In: *Sensors (Basel)*. Vol. 16. 1. MDPI, p. 115.

Peppas, Konstantinos et al. (2020). "Real-time human activity recognition using a lightweight convolutional neural network on mobile devices". In: *IEEE Sensors Journal* 20.16, pp. 9091–9099.

Shakerian, Zahra et al. (2023). "Real-time human activity recognition on embedded devices using a quantized convolutional neural network". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 7.2, pp. 183–191.

Vu, Huong Thi Thu et al. (2022). "Comparison of machine learning and deep learning-based methods for locomotion mode recognition using a single inertial measurement unit". In: *Frontiers in Neurorobotics* 16, p. 923164. DOI: 10.3389/fnbot.2022.923164.

Vu, Nam et al. (2022). "IMU-based Activity Recognition for Wearable Assistive Devices: A Comparative Study". In: *IEEE Sensors Journal* 22.7, pp. 3001–3009.

Wang, Li et al. (2018). "LSTM based activity recognition for lower limb exoskeleton control". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26.4, pp. 791–800.

Weiss, Gerhard, Andreas Kirchgessner, and Gerhard Troster (2011). "Comparative study of different classification algorithms for human activity recognition using smartphone accelerometer data". In: *IEEE Sensors Journal* 11.1, pp. 82–90.

Xu, Ling et al. (2019). "InnoHAR: Inception based hybrid network for human activity recognition". In: *2019 International Conference on Body Sensor Networks (BSN)*. IEEE, pp. 1–6.