# Advancing Human Activity Recognition: Innovations in Signal Representations and Machine Learning

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Christoph Wieland

---

---

Tag der mündlichen Prüfung: 20.10.2025

1. Referent: PD Dr. Victor Pankratius
2. Referent: Prof. Dr. Michael Beigl

This document is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0): https://creativecommons.org/licenses/by-nc-nd/4.0/deed.en

# Abstract

The Wearable Human Activity Recognition (WHAR) research field focuses on the automated analysis of human motion and behavior in signals from body-worn sensors like accelerometers or gyroscopes. With modern wearable devices, ranging from smartwatches and fitness trackers to earbuds, it now becomes possible to collect such signals continuously in a variety of everyday situations. Users typically want to monitor and get insights on how to improve their daily activities, performance in sports, or health-related metrics.

Machine Learning (ML) models are commonly used to model and classify WHAR data, as they excel in adapting to the many types of individuals and their variations in performing activities. Recently, neural networks have become prominent because they offer better generalizability and adaptability than other approaches. However, deploying such models on mobile devices poses significant challenges. Limited computing power, short battery life, and high variability in activity data caused by different users, sensor placements, and movement patterns can all hinder model performance. As a result, a key objective of WHAR is to develop robust and efficient models that are capable of not only recognizing basic actions like walking or sitting but also complex motion patterns such as compound exercises.

Developing ML for WHAR comes with a major tradeoff: Either one can focus on making models as small as possible at the expense of generalizability in application, or vice versa. This dissertation explores methods at both extremes, building on flexible graph representations with Graph Neural Networks (GNNs) when it comes to small, specialized applications, and on Foundation Model (FM) techniques for scenarios that require greater generalization across diverse sensing setups. Its key contributions include (i) resource-efficient and reusable graph

transformations that capture temporal dependencies in WHAR signals and complex activity sequences, (ii) GNNs that facilitate lightweight activity recognition and sequential activity prediction, (iii) a framework for harmonizing heterogeneous signals from various sensors that allows general-purpose WHAR models to learn from large-scale data, and (iv) FMs that enable precise signal forecasting and classification through novel dual-view and multi-resolution signal analysis.

In its first part, this work presents techniques for converting signal and activity representations into graph representations, which are necessary for the classification and reasoning of interrelated activity patterns at different levels of abstraction using GNNs. The first approach matches the classification performance of state-of-the-art (SOTA) classifiers on four benchmark datasets while requiring up to three orders of magnitude fewer parameters. Short inference times on a smartwatch demonstrate the model's ability to operate close to real time. The second approach proves effective for sequential activity recognition on two benchmark datasets, underscoring the versatility of GNNs within the WHAR domain.

The second part presents two FMs that extract versatile signal representations from eight aligned datasets. These models help address the diversity of sensor signals and activities. The first WHAR-FM outperforms SOTA in signal forecasting by up to $12\,\%$ in mean absolute error. Results also show that its generalizability improves steadily with the amount of training data. The second WHAR-FM integrates with neural network-based classifiers, enhancing their F1 scores by up to 0.286 points while only slightly increasing inference times on a smartwatch.

In sum, this dissertation demonstrates how innovative signal representations and ML models can improve real-world WHAR applications. By representing signals as graphs, GNNs capture complex relationships within the data, while WHAR-FMs learn generalizable signal representations from large datasets. Extensive experiments show that GNNs and FMs help overcome key challenges such as device limitations, signal variability, and the broad range of users and activities. Together, these findings support the development of more efficient and accurate WHAR solutions and provide researchers and engineers with a new toolbox of methods that they can employ to handle a variety of use cases.

# Kurzfassung

Das Forschungsgebiet der Wearable Human Activity Recognition (WHAR) befasst sich mit der automatisierten Analyse menschlicher Bewegungen anhand von Sensorsignalen wie Beschleunigungs- oder Gyroskopdaten. Mit modernen tragbaren Geräten, die von Smartwatches und Fitness-Trackern bis hin zu Kopfhörern reichen, ist es nun möglich, solche Signale kontinuierlich in einer Vielzahl von Alltagssituationen zu erfassen. Nutzer möchten typischerweise ihre täglichen Aktivitäten, sportlichen Leistungen oder gesundheitsbezogenen Metriken überwachen und Erkenntnisse darüber gewinnen, wie sie diese verbessern können.

Zur Modellierung und Klassifikation von WHAR-Daten kommen überwiegend Methoden des Maschinellen Lernens (ML) zum Einsatz, da sie sich hervorragend an verschiedene Nutzertypen und Aktivitäten adaptieren lassen. In jüngerer Zeit werden verstärkt neuronale Netze verwendet, da sie anderen Verfahren in Generalisierungs- und Anpassungsfähigkeit überlegen sind. Der Einsatz solcher Modelle auf mobilen Endgeräten ist jedoch herausfordernd. Begrenzte Rechenleistung, kurze Akkulaufzeiten sowie die große Variabilität der Aktivitätsdaten, bedingt durch unterschiedliche Nutzer, Sensorplatzierungen und Bewegungsmuster, können die Modellleistung einschränken. Eine zentrale Aufgabe der WHAR ist daher die Entwicklung robuster und effizienter Modelle, die nicht nur elementare Aktivitäten wie Gehen oder Sitzen, sondern auch komplexe Handlungsabfolgen wie zusammengesetzte Sportübungen zuverlässig erkennen können.

Die Entwicklung solcher ML-Modelle erfordert einen Kompromiss zwischen Modellgröße und Generalisierungsfähigkeit. Für kleine, spezialisierte Anwendungen setzt diese Arbeit daher auf flexible Graph Neural Networks (GNNs) und für Szenarien, die eine bessere Generalisierung über verschiedene Sensormodalitäten

hinweg erfordern, auf Foundation Model (FM)-Techniken. Ihre Kernbeiträge sind (i) effiziente und wiederverwendbare Graphtransformationen, die zeitliche Abläufe in Sensorsignalen und komplexen Aktivitätssequenzen abbilden, (ii) GNNs zur leichtgewichtigen Aktivitätserkennung und Vorhersage komplexer Aktivitätssequenzen, (iii) ein Konzept zur Harmonisierung heterogener Sensorsignale für das Training universeller WHAR-Modelle, sowie (iv) FMs, die präzise Signalvorhersagen und -klassifikationen durch neuartige Signalanalysen ermöglichen.

Der erste Teil dieser Arbeit präsentiert Graphtransformationen, die für die Erkennung von Aktivitäten auf verschiedenen Abstraktionsebenen mittels GNNs notwendig sind. Der erste Ansatz erreicht auf vier Referenzdatensätzen ähnliche Erkennungsraten wie topaktuelle Klassifikatoren, benötigt dafür jedoch bis zu drei Größenordnungen weniger Parameter. Kurze Inferenzzeiten auf einer Smartwatch demonstrieren die Echtzeitfähigkeit des Modells. Der zweite Ansatz zeigt die Wirksamkeit von GNNs für die sequenzielle Aktivitätserkennung auf zwei Datensätzen und unterstreicht damit deren Vielseitigkeit im WHAR-Bereich.

Der zweite Teil stellt zwei FMs vor, die vielfältig einsetzbare Signaldarstellungen aus acht vereinheitlichten Datensätzen extrahieren und so die Diversität von Sensorsignalen und Aktivitäten kompensieren. Das erste WHAR-FM erzielt eine um bis zu $12\,\%$ geringere absolute Abweichung bei der Signalvorhersage als aktuelle Referenzmodelle. Die Evaluation zeigt zudem eine mit wachsender Datenmenge stetig steigende Generalisierungsfähigkeit. Das zweite WHAR-FM lässt sich mit nachgeschalteten Klassifikatoren kombinieren, was deren F1-Maße um bis zu 0,286 Punkte verbessert, während sich die Inferenzzeiten nur geringfügig erhöhen.

Diese Dissertation demonstriert somit das Potenzial innovativer Signaldarstellungen und ML-Modelle zur Verbesserung realer WHAR-Anwendungen. Umfangreiche Experimente zeigen, dass GNNs und FMs zentrale Herausforderungen wie Gerätebeschränkungen, Signalvariabilität und Aktivitätsvielfalt erfolgreich adressieren und so den Weg für effizientere und präzisere Anwendungen ebnen. Die vorgestellten Methoden bilden einen Werkzeugkasten, den Forscher und Ingenieure zur Bewältigung einer Vielzahl von Anwendungsfällen einsetzen können.

# Acknowledgements

Completing this dissertation marks the end of a challenging but rewarding journey. Along the way, I had the privilege of learning from and working with many inspiring individuals. I am deeply grateful for their support, encouragement, and belief in me throughout this time.

First and foremost, I would like to thank my supervisor, Dr. Victor Pankratius, for giving me the opportunity to conduct my research at Bosch Sensortec. His trust and continuous support, his insightful ideas, and the substantial time he devoted to proofreading my written work were invaluable to the success of this dissertation.

I further thank Prof. Dr. Michael Beigl for kindly agreeing to serve as the second reviewer of this dissertation.

I send many thanks to my colleagues at Bosch Sensortec, especially Dr. David John, Dr. Lukas Grinewitschus, and Dr. Dirk Staneker for our enlightening discussions, as well as Arjun Haritsa Krishnamurthy and Timo Giesselmann for their assistance with training hardware.

Special thanks go to my fellow PhD colleagues Alexandra Küster, Maximilian Rutz, Wolfram Mayer, Ferdinand Auerswald, and Tobias King for the many motivating conversations and shared experiences.

Finally, I am deeply thankful to my family and my girlfriend for their unwavering support and encouragement throughout this entire endeavor. Their presence made the most difficult moments manageable and successes even more enjoyable.

# Contents


**I The Landscape of Human Activity Recognition 1**

**1 Introduction . . . 3**
1.1 Motivation . . . 3
1.2 Aim and Objectives . . . 5
1.3 Research Questions . . . 7
1.4 Contributions . . . 8
1.5 Thesis Structure . . . 9

**2 Basics of Wearable Human Activity Recognition . . . 13**
2.1 Signals, Users, and Activities . . . 13
2.2 Signal Pre-Processing with Wavelets . . . 16
2.2.1 Discrete Wavelet Transform . . . 16
2.2.2 Related Work . . . 18
2.3 Activity Classification . . . 19
2.3.1 Task and Goal . . . 19
2.3.2 Evaluation Metrics . . . 20
2.3.3 Related Work . . . 21
2.4 Signal Forecasting . . . 22
2.4.1 Task and Goal . . . 22
2.4.2 Evaluation Metrics . . . 23
2.4.3 Related Work . . . 24
2.5 Public Benchmark Datasets . . . 25
2.6 Key Challenges . . . 26
2.7 Summary . . . 29

**3 Basics of Graph Neural Networks for Human Activity Recognition . . . 31**
3.1 Graphs . . . 31
3.2 Graph Neural Networks . . . 33
3.3 Applications in Wearable Human Activity Recognition . . . 37
3.3.1 Graph-Level Recognition . . . 41
3.3.2 Vertex-Level Recognition . . . 44
3.3.3 Edge-Level Recognition . . . 44
3.4 Summary . . . 45

**4 Basics of Foundation Models for Human Activity Recognition 47**
4.1 Foundation Models . . . 48
4.2 Self-Supervised Pre-Training . . . 49
4.3 Task-Specific Fine-Tuning . . . 53
4.4 Summary . . . 54

**II Innovative Signal Representations with Graph Neural Networks 55**

**5 Lightweight Activity Classification on the Graph-Level . . . 57**
5.1 Signal-Graph Transformation . . . 57
5.2 Model Design . . . 60
5.3 Implementation and Evaluation Details . . . 62
5.3.1 Hyperparameters . . . 62
5.3.2 Data Preparation . . . 63
5.3.3 Training Setup . . . 66
5.4 Performance Analysis . . . 66
5.4.1 Comparison of Model and Graph Configurations . . . 67
5.4.2 Comparison with State-of-the-Art Classifiers . . . 73
5.4.3 Efficiency Analysis . . . 76
5.5 Summary . . . 77

**6 Sequential Activity Prediction on the Edge-Level . . . 79**
6.1 Activity-Graph Transformation . . . 79
6.2 Model Design . . . 82

6.3 Implementation and Evaluation Details . . . 84
6.3.1 Hyperparameters . . . 84
6.3.2 Data Preparation . . . 85
6.3.3 Training Setup . . . 90
6.4 Performance Analysis . . . 91
6.5 Summary . . . 95

**III Advanced Signal Generalization with Foundation Models 97**

**7 Preparation Steps for Effective Pre-Training . . . 99**
7.1 Harmonizing Heterogeneous Signals . . . 99
7.1.1 Dataset Selection . . . 100
7.1.2 Signal Alignment . . . 102
7.2 Location-Invariant Pre-Training . . . 105
7.3 Summary . . . 107

**8 Signal Forecasting with Foundation Models . . . 109**
8.1 Model Design . . . 109
8.2 Location-Invariant Pre-Training . . . 114
8.3 Location-Invariant Fine-Tuning . . . 114
8.4 Implementation and Evaluation Details . . . 115
8.4.1 Relevance of Location Embeddings . . . 116
8.4.2 Hyperparameters . . . 116
8.4.3 Training Setup . . . 117
8.5 Performance Analysis . . . 119
8.5.1 Pre-Training . . . 119
8.5.1.1 Causal Signal Modeling . . . 120
8.5.1.2 Masked Signal Modeling . . . 122
8.5.1.3 Dataset Size . . . 124
8.5.2 Fine-Tuning . . . 126
8.5.2.1 Comparison with State-of-the-Art Forecasters . . . 126
8.5.2.2 Activity-Level Performance . . . 127
8.6 Summary . . . 129

**9 Activity Classification with Foundation Models . . . . . 131**
9.1 Model Design . . . . . 132
9.2 Location-Invariant Pre-Training . . . . . 139
9.3 Multi-Location Fine-Tuning . . . . . 140
9.4 Implementation and Evaluation Details . . . . . 141
9.4.1 Hyperparameters . . . . . 141
9.4.2 Training Setup . . . . . 142
9.5 Performance Analysis . . . . . 145
9.5.1 Comparison with State-of-the-Art Classifiers . . . . . 146
9.5.2 Activity-Level Performance . . . . . 151
9.5.3 Efficiency Analysis . . . . . 153
9.5.4 Comparison with other Wavelet-Integrated Models . . . . . 155
9.6 Summary . . . . . 157

**IV Conclusion and Future Directions 159**

**10 Conclusion . . . . . 161**

**A Appendix . . . . . 165**
A.1 Signal Forecasting with Foundation Models – Comparison Models 165
A.2 Activity Classification with Foundation Models – Model Efficiency 167

**List of Figures . . . . . 171**

**List of Tables . . . . . 175**

**List of Relevant Publications . . . . . 177**

**Bibliography . . . . . 179**

# Part I

# The Landscape of Human Activity Recognition

# 1 Introduction

Wearable Human Activity Recognition (WHAR) focuses on the detection and analysis of human motion and behavior through sensor signals obtained from wearable devices, such as smartwatches and earables [56, 89]. These wearables utilize inertial sensors, including accelerometers and gyroscopes, to capture motion data from various body parts but suffer from limited computational resources, which restrict the complexity of deployable WHAR applications [56]. Nevertheless, the inherent flexibility of body-worn sensors, which typically exhibit only a few degrees of freedom [19], facilitates the collection and analysis of personalized data [89]. This capability enables real-time and ubiquitous monitoring of human behavior [89], thereby enhancing the understanding of physical activity patterns and presenting new opportunities for supporting users in their everyday lives.

## 1.1 Motivation

Due to the increasing prevalence of smartphones [21] and wearables [74], WHAR has the potential to enrich the lives of billions of users. In sports, for example, WHAR applications can detect and assess the quality of movements [158], enabling users to improve their techniques based on learned insights. Beyond sports, WHAR is also used for detecting falls [167], daily activities [92], and more [75]. Detectable activities range from complex high-level activities (e.g., compound sports exercises or preparing meals) [9], over basic low-level activities (e.g., walking or running) [117], to individual limb movements (e.g., arm or leg movements) [101].

Machine Learning (ML) plays a pivotal role in the analysis of these diverse activities, providing flexible methodologies for the automatic processing of vast amounts of data. The adaptability of ML algorithms is particularly important for WHAR, where sensor signals can vary significantly between users, activities, and the sensor placements on the human body [7, 89]. Earlier methods rely on Decision Trees (DT) [14] or Support Vector Machines (SVM) [66]. In contrast, contemporary approaches leverage or combine different neural network architectures, including Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), with recent advancements exploring Transformer models and attention mechanisms [17, 41, 177]. Training sophisticated ML models requires large amounts of data. However, due to the costly and time-consuming acquisition of human inertial signals, current WHAR datasets are often restricted to data from a few users and activities [10, 87] and consequently, limit the generalizability of trained models.

Three key challenges emerge from the previous paragraphs (cf. [7]): (i) the diversity of signals from different users, sensors, and sensor placements, (ii) the computational limitations of wearable devices, and (iii) the complexity of activities. However, addressing all three challenges simultaneously presents significant difficulties. The detection of complex activities with resource-efficient ML models, for example, restricts the signal diversity to small scopes. Thus, Figure 1.1 illustrates the challenges as a tradeoff triangle, where improving any two aspects typically comes at the expense of the third challenge.

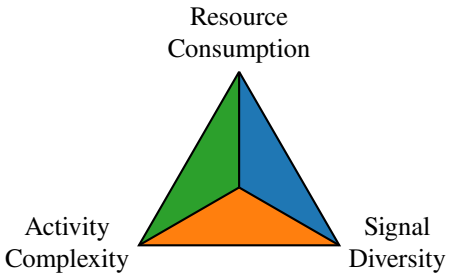



**Figure 1.1:** Tradeoff triangle of key challenges in WHAR, where improving two aspects typically comes at the cost of the third.

To address different combinations of tradeoffs in the shown triangle, this dissertation explores the integration of advanced neural networks into the WHAR landscape, specifically Graph Neural Networks (GNNs) and Foundation Models (FMs).

GNNs are a unique class of neural networks with typically shallow architectures that analyze vertex features and edge relationships in input graphs [150, 175]. Due to their shallow depth and the flexible structure of their input graphs, GNNs support the efficient recognition of both basic and complex activities. However, since raw sensor signals are not inherently structured as graphs, the practical integration of GNNs into the WHAR domain necessitates the careful design of sophisticated signal-to-graph transformations. This limits GNNs to small-scale applications with a well-defined scope, as the design of effective graph transformations for sensor data presents a substantial effort.

FMs, on the other hand, are pre-trained neural networks capable of deriving generalized representations from large amounts of unlabeled data [18]. This generalization ability positions FMs as a promising framework for improving WHAR by uncovering important signal properties while reducing costly labeling efforts. However, FMs typically comprise millions, billions, or even trillions of parameters [18]. Consequently, the integration of FMs into the WHAR domain requires key research steps, including the homogenization of signals from different sources to support streamlined learning on large data volumes and the development of FM architectures that can operate within the computational limitations of small-scale wearable devices.

## 1.2 Aim and Objectives

This dissertation aims to advance WHAR by exploring novel signal representations and neural network architectures that enable a comprehensive, versatile, and resource-efficient analysis of human activities in diverse real-world scenarios. Specifically, this work pursues the following objectives:

(O1) Explore advanced signal and activity representations that efficiently capture intricate patterns in WHAR signals.

(O2) Harmonize wearable sensor signals from various sources using comprehensive signal generalization techniques.

(O3) Develop and implement innovative neural networks that integrate objectives O1 and O2 to enhance recognition accuracy and robustness.

(O4) Empirically validate the performance and wearable applicability of the developed signal representations and models through extensive experiments.

Figure 1.2 maps the objectives onto the tradeoff triangle for WHAR. O1 aims at balancing resource consumption with activity complexity and signal diversity by enhancing input features. O2 focuses on ensuring consistent data integration across diverse sensing setups, addressing the challenge of signal diversity. O3 contributes ML models capable of recognizing both low-level activities from various sensing setups and complex high-level activities while maintaining low resource consumption. O4 validates different aspects of the developed models, positioning it at the center of the triangle.

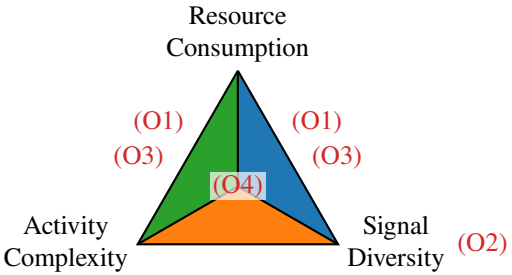



**Figure 1.2:** Tradeoff triangle of key challenges in WHAR labeled with this work's objectives.

## 1.3 Research Questions

Achieving accurate, efficient, and generalizable WHAR remains challenging due to the tradeoffs between signal diversity, activity complexity, and the computational constraints of wearable devices. This dissertation addresses these tradeoffs by employing GNNs to model structured relationships in sensor signals and FMs to improve generalization and scalability, ultimately enhancing the versatility of WHAR. However, the integration of these advanced techniques raises several research questions that are closely tied to this work's objectives. Figure 1.3 maps these questions onto the tradeoff triangle.

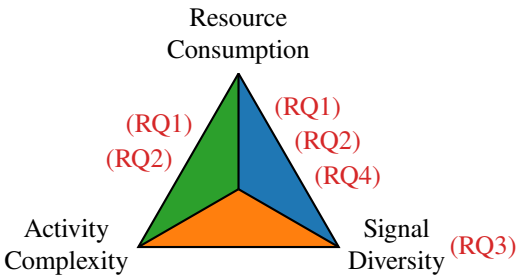



**Figure 1.3:** Tradeoff triangle of key challenges in WHAR labeled with this work's research questions.

(RQ1) How can WHAR signals and human activities be transformed into meaningful graph representations that GNNs can effectively process?

(RQ2) How suitable are GNNs for recognizing human activities in WHAR signals?

RQ1 focuses on the preparatory steps needed to leverage GNNs in the WHAR domain. Here, the key task is to design lightweight, resource-efficient transformations of sensor signals and activities into signal and activity graphs. RQ2 is a fundamental question that investigates whether GNNs can serve as viable alternatives to CNN- and RNN-based approaches for recognizing both basic and complex activities while maintaining computational efficiency.

A growing trend in the machine learning community is shifting from highly specialized models like the GNNs toward more generalized FMs, which learn from large datasets and adapt to various downstream tasks. Due to the diversity of sensor signals and the resource constraints of wearable devices, however, FMs are less thoroughly explored in the WHAR domain. Therefore, this work raises the following fundamental questions about the development of FMs for WHAR:

(RQ3) How can heterogeneous sensor signals be aligned for the effective development of comprehensive WHAR foundation models?

(RQ4) Can FMs improve the robustness and overall performance of real-world WHAR applications?

RQ3 addresses the challenge of aligning diverse wearable sensor signals as a necessary prerequisite for developing generalizable FMs. RQ4 examines the feasibility of on-device FMs for WHAR, considering both signal diversity and computational complexity.

## 1.4 Contributions

In sum, this dissertation advances the WHAR field by presenting

(C1) resource-efficient and reusable graph transformations that capture temporal dependencies in WHAR signals and complex activity sequences,

(C2) graph neural networks that facilitate lightweight activity recognition and sequential activity prediction,

(C3) a framework for harmonizing heterogeneous signals from various sensors that allows general-purpose WHAR models to learn from large-scale data, and

(C4) foundation models that enable precise signal forecasting and classification through novel dual-view and multi-resolution signal analysis.

Figure 1.4 maps this work's contributions onto the tradeoff triangle. As demonstrated through empirical validation against state-of-the-art approaches on publicly available benchmark datasets, these contributions push the boundaries of WHAR by enabling the efficient handling of diverse sensor signals and the recognition of both basic and complex activities in resource-constrained environments. Significant parts of this work have already been published in peer-reviewed journals or presented at academic conferences [1, 2, 3, 4].

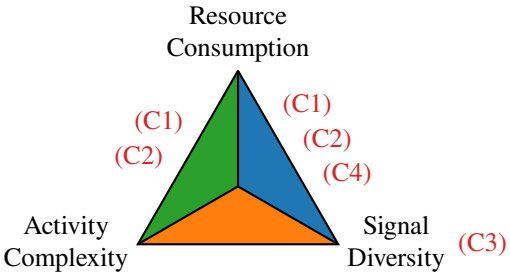



**Figure 1.4:** Tradeoff triangle of key challenges in WHAR labeled with this work's contributions.

## 1.5 Thesis Structure

This dissertation is divided into four parts. Part I motivates the work, defines its goals and important terms, and describes state-of-the-art; Part II explores the integration of GNNs into WHAR; Part III examines the feasibility of FMs in the WHAR domain; and Part IV concludes this thesis with an overview of its findings and potential future directions. Figure 1.5 presents a detailed view of the relationships between the parts and their corresponding chapters. Solid arrows indicate informational dependencies, where the source chapter provides important context or inspiration for the target chapters, while dashed arrows represent necessary dependencies, where the source chapter provides essential foundations for the target chapters. The following paragraphs outline the individual chapters.

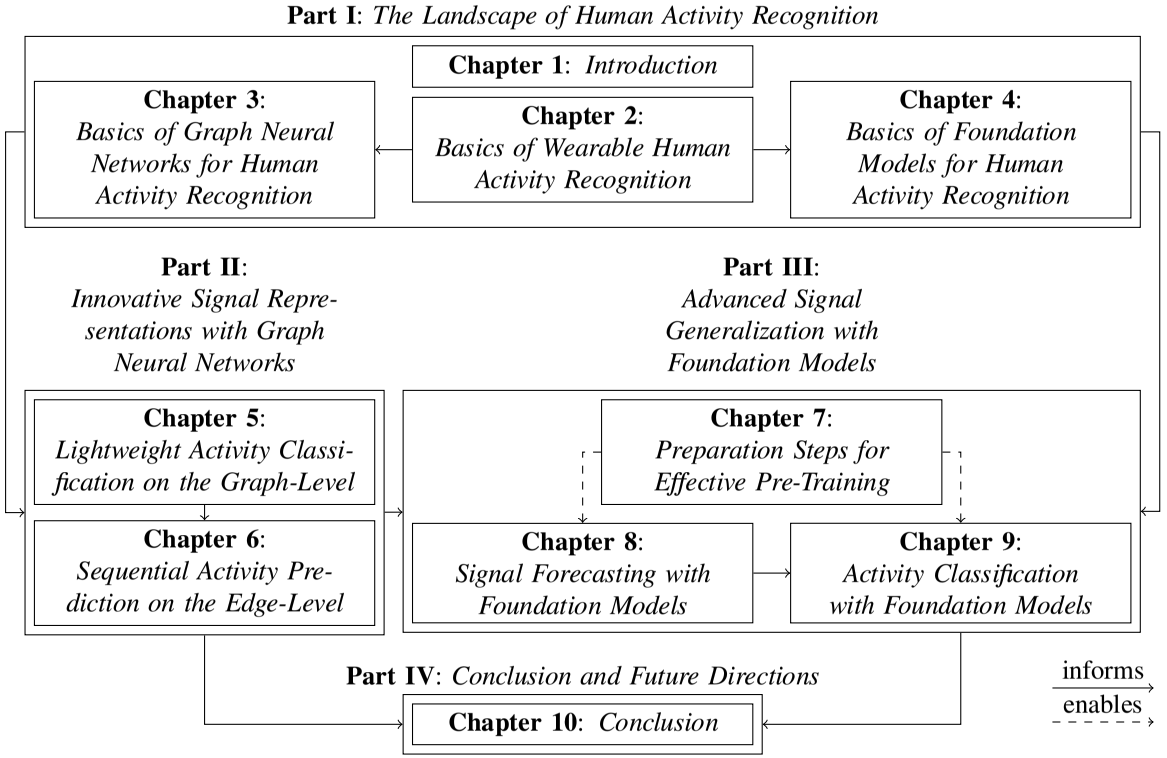



**Figure 1.5:** Thesis outline and interplay between chapters.

Chapter 1 motivates the research by highlighting key challenges of WHAR and the relevance of innovative signal representations and machine learning for solving these challenges.

Chapter 2 introduces terms like signals, micro-activities, and macro-activities, along with common WHAR tasks like activity classification and signal forecasting. Moreover, it describes the Discrete Wavelet Transform (DWT) as a sophisticated pre-processing technique for sensor signals, presents popular benchmark datasets, and discusses the key challenges of WHAR in more detail. Thus, this chapter lays the foundation for all subsequent chapters.

Chapter 3 explains the theory behind GNNs, covering important concepts such as graph representations, GNN layer types, graph-level classification, and graph link prediction. It closes with an in-depth literature review of GNN applications in the WHAR domain. This chapter is the basis for Chapters 5 and 6.

Chapter 4 presents a general definition of the term Foundation Model and its adaptation to the WHAR domain. It also discusses related concepts like self-supervised pre-training and task-specific fine-tuning. This chapter concludes with an overview on previous research efforts regarding self-supervised pre-training and cross-dataset generalization in WHAR. This chapter is the basis for Chapters 7, 8, and 9.

Chapters 5 and 6 explore signal- and activity-graph transformations that enable the use of GNNs in a variety of WHAR scenarios. They further present GNN models tailored to lightweight activity classification and sequential activity prediction and evaluate their prediction quality on several benchmark datasets.

Chapter 7 presents signal alignment steps that enable the effective pre-training of FMs for WHAR. Chapters 8 and 9 present novel, WHAR-tailored FM architectures, train them on the aligned signals from eight publicly available datasets, and evaluate their generalizability and adaptability to the downstream tasks signal forecasting and activity classification.

Finally, Chapter 10 reflects on the main contributions of this work and explores their implications for future research.

# 2 Basics of Wearable Human Activity Recognition

WHAR is the process of identifying and classifying human activities through the analysis of signals from wearable sensors (e.g., accelerometers, gyroscopes) [56, 89]. This process involves pre-processing techniques such as segmentation and filtering, alongside ML and deep learning methods [56, 89]. Consequently, WHAR is a multi-facetted research field that aims at delivering personalized user feedback and skill assessments across various application domains, including sports and fitness, daily living, and healthcare [56, 89]. This chapter explains key concepts, including definitions of the terms signal, activity classification, and signal forecasting. Additionally, it outlines common pre-processing techniques and datasets, as well as current challenges in WHAR.

## 2.1 Signals, Users, and Activities

Wearable sensors continuously measure $f$-dimensional signals over time. ML approaches typically process these time-series data in short, fixed-length windows $s = (s_1, s_2, \ldots, s_w) \in \mathbb{R}^{w \times f}$ that are shifted through the signal. Each signal window contains $w$ multi-dimensional sensor readings, where each data point $s_t \in s$ represents the $f$-dimensional sensor measurement at time $t \in \{1, \ldots, w\}$. Figure 2.1 illustrates a sliding window on a three-axis accelerometer signal. From this point onward, this thesis uses the terms signal and window interchangeably.

During each time window, a user performs an activity, which can be either dynamic (e.g., walking or jumping) or static (e.g., sitting or standing still). Each activity

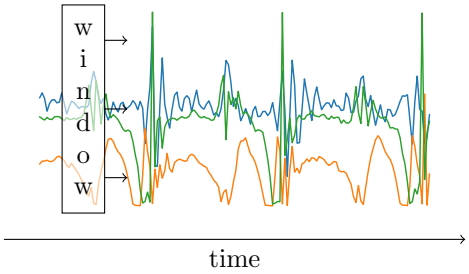



**Figure 2.1:** Sliding window that is shifted through a three-axis accelerometer signal recorded at the ankle during walking (signal from [13]).

is associated with a label $y \in Y$ (e.g., walking), where $Y$ is the set of detectable activities.

Detectable activities can vary in granularity. Ramírez et al. [117] define high-level activities (e.g., shopping) as longer-lasting actions that are composed of several low-level activities (e.g., walking). Alia et al. [9] describe macro-activities (e.g., preparing a sandwich) as complex tasks comprising several micro-activities (e.g., taking or cutting). Xie et al. [163] characterize complex activities (e.g., dumbbell curls or rope skipping) as sequences of meta-activities (e.g., certain arm movements). Liu et al. [101] introduce motion units as fine-grained phases of movement (e.g., stance or swing in walking). Similarly, Gehrig [51] decomposes concurrent or sequential actions (e.g., slicing an apple) into motion primitives (e.g., taking, cutting).

Given the variety of terms and their overlapping meanings, this work proposes the following activity hierarchy:

$$\text{motion unit} \subset \text{micro-activity} \subset \text{macro-activity}, \tag{2.1}$$

where motion units represent fine-grained limb movements (e.g., fine hand rotations or certain arm movements), micro-activities describe basic low-level activities (e.g., walking, running, taking, cutting), and macro-activities capture complex high-level tasks (e.g., compound sports exercises or cooking recipes) that involve

multiple concurrent or sequential steps. Table 2.1 organizes the terms from the literature according to this hierarchy.

**Table 2.1:** Activity hierarchy with related terms from literature.

| motion unit | $\subset$ | micro-activity | $\subset$ | macro-activity |
|---|---|---|---|---|
| motion units [101] | | micro-activities [9] | | macro-activities [9] |
| meta-activities [163] | | motion primitives [51] | | high-level activities [117] |
| | | low-level activities [117] | | complex activities [163] |

While the majority of this thesis deals with the recognition of independent activities in signal windows, Chapter 6 focuses on the detection of micro-activities $m_t$ and sequential macro-activities (or micro-activity sequences) $M_T = (m_1, m_2, \ldots, m_T)$ with sequence index $t \in \{1, \ldots, T\}$. The micro-activities $m_t$ label signal windows $s \in \mathbb{R}^{w \times f}$. Figure 2.2 illustrates the table tennis exercise "Wide Falkenberg" (cf. [118]) as an example macro-activity. This exercise consists of three subsequent micro-activities, i.e., certain table tennis strokes.

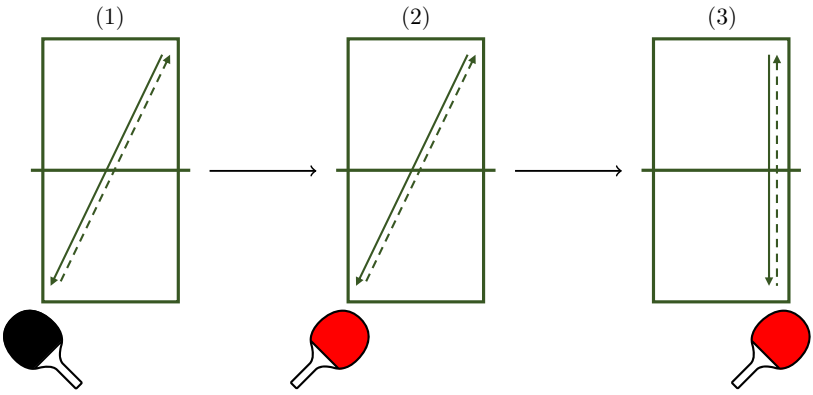



**Figure 2.2:** Table tennis exercise "Wide Falkenberg" (macro-activity) with three subsequent strokes (micro-activities): (1) backhand (black) from the backhand side of the table, (2) forehand (red) from the backhand side, and (3) forehand (red) from the forehand side. Solid arrows illustrate trajectories of incoming balls, dashed arrows indicate trajectories of the returns.

## 2.2 Signal Pre-Processing with Wavelets

WHAR applications utilize either raw or pre-processed signals. Common pre-processing techniques include, for example, filtering with high-pass, low-pass, or band-pass filters [111], the Fast Fourier Transform (FFT) to extract frequency-domain features [111], or the Wavelet Transform (WT) to capture time-frequency components in signals [107, 144]. Due to the time-localized nature of human activity signals, the WT excels in extracting meaningful features that are crucial for precise WHAR [102, 136]. Therefore, this section introduces the core concepts of the Discrete Wavelet Transform (DWT), which also forms the basis for the learnable DWT implemented in Chapter 9.

### 2.2.1 Discrete Wavelet Transform

The single-level DWT (Figure 2.3) decomposes an input signal $s$ into time-frequency features by convolving $s$ separately with a decomposition low-pass filter $g$ and a decomposition high-pass filter $h$ [107, 144]. Subsequently, the DWT down-samples the filtered signals with a stride of 2 ($\downarrow$) [144]. This process produces two sets of coefficients: approximation coefficients $cA$ represent the low-pass filtered signal and capture the overall trend of $s$; detail coefficients $cD$ represent the high-pass filtered signal and capture finer details and variations in $s$ [90]. The inverse DWT reconstructs a signal $s'$ from $cA$ and $cD$ by inserting a zero after each entry in $cA$ and $cD$ (up-sampling, $\uparrow$), applying a reconstruction low-pass filter $\tilde{g}$ to the upsampled $cA$ and a reconstruction high-pass filter $\tilde{h}$ to the upsampled $cD$, and summing the results [144]. Modifying the coefficients prior to the reconstruction step alters the reconstructed signal [144].

The multi-level DWT, on the other hand, offers a more fine-grained analysis of an input signal $s$. The maximum decomposition level $J$ is defined as

$$J = \left\lfloor \log_2 \left( \frac{|s|}{|g| - 1} \right) \right\rfloor, \tag{2.2}$$

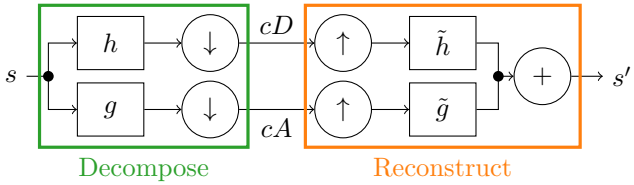



**Figure 2.3:** Illustration of a single-level DWT (adapted from [4, 144]).

where $|.|$ denotes the length of the input signal or the filter, respectively [90]. For each level $j \in \{1, \ldots, J\}$, the multi-level DWT applies the decomposition step of the single-level DWT iteratively to the approximation coefficients $cA_{j-1}$ of its previous level [144]. The multi-level Wavelet decomposition produces $J + 1$ Wavelet coefficients $(cD_1, cD_2, \ldots, cD_J, cA_J)$ [90]. The detail coefficients $cD_j$ capture frequency components in $[f_s/2^j, f_s/2^{j-1}]$, where $f_s$ is the sampling rate. The approximation coefficients $cA_J$ represent the low-frequency content of $s$ with frequencies below $f_s/2^J$.

Orthogonal Wavelets, such as the Haar Wavelet [57] or Daubechies [34], ensure perfect reconstruction ($s' = s$) as long as $cA$ and $cD$ are not altered before reconstruction [144]. A possible method for constructing orthogonal Wavelet filter banks $(g, h, \tilde{g}, \tilde{h})$ is given as follows [144]:

1. Initialize a low-pass filter $g$ that satisfies the orthogonality constraint [144]:

$$\sum_{k=0}^{K-1} g(k)g(k-2n) = \delta(n), \tag{2.3}$$

where $K$ is the filter size, $n \geq 0$ is a shifting parameter, and

$$\delta(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n \neq 0 \end{cases} \tag{2.4}$$

is the Kronecker delta function.

2. Derive an orthogonal high-pass filter $h$ as the alternating flip of $g$ [144]:

$$h(k) = (-1)^k g(K-k). \tag{2.5}$$

3. Derive orthogonal reconstruction low-pass and high-pass filters $\tilde{g}$ and $\tilde{h}$ by reversing $g$ and $h$ [144]:

$$\tilde{g}(k) = g(K-k), \tag{2.6}$$
$$\tilde{h}(k) = h(K-k). \tag{2.7}$$

### 2.2.2 Related Work

Researchers leverage the DWT to reduce noise in sensor signals by modifying or suppressing Wavelet coefficients [158, 159]. Moreover, decomposed Wavelet coefficients are also feasible as ML inputs as they closely resemble input signals [1, 102]. Similarly, scalograms, which are the intermediate representations from Continuous Wavelet Transforms (CWT), can improve WHAR tasks by providing a different view on the time-frequency variations within signals [105, 126, 136, 153].

Researchers also integrate the DWT directly into the processing pipeline of ML methods, rather than just applying it to the input as a pre-processing step. The proposed solutions either leverage literature-defined Wavelets or learn custom Wavelet filter banks from data.

Li et al. [94] introduce a DWT-based pooling operation that reduces the hidden dimensions of CNNs by decomposing their hidden representations with pre-defined Wavelets and keeping only the low-frequency approximation coefficients. Mei et al. [110] apply this pooling operation to WiFi-based HAR. Similarly, Dahou et al. [32] decompose the hidden representations of a CNN with a custom DWT block that leverages Daubechies Wavelet at different scales. Zhao et al. [174] propose a neural network layer that convolves input signals with a subset of discrete Wavelets from [90] that is selected with K-Means clustering. The resulting Wavelet coefficients are then weighted with a learned weight vector.

Wolter et al. [160] replace the gating operations in RNNs with custom layers that learn Wavelet filter banks with six coefficients to reduce the size of RNNs. The authors introduce loss functions that implement the perfect reconstruction and alias cancellation properties of Wavelets as product filters (cf. [144]). Gao et al. [47] propose a learnable Wavelet convolution that conditions its kernels with similar perfect reconstruction and alias cancellation losses. Recoskie [132] and Michau et al. [114] present CNN-based autoencoders that reflect the structure of the DWT. The decoders implement the decomposition step, while the encoders implement reconstruction. [132] learns a single filter bank for all decomposition levels, while [114] learns separate filter banks per decomposition level.

## 2.3 Activity Classification

Activity classification involves the identification and categorization of human activities using wearable sensor signals [56, 89]. It supports applications in different areas, such as daily living, fitness tracking, or health monitoring [56, 89]. Detectable activities range from motion units over low-level micro-activities to high-level macro-activities (cf. Section 2.1).

### 2.3.1 Task and Goal

The goal of activity classification is to successfully assign the correct activity class $y \in Y$ to a signal $s \in \mathbb{R}^{w \times f}$ using a learned classification function [53]

$$z : \mathbb{R}^{w \times f} \rightarrow Y, \tag{2.8}$$

where $w$ denotes the length of the signal in time steps (i.e., a time window), $f$ represents the number of features per time step, and $Y$ is the set of possible activity classes. The predicted activity class $\hat{y}$ is therefore given by

$$\hat{y} = z(s). \tag{2.9}$$

$z$ is typically represented by a ML classifier, which learns its classification capability from labeled training data. Traditional ML models, such as SVMs or DTs, utilize handcrafted features extracted from sensor signals to classify activities based on statistical patterns [14, 66, 81]. These features include, for example, statistical measures, such as mean or variance, and frequency domain features. In contrast, modern approaches leverage neural networks, such as CNNs or Long Short-Term Memories (LSTMs, [67]), which automatically learn complex features from raw sensor data without the need for manual feature engineering [122, 159, 177].

Figure 2.4 illustrates the classification of a three-axis accelerometer signal with a classification function $z$ into one of five activities.

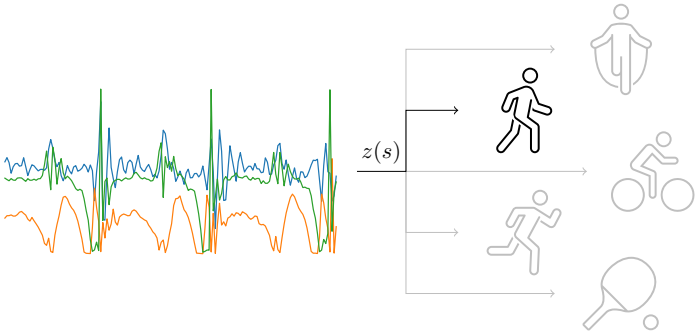



**Figure 2.4:** Classification of a three-axis accelerometer signal recorded at the ankle during walking with a learned classification function $z$ (signal from [13]).

### 2.3.2 Evaluation Metrics

Evaluation metrics measure the quality of $z$'s classifications. This work leverages the accuracy score [127]

$$\text{Acc} = \frac{TP}{N} \tag{2.10}$$

and the macro F1-score [127], which is also called the averaged F1-score [121]

$$\text{Macro F1} = \frac{1}{|Y|} \sum_{y \in Y} \frac{2TP_y}{2TP_y + FP_y + FN_y}. \tag{2.11}$$

The accuracy measures the proportion of correctly predicted samples (true positives, $TP$) out of the total number of samples ($N$) in a test set. The macro F1-score evaluates the performance for each class individually by considering true positives ($TP_y$), false positives ($FP_y$), and false negatives ($FN_y$) for each class $y \in Y$. It then averages these scores across all classes, giving equal importance to each class regardless of their frequencies in the dataset. Both metrics range from 0 (worst) to 1 (best), with accuracy focusing on overall correctness and macro F1-score balancing performance across all classes.

### 2.3.3 Related Work

State-of-the-art classifiers typically apply neural networks to detect activity patterns within sensor signals and to label them with a corresponding activity class. For instance, Ordóñez et al. [122] achieve great classification performance on two benchmark datasets by stacking four convolution layers and two LSTM layers. However, with more than one million parameters, this model surpasses the recommended size of 100k parameters of neural networks for wearable devices [166] by at least one order of magnitude. By dropping one LSTM layer, the parameter count decreases by approximately $62\,\%$ while the performance scores remain similar [17]. Zhou et al. [177] present a lightweight neural network that combines convolutional layers and LSTMs with advanced network components, such as Transformer Encoders and attention. This model outperforms [17] on three benchmark datasets while requiring less than 100k network weights.

As demonstrated by these publications, neural networks are universally applicable to both single- and multi-modal sensor signals from various sensor configurations. However, research has shown that activity classification benefits significantly from multi-modal signals gathered from multiple wearable sensors and sensor

placements [122, 158]. Building on activity classification, some researchers also explore applications that analyze and assess the skills of users in performing certain activities [81].

## 2.4 Signal Forecasting

Signal forecasting involves predicting future signal patterns based on past sensor readings, which enables the detection of anomalies within signals [173]. Additionally, forecast signals can be used to augment training datasets for activity classification tasks [79]. Other approaches predict future activities instead of signal curves [95, 124]. However, this work does not cover activity forecasting.

### 2.4.1 Task and Goal

Signal forecasting is a regression task with the goal of predicting future signal values $s' \in \mathbb{R}^{w' \times f}$ based on previous observations $s \in \mathbb{R}^{w \times f}$ using a forecasting function [53, 73]

$$z' : \mathbb{R}^{w \times f} \rightarrow \mathbb{R}^{w' \times f}, \tag{2.12}$$

where $w$ denotes the length of the previous signal in time steps, $w'$ is the number of predicted time steps, and $f$ represents the number of features per time step. The forecast signal $\hat{s}'$ is therefore given by

$$\hat{s}' = z'(s). \tag{2.13}$$

In this setup, $z'$ is typically fit on historical signal data. Traditional forecasting approaches, such as ETS (Error, Trend, Seasonality) or AutoRegressive Integrated Moving Average (ARIMA), rely on historical signal patterns to make predictions based on linear relationships within the data [73]. In contrast, modern approaches leverage neural networks, such as LSTMs or Generative Adversarial Networks (GANs) [54], to capture more complex, non-linear relationships in multivariate sensor signals [63, 79]. Beyond the WHAR domain, Transformer-based

Foundation Models [48] and fully-connected neural networks [26] have shown great success in general time-series forecasting.

Figure 2.5 illustrates the forecast of a three-axis accelerometer signal with a learned forecasting function $z'$.

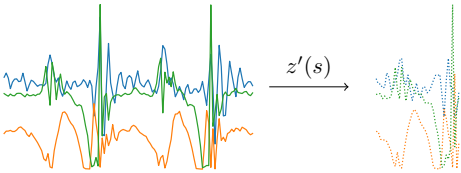



**Figure 2.5:** Forecast of a three-axis accelerometer signal recorded at the ankle during walking with a learned forecasting function $z'$ (signal from [13]).

### 2.4.2 Evaluation Metrics

This work evaluates the quality of $z'$'s forecasts with the mean absolute error [29]

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|s_i' - \hat{s}_i'| \in \mathbb{R}_{\geq 0} \tag{2.14}$$

and the root mean squared error [29]

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i' - \hat{s}_i')^2} \in \mathbb{R}_{\geq 0} \tag{2.15}$$

between $N$ sensor readings and their forecasts. The MAE measures the average absolute difference between the true future values $s'$ and the forecast values $\hat{s}'$. The RMSE squares these differences to penalize larger deviations more heavily and

then takes the square root to normalize the units. For both metrics, smaller values indicate better forecasting performance. Furthermore, the cosine similarity [29]

$$\text{CS} = \sum_{i=1}^{N} (\|s_i'\| \cdot \|\hat{s}_i'\|) \in [-1, 1] \tag{2.16}$$

measures the similarity of $s'$ and $\hat{s}'$ based on their shapes (higher values indicate greater similarity).

### 2.4.3 Related Work

Signal forecasting approaches focus on learning patterns from historical sequences to generate accurate future predictions. For instance, Kim et al. [82] forecast gyroscope signals using a neural network that incorporates Fourier Neural Operators [98], which combine the Fourier Transform with convolutional operations. The authors train user-specific models for the forecasting of walking and running signals from five users; one model per user. The forecasters achieve good MSEs on the test data of their respective users, but fail on the test data of other users. Thus, individuality between users is a limiting factor for the prediction quality. Jaramillo et al. [79] combine LSTMs with multi-head attention to forecast accelerometer signals from a subset of users and activities in the PAMAP2 dataset [133]. This model achieves favorable RMSEs for activities like walking, running, and stair navigation. Additionally, the authors train a downstream classifier on the forecast signals that performs comparably to a classifier that has been trained on the actual signals. Thus, forecast signals can effectively replace ground-truth signals in activity classification tasks.

GANs offer an alternative approach for signal generation. GANs consist of two components: a generator that creates synthetic data and a discriminator that evaluates its authenticity [54]. Hazra et al. [63] introduce a GAN that integrates LSTM and CNN layers to generate biomedical signals. By applying a DWT to smooth the signals before processing them through the GAN, the authors achieve significantly lower MAEs and RMSEs compared to state-of-the-art models. Li et

al. [96] develop a Transformer-based GAN for generating ECG and accelerometer signals. The evaluation demonstrates that this GAN generates signals that closely resemble the shape and distribution of the true signals, highlighting the potential of GANs for generating realistic signals.

## 2.5 Public Benchmark Datasets

Public benchmark datasets are essential for developing effective WHAR applications. They provide researchers with valuable insights into the application domain, enabling them to design solutions tailored to specific downstream tasks. Additionally, since most WHAR applications rely on ML models, representative datasets are crucial for training accurate models that generalize well to real-world scenarios.

To facilitate the implementation and comparability between approaches, many research groups have collected and published WHAR datasets over the past years (cf. [24]). This work utilizes the 13 established benchmark datasets listed in Table 2.2. These datasets provide raw, unprocessed sensor signals that are either available as continuous time-series or segmented into short signal windows. Some of the studies collect signals in controlled environments [87], simplifying the collection and labeling process, while others focus on real-world scenarios [146], where environmental effects can influence the signals. Moreover, the datasets differ in their available sensors (types, quantities, manufacturers, accuracies, placements), their scopes (e.g., daily living activities, sports activities, health-related activities), and the demographics and number of users involved in the studies. Therefore, the table presents the number of distinct activities $|Y|$, the number of users, the sampling rates, and the available inertial sensors for each dataset.

Some of the datasets also provide additional information, such as ambient temperature [92], air pressure [159], or the electrocardiogram of the users [13]. While these features are not relevant for this work on human activity recognition

**Table 2.2:** The WHAR datasets used by this work. $|Y|$: number of distinct activities, A: accelerometer, G: gyroscope, M: magnetometer. PAMAP2 provides signals of 12 protocol activities performed by most users and 6 optional activities with less participation.

| Dataset | $\|Y\|$ | Users | Sampling Rate | Sensors |
|---|---|---|---|---|
| Complex HAD [138, 139] | 13 | 10 | 50 Hz | A, G |
| Cooking [9, 88] | 3 | 4 | 50-100 Hz | A |
| Daphnet [12] | 2 | 10 | 64 Hz | A |
| DLA [92] | 17 | 8 | 100-256 Hz | A, G, M |
| DSADS [15] | 19 | 8 | 25 Hz | A, G, M |
| FLAAP [87] | 10 | 8 | 100 Hz | A, G |
| KU-HAR [141] | 18 | 90 | 100 Hz | A, G |
| mHealth [13] | 12 | 10 | 50 Hz | A, G, M |
| PAMAP2 [133] | 12 + 6 | 9 | 100 Hz | A, G, M |
| RealWorld (HAR) [146] | 8 | 15 | 50 Hz | A, G, M |
| TT-Strokes [159] | 8 | 2 | 25-50 Hz | A, G, M |
| UCI-HAR [10] | 6 | 30 | 50 Hz | A, G |
| USC-HAD [172] | 12 | 14 | 100 Hz | A, G |

from wearable inertial sensors, they could potentially enhance the performance of WHAR applications by providing richer contextual information.

## 2.6 Key Challenges

To enhance the effectiveness of WHAR applications in real-world environments, it is essential to address several key challenges. These challenges include the development of lightweight and resource-efficient models, the exploration and identification of complex activities in sensor signals, and a thorough consideration of signal diversity resulting from different sensors, sensor placements, and demographic variations among users [7].

### Resource Consumption.

The key enablers of ubiquitous WHAR are wearable sensing devices, such as smartphones, smartwatches, earables, fitness trackers, or future devices equipped with sensors [7, 56]. Although the performance of these devices improves with each new generation, their available computational resources, such as CPU power, memory, storage, and battery life, remain limiting factors for the development of novel, real-time capable solutions [7, 56]. As devices become smaller, these resource constraints become increasingly significant. For example, recent reports indicate that current smartwatches can handle neural networks with up to 100k parameters [166], which are sufficient for detecting basic activities or gestures in sensor signals. To enable comprehensive real-time applications, this work seeks for innovative model designs that reduce the number of required model parameters while maintaining high classification rates and short inference times on smartwatches.

### Complex Activities.

As stated in Section 2.1, human activities encompass a spectrum from fine-grained motion units, such as limb or wrist movements, to low-level, self-contained micro-activities like walking, running, or postural transitions, which are composed of several motion units. At the most complex level, macro-activities combine sequential or concurrent micro-activities into cohesive tasks, such as practicing a compound sports exercise or preparing a meal. The recognition of these macro-activities is further complicated by interactions with objects (e.g., cooking utensils) or other individuals (e.g., cooperative sports exercises) [7]. As the complexity of activities and their interrelationships grows, recognizing them in a resource-efficient manner becomes increasingly challenging.

Given the vast diversity of complex activities, this work focuses specifically on the processing and recognition of sequential activities. These activities can be effectively recognized from the perspective of a single individual, thereby eliminating the need for intricate modeling of interactions between users and objects.

### Data Diversity.

Wearable sensor signals vary depending on the sensor placement, sensor types, signal resolutions, and the performed activities (cf. Table 2.2). Moreover, demographic factors also influence the signals. For instance, people at various ages exhibit distinct movement patterns and men may execute activities differently than women. These variations, along with interruptions, waiting times, or labeling errors, can lead to notable discrepancies even in signals representing the same activity. For example, Figure 2.6 compares two acceleration signals recorded at the ankle during running. The signal patterns differ significantly, despite both being labeled with the same activity.

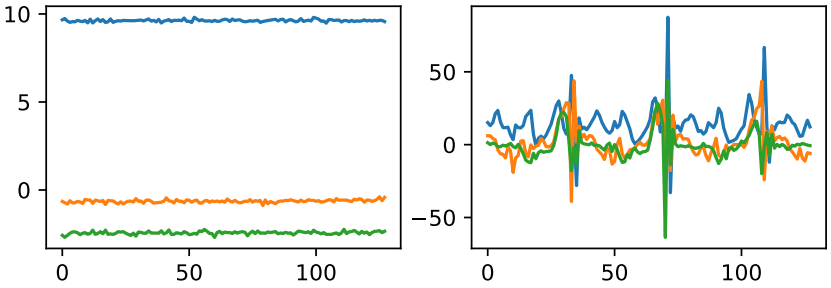



**Figure 2.6:** Three-axis accelerations recorded at the ankle during running (signals from [133]). The right plot contains a typical running pattern with distinct peaks corresponding to steps, while the left plot shows low-variance signals that are nearly constant, indicating minimal movement.

Comprehensive datasets that account for signal diversity are essential for developing powerful and general WHAR models [7]. However, large-scale collection and precise labeling of unbiased datasets that cover a wide variety of users and activities at different granularity is time-consuming and expensive. Consequently, many datasets only consider basic activities (e.g., walking, running) performed by a limited number of users (cf. Section 2.5). To overcome the limitations of

current datasets, this work explores methods for fusing sensor signals from diverse application scenarios and sensing configurations, thereby enabling a broader understanding of human activity patterns in sensor signals.

## 2.7 Summary

WHAR is a multifaceted field of research that aims to understand and interpret human activities in wearable sensor signals. These activities range from atomic limb movements, over self-contained micro-activities like walking or running, to complex macro-activities, such as playing cooperative sports exercises or preparing meals, that combine several micro-activities and even include interactions between people and objects. ML-based activity classifiers and signal forecasting models play a crucial role in managing this diverse spectrum of activities. Despite their potential, these tools face significant challenges due to the limited processing power of wearable devices and the lack of large-scale, versatile training datasets. The individuality of users further complicates the automatic recognition of activities in real-world scenarios.

Building upon these fundamental insights, this work explores innovative signal representations and fusion techniques, and develops advanced ML models that enhance wearable human activity recognition. In particular, this dissertation presents novel GNNs and FMs that enable versatile WHAR applications while addressing computational constraints and improving adaptability across users and activities. The subsequent Chapters 3 and 4 lay the foundation for the development of WHAR-tailored GNNs and FMs by introducing their key concepts and summarizing previous research efforts regarding the application of these architectures to the modeling of sensor signals and activity patterns.

# 3 Basics of Graph Neural Networks for Human Activity Recognition

With the increasing prevalence of wearable technology, efficient activity classification has emerged as a crucial research focus within the field of WHAR. Traditional classifiers often rely on CNNs [53] or RNNs [67], which excel at processing temporal information within wearable sensor signals [17, 177]. In contrast, Graph Neural Networks (GNNs) offer a flexible approach to modeling complex relationships within data by processing graphs with arbitrary vertex relationships [109, 150]. This chapter bridges the gap between GNNs and WHAR by explaining fundamental GNN concepts and exploring their latest advancements within the WHAR domain.

## 3.1 Graphs

According to [109], a graph $G = (V, E)$ consists of a set of vertices $V$ and a set of edges $E \subseteq V \times V$. Each edge $(v_i, v_j) \in E$ defines a directed connection from a source vertex $v_i \in V$ to a target vertex $v_j \in V$. In undirected graphs, all edges are bidirectional, i.e., $\forall (v_i, v_j) \in E : (v_j, v_i) \in E$. Edges are called self-loops if $v_i$ and $v_j$ are identical ($i = j$). Furthermore, edges can have weights $a_{ij} \in \mathbb{R}$ that define the strength of the relationships.

The neighborhood $N(v_i)$ of a vertex $v_i$ is the set of vertices $v_j$ that are connected to $v_i$ via an edge [40]:

$$N(v_i) = \{v_j \mid (v_i, v_j) \in E\}. \tag{3.1}$$

Figure 3.1a illustrates an unweighted, directed graph with self-loops; Figure 3.1b shows a weighted, undirected graph without self-loops.


**(a)** An unweighted, directed graph with self-loops. **(b)** A weighted, undirected graph without self-loops.

**Figure 3.1:** Example graphs with three vertices.

An adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$ is a square matrix that defines the edge relationships within a graph [31]. The rows and columns represent the graph vertices. An edge exists between two vertices $v_i, v_j \in V$ if $a_{ij} \neq 0$. In case of unweighted graphs, the adjacency matrix is binary with each cell $a_{ij} \in \{0, 1\}$. Tables 3.1a and 3.1b present the adjacency matrices of the example graphs from Figure 3.1.

**Table 3.1:** Adjacency matrices of the graphs from Figure 3.1.

**(a)** Unweighted adjacency matrix of graph 3.1a.

| | $v_1$ | $v_2$ | $v_3$ |
|---|---|---|---|
| $v_1$ | 1 | 1 | 1 |
| $v_2$ | 0 | 1 | 1 |
| $v_3$ | 0 | 0 | 0 |

**(b)** Weighted adjacency matrix of graph 3.1b.

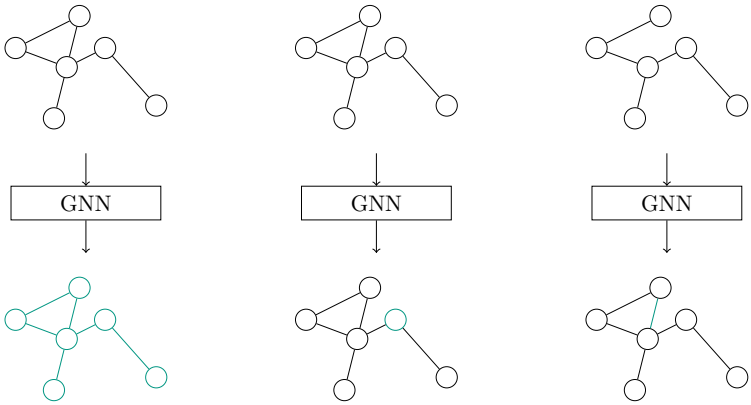
| | $v_1$ | $v_2$ | $v_3$ |
|---|---|---|---|
| $v_1$ | 0 | $a_{12}$ | $a_{13}$ |
| $v_2$ | $a_{12}$ | 0 | $a_{23}$ |
| $v_3$ | $a_{13}$ | $a_{23}$ | 0 |

This work uses sparse adjacency matrices with many cells being zero. A storage efficient method for handling sparse matrices is the coordinate list format (COO), which represents sparse matrices with three lists that hold the row indices, the column indices, and the values of the filled matrix cells [152].

## 3.2 Graph Neural Networks

GNNs represent a powerful class of neural networks that leverage the inherent structure of graphs to solve three distinct prediction tasks [150]:

1. For graph-level tasks, GNNs predict a label or property of the whole graph (Figure 3.2a).
2. For vertex-level tasks, GNNs predict a label or property of a single vertex (Figure 3.2b).
3. For edge-level tasks, GNNs predict the existence or attributes of edges between two vertices (Figure 3.2c).




**(a)** Graph-level **(b)** Vertex-level **(c)** Edge-level

**Figure 3.2:** Graph-level, vertex-level, and edge-level prediction tasks. The predicted graph components are highlighted in teal color.

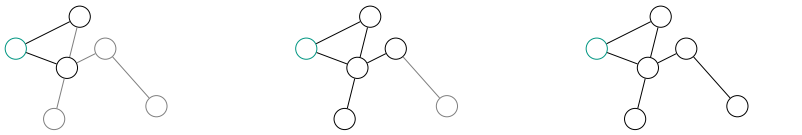
According to [20, 150], GNNs operate on graphs $G = (V, E)$ where each vertex $v_i \in V$ has an $f$-dimensional feature vector $x_i \in \mathbb{R}^f$ that describes its internal state. Consequently, $X \in \mathbb{R}^{|V| \times f}$ represents the vertex feature matrix. The edges $E$ are usually represented as adjacency matrices $A^{|V| \times |V|}$. Thus, the input to a GNN is a graph $G = (X, A)$.

GNNs iteratively update the graph representations by aggregating features from neighboring vertices using stacked GNN layers. The message passing framework [52] provides the mathematical foundation for these layers. Following the definitions in [20, 52, 150], message passing can be expressed as follows:

$$x_i' = \phi(x_i, \oplus_{v_j \in N(v_i)} \psi(x_i, x_j, a_{ij})), \tag{3.2}$$

where the learnable message passing function $\psi$ computes messages between a vertex $v_i$ and its neighbor $v_j \in N(v_i)$, which may incorporate edge weights $a_{ij}$. $\oplus$ is a permutation-invariant aggregation function (e.g., sum or mean), applied element-wise across the feature dimensions of the neighboring vertices $v_j \in N(v_i)$ (cf. Equation 3.1). $\phi$ is a learnable update function that transforms the aggregated messages and sets the updated vertex feature vectors $x_i'$.

The receptive field of a vertex $v_i$ is given by its direct neighbors $N(v_i)$ [20]. Stacking multiple GNN layers expands this field as each layer successively updates vertex features by aggregating information from an increasingly large set of neighbors. Consequently, in a GNN with $k$ layers, each vertex can incorporate information from vertices up to $k$ hops away. However, stacking too many layers increases the risk of over-smoothing vertex features [93]. Thus, common GNN architectures use less than four layers [175]. Figure 3.3 illustrates the receptive field expansion for a vertex after applying up to three GNN layers to a graph.



**(a)** Receptive field after one layer. **(b)** Receptive field after two layers. **(c)** Receptive field after three layers.

**Figure 3.3:** Receptive fields (black) of the left-most vertex (teal) after applying one (Figure 3.3a), two (Figure 3.3b), and three (Figure 3.3c) GNN layers to a graph.

Common GNN layers include the graph convolutional layer (GCN) [86], the gated graph convolutional layer (GGC) [97], or the graph attention layer (GAT) [151].

The following paragraphs explain the GNN layers that are relevant for this dissertation. This work implements GNNs with the Spektral API [55] in TensorFlow [5] and Keras [29].

### Gated Graph Convolutional layer (GGC).

The GGC layer updates graph representations by integrating message passing (cf. Equation 3.2) with a Gated Recurrent Unit (GRU) [27] as follows [55, 97]:

1. Given a desired number of hidden channels $c \geq f$, where $f$ is the dimensionality of the input vertex feature vectors $x_i \in \mathbb{R}^f$, pad each feature vector to match the number of hidden channels:

$$h_i^{(0)} = x_i \| \mathbf{0}, \tag{3.3}$$

   where $\mathbf{0} \in \mathbb{R}^{c-f}$ is a zero vector of size $c - f$.

2. Update the hidden states $h_i^{(l)} \in \mathbb{R}^c$ by iteratively applying the following operations $L$ times:

$$m_i^{(l)} = \sum_{v_j \in N(v_i)} h_j^{(l-1)} W^{(l)}, \tag{3.4}$$

$$h_i^{(l)} = \text{GRU}(m_i^{(l)}, h_i^{(l-1)}), \tag{3.5}$$

   where $m_i^{(l)}$ represents the aggregated messages from neighboring vertices and $h_i^{(l)}$ are the updated hidden states after the $l$-th iteration. $W^{(l)} \in \mathbb{R}^{c \times c}$ is a trainable weight matrix. GRU updates the hidden states based on the current messages $m_i^{(l)}$ and the previous hidden states $h_i^{(l-1)}$.

3. After $L$ iterations, update the vertex feature vectors with the final hidden states $h_i^{(L)}$:

$$x_i' = h_i^{(L)}. \tag{3.6}$$

By combining message passing with GRUs, the GGC layer effectively captures both local graph structure and temporal dependencies between vertices.

### Graph Attention layer (GAT).

The GAT layer updates vertex features by incorporating masked self-attention into the message passing framework [55, 151]:

$$X' = \alpha XW + b, \tag{3.7}$$

where $W \in \mathbb{R}^{f \times c}$ is a trainable weight matrix with $c > 0$ channels, $b \in \mathbb{R}^c$ is a bias term, and $\alpha$ is the attention matrix that weights the importance of each vertex:

$$\alpha_{ij} = \frac{e^{\left(\text{LeakyReLU}\left(a^\top [(XW)_i \parallel (XW)_j]\right)\right)}}{\sum_{v_k \in N(v_i)} e^{(\text{LeakyReLU}(a^\top [(XW)_i \parallel (XW)_k]))}}, \tag{3.8}$$

where $a \in \mathbb{R}^{2C}$ is a trainable weight vector. In contrast to the standard softmax, the sum in the denominator does not iterate over all vertices of the input graph but considers only the vertices in the neighborhood of the source vertex $v_i$. This modification enables the attention mechanism to incorporate the structure of the input graph in the calculation of the attention scores.

### Pooling layers.

Pooling layers change the graph structure or determine global graph features by extracting certain vertices or edge relationships. Top-$k$ pooling [22, 46], for example, multiplies the vertex feature vectors $x_i$ with an $f$-dimensional vector $p$ and extracts the $k > 0$ vertices with the largest values $y_i = x_i p$ and their edge relationships. Global pooling layers, on the other hand, aggregate the entire set of vertex feature vectors and output a single $f$-dimensional vector $g$ that represents the whole graph. This work leverages average (AVG) [55] and maximum (MAX) [55] pooling, as well as custom minimum (MIN) [1], standard deviation (STD) [1], and last vertex pooling (LVP) [2] operations as global feature aggregation functions:

$$g_j^{\text{AVG}} = \frac{1}{|V|} \sum_{i=1}^{|V|} x_{ij}, \tag{3.9}$$

$$g_j^{\text{MAX}} = \max_{i=1}^{|V|} x_{ij}, \tag{3.10}$$

$$g_j^{\text{MIN}} = \min_{i=1}^{|V|} x_{ij}, \tag{3.11}$$

$$g_j^{\text{STD}} = \sqrt{\frac{1}{|V|} \sum_{i=1}^{|V|} (x_{ij} - g_j^{\text{AVG}})^2}, \tag{3.12}$$

$$g_j^{\text{LVP}} = x_{|V|j}, \tag{3.13}$$

where $x_{ij}$ represents the $j$-th feature of the $i$-th vertex and $|V|$ denotes the number of vertices in a graph.

## 3.3 Applications in Wearable Human Activity Recognition

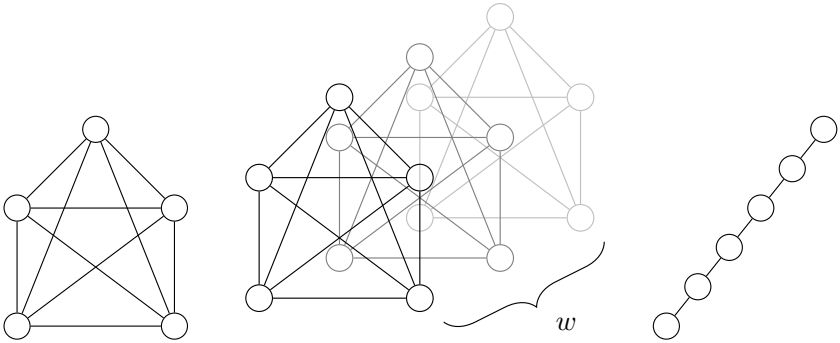
Although wearable sensor signals are typically represented as time-series, converting them into graphs enables the application of GNNs, which can capture complex structural dependencies in the data. To ensure competitive recognition rates, such graphs must accurately reflect essential signal properties. As demonstrated in [113], GNNs can outperform state-of-the-art classifiers like DeepConvLSTM [122]. Table 3.2 categorizes previous GNN research with applications in WHAR based on the GNN task (cf. Figure 3.2), the graph representations, the edge relationships, and the vertex representations.

The vast majority of researchers present GNNs that operate on the graph-level. These approaches typically employ signal-graph transformations, where graphs represent either (1) a signal window (window graphs) or (2) an individual time step (time step graphs). Additionally, some methods construct (3) activity graphs, where vertices correspond to activities or key objects involved in executing a specific activity. Figure 3.4 illustrates examples of these three graph structures. However, other edge relationships and vertex representations are also possible,

**Table 3.2:** State-of-the-art GNNs for WHAR. Italic references mark this work's publications. MHAP [169]: multivariate highly activated period.

| Task | Graph Representations | Edge Relationships | Vertex Representations | References |
|---|---|---|---|---|
| graph-level | window | fully-connected | channels | [71] |
| | | | locations | [113] |
| | | | sensors | [157] |
| | | similarity | channels | [100, 155, 168] |
| | | skeleton | locations | [103] |
| | | skeleton, sensor type, similarity | sensors | [25] |
| | | skeleton, temporal | locations, time steps | [167] |
| | | temporal | MHAPs | [169] |
| | | | time steps | *[1]* |
| | time step | fully-connected | sensors | [23] |
| | | skeleton | locations | [58, 154, 176] |
| | activities | temporal | windows | [65, 116] |
| | activities, objects | external knowledge | word embeddings | [72, 123] |
| vertex-level | activities | fully-connected | windows | [115, 119] |
| | | similarity | windows | [135] |
| edge-level | activities, users, context | hyper-edges | activities, users, context | [49, 50] |
| | activities | temporal | windows | *[2]* |

especially when switching between graph-level, vertex-level, and edge-level recognition tasks (cf. Table 3.2).

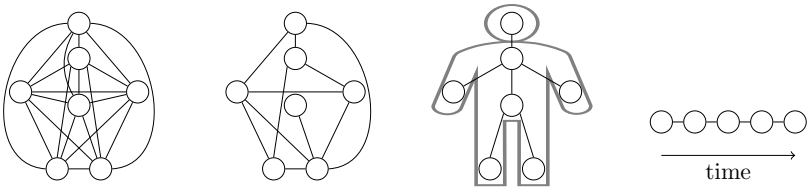



**(a)** Window graph **(b)** Time step graph. **(c)** Activity graph.

**Figure 3.4:** Example window, time step, and activity graphs. For instance, [71] presents a fully-connected window graph (a), where each vertex represents a sensor channel (e.g., x-axis acceleration) or an output channel of a CNN; [23] presents a fully-connected time step graph (b), where each vertex represents the reading of a specific sensor at a given time step within a window of length $w$; and [116] presents a linear activity graph (c) that represents a basic activity, where each vertex contains the average sensor readings during a sliding window of that activity.

Typically, graphs represent fixed-length signal windows or sensor readings at particular time steps and are either fully-connected (Figure 3.5a), partially connected (Figure 3.5b) based on a similarity metric (e.g., Pearson correlation, Mahalanobis distance, cosine similarity), structured like the human skeleton (Figure 3.5c), or connected according to the temporal order of their vertices (Figure 3.5d). Instead of representing sensor signals as graphs, some researchers model interactions between activities using graphs with partially-connected edges, temporally connected edges, hyper-edges that connect activity, user, and context vertices, or edges derived from external knowledge bases.

In skeletal graphs, vertices represent either the sensor locations on the human body or derived body joints; edges connect vertices according to the spatial structure of the human skeleton. Researchers that leverage skeletal graphs follow the premise




**(a)** Fully-connected **(b)** Partially connected **(c)** Skeleton **(d)** Temporal

**Figure 3.5:** Illustrations of graph structures that have been used in WHAR literature. Fully-connected graphs (3.5a) connect all vertices with each other. Partially connected graphs (3.5b) connect vertices based on similarity metrics or external knowledge. Skeletal graphs (3.5c) connect vertices according to the spatial structure of the human skeleton. Temporal graphs (3.5d) connect vertices through time. Self-loops can improve the graph structures.

that the human skeleton restricts the movement radius of physically connected body parts [103]. Connected body parts cannot move independently and thus, sensor signals from such body parts exhibit correlating properties [103]. However, some movements or activities also exhibit parallels between unconnected body parts that can be explored based on the similarities of their signals [167].

Vertices either represent individual sensor channels (i.e., the x-, y-, and z-axis of a three-axis accelerometer are three different vertices), all channels of one or more sensors, the sensor locations on the human body, or body joints. The vertex features vary between signal windows, signals at time steps, or aggregated signal representations. Signal aggregations can be simple statistical functions, such as the mean or the standard deviation, complex signal representations, such as spectrograms, or features derived from inherent neural networks, e.g., CNN channels, hidden states of neural networks, or multivariate highly activated periods (MHAPs). Some researchers also investigate the coupling of WHAR with word embeddings from external knowledge bases.

The following sections briefly describe the publications from Table 3.2, focusing on signal-graph transformations and graph processing.

### 3.3.1 Graph-Level Recognition

Graph-level approaches leverage signal-graph and activity-graph transformations to construct window graphs, time step graphs, or activity graphs (cf. Figure 3.1).

#### Window Graphs.

Huang et al. [71], Miao et al. [113], and Wang et al. [157] construct weighted fully-connected graphs from signal windows. The vertices represent sensor channels (e.g., x-axis acceleration) or output channels of an upstream CNN [71], combine all axes of a single sensor (e.g., three-axis acceleration) [157], or all axes from all sensors at a given sensor location [113]. The researchers leverage different similarity metrics to weight the interaction between vertices. [71] sets the edge weights to the softmax of the negative square distances between two vertex feature vectors, [113] learns edge weights via a two-layered CNN with softmax output, and [157] takes the softmax of the dot product similarity between two vertex feature vectors. [71] and [113] update the vertex representations by summing the weighted features vectors in their neighborhoods, while [157] presents a hierarchical graph pooling layer to compute vertex representations.

Another application for similarity metrics is the construction of partially connected graphs. Liao et al. [100], Wang et al. [155], and Yang et al. [168] combine Pearson correlation with thresholds to build partially connected graphs. The vertices represent individual sensor channels during short signal windows. Two vertices are connected if the Pearson correlation between their feature vectors exceeds a predefined threshold. [100] and [168] construct unweighted graphs, while [155] uses the correlation coefficients as edge weights. [100] leverages Chebyshev graph filtering with residual connections to update the graph; [168] uses ChebNet layers from [36]. [155] implements a custom graph attention layer that can handle weighted graphs.

Liu et al. [103], Chen et al. [25], and Yan et al. [167] leverage skeletal information to transform signal windows into partially connected graphs. In [103], vertices represent the sensor signals at different body positions during fixed-length windows and unweighted, undirected edges link vertices of connected body parts.

[103] leverages message passing with a customized gating function to update the vertex features. In [25], vertices represent individual sensors. In addition to the skeleton graph, [25] builds two additional graphs with the same vertex representations to gain more detailed insights from signal windows. The first graph connects vertices that represent the same sensor type; the second graph is a fully-connected graph that weights its edges based on the cosine similarity between the features of adjacent vertices. [25] updates the vertex feature vectors using separate GCNs per graph. [167] builds graphs that fuse spatial and temporal information. The vertices represent the sensor locations on the human body. The vertex features are the sensor readings at a given point in time during a window. Spatial edges connect vertices from the same time step based on the human skeleton, while temporal edges link vertices that represent the same body location in their temporal sequence. [167] processes the graphs with alternating GCN and CNN layers.

Younis et al. [169] take a unique approach to derive partially connected graphs from signal windows. Vertices represent multivariate highly activated periods (MHAPs) in CNN layers. Directed edges connect MHAP vertices within the same layer and between subsequent layers. The authors apply DeepWalk [128] to generate fixed-size graph embeddings from the resulting structures.

#### Time Step Graphs.

Cao et al. [23] construct three weighted fully-connected graphs per time step within a sliding window, where each vertex represents the reading of a specific sensor at a given point in time. The first graph is a similarity graph. It updates the vertex features using a multi-layer perceptron. The edge weights are the pairwise cosine similarities between the updated vertex feature vectors. The second graph is a discrete graph. It updates the vertex features and the edge weights based on the Gumbel reparameterization [78, 106, 137]. The third graph is a dilated graph. The edge weights of the dilated graph represent the cosine similarities between the hidden representations of two vertices after applying a dilated 1D convolution to the vertex feature vectors. [23] processes these time step graphs separately using GAT layers and aggregates them using a Graph LSTM [99].

Han et al. [58], Wang et al. [154], and Zhao et al. [176] build unweighted time step graphs that leverage the spatial structure of the human skeleton. In [58] and [176], vertices contain sensor signals at a given point in time. [176] introduces additional vertices that represent the coordinates of body joints in videos. [154] derives body joints from the sensor placements, the length of the users' limbs, accelerometer signals, and gyroscope signals. The coordinates of these body joints are the vertex feature vectors. In all three publications, the edges represent natural skeletal relationships. [58] uses ChebNet layers to update the graphs, [176] uses graph convolutions, and [154] uses GCNs. The researchers use either CNNs [176], LSTMs [154], or both [58] to aggregate subsequent time step graphs.

#### Activity Graphs.

Mondal et al. [116] and He et al. [65] model graphs per user and activity, where each graph represents a basic activity. Each vertex contains the average signal features during a sliding window of that activity. Undirected, unweighted edges connect the vertices based on their temporal order, forming linear graphs. [116] processes activity graphs with GCN and Top-$k$ pooling layers; [65] uses GAT.

Huang et al. [72] combine WHAR graphs with external knowledge graphs. The resulting heterogeneous graphs contain vertices that either represent an activity or an object. The authors detect activities in sensor signals and video clips using LSTMs and recognize relevant objects in the video streams using Inception-V3 [145]. They further use the English subgraph of ConceptNet [143] to initialize the vertex feature vectors with their corresponding word embeddings and to derive edges between activity and object vertices. The authors implement three separate GNNs to model activity-object relationships with the external knowledge graph. In [123], Pan et al. introduce a knowledge-aware human activity classifier for few-shot learning that also leverages external knowledge from the English subgraph of ConceptNet to build graphs with activity and objects vertices.

### 3.3.2 Vertex-Level Recognition

Mohamed et al. [115] and Nian et al. [119] exploit the idea that human activities follow a natural order (e.g., showering after exercising [115] or stretching before exercising [119]) to classify unlabeled activities based on surrounding activities. The researchers construct fully-connected, unweighted graphs, where vertices represent signal windows. The vertex representations are split into two parts: the sensor signals during the window and the label of the performed activity. If a label is unknown, the label part of the feature vector is zero. [115] leverages GCN and CNN layers to infer missing labels; [119] uses GCN and fully-connected layers.

Sarkar et al. [135] follow a similar approach as [115, 119]. However, they model partially connected activity graphs per user instead of fully-connected graphs. Each vertex represents a fixed-size window of a certain activity. The vertex feature vectors are the aggregated features of a window, e.g., mean, skewness, or standard deviation, and their activity labels. Edges connect two vertices if the Mahalanobis distance of their features is less than a given threshold. The proposed model contains up to three GCN layers and two subsequent fully-connected layers.

### 3.3.3 Edge-Level Recognition

Ge et al. [49] model user-activity-context relationships as heterogeneous hyper-graphs with weighted, undirected hyper-edges. A vertex represents either a user, a context, or an activity. The vertex features are the average values of all signals that belong to the corresponding activity, user, or context. Since a user can perform various activities within the same context (e.g., the same phone placement), each weighted hyper-edge connects a user vertex with a context vertex and at least one activity vertex. The edge weights are the average values of signals that belong to the corresponding user-activity-context triples. The authors present a custom GNN that classifies the activities and context of heterogeneous hyper-edges. This is an edge-level recognition tasks, since each new input signal represents a hyper-edge that connects heterogeneous vertices of unknown classes. In [50], the

same authors improve their context-aware GNN by adding support for incomplete hyper-edges, i.e., edges that have no information regarding the activity or context. Furthermore, they improve the distinguishability between the features of different vertex types by adding a contrastive loss function.

## 3.4 Summary

GNNs facilitate the seamless integration of domain knowledge into WHAR applications through flexible graph structures, enhancing their effectiveness and versatility in real-world scenarios. Additionally, shallow architectures and minimal layer counts make GNNs particularly well-suited for wearable applications, where resource constraints demand efficient and compact network designs. For these reasons, GNNs represent a valuable alternative to traditional neural network architectures for WHAR. However, many existing solutions rely on signals from multiple body locations to create comprehensive skeletal graphs (e.g., [25, 103, 167]), construct graphs that overlook temporal information in sensor signals (e.g., [100, 155, 168]), or involve large models with one million or more trainable parameters (e.g., [25, 71, 113]). Hence, there is significant room for improving signal-graph transformations and for developing WHAR-tailored GNN architectures. Chapter 5 introduces an efficient signal-graph transformation and a lightweight GNN architecture for wearable activity classification. Chapter 6 takes GNN-based WHAR a step further by transforming the complex task of sequential activity prediction into a graph link prediction problem.

# 4 Basics of Foundation Models for Human Activity Recognition

WHAR models are typically trained in an end-to-end manner, where the learning process involves updating their trainable parameters based on predefined input-output pairs (cf. Sections 2.3 and 2.4). This approach requires a sufficient amount of labeled training data to ensure the models generalize well. The more complex the model architectures get, the more data is needed to prevent overfitting. In WHAR, however, labeled datasets are often limited in size, covering only a small number of activities and users due to the time-intensive processes of data collection, preparation, and annotation (cf. Section 2.5).

Foundation Models (FMs) [18] offer an alternative training approach that reduces the need for vast amounts of labeled training data. Instead of traditional end-to-end training, FMs employ a two-phase learning process, enabling the development of larger, more complex, and more adaptable architectures. During the pre-training phase, self-supervised learning updates the model weights to capture underlying structures and patterns in unlabeled data [18]. This process allows the model to develop general representations that can later be refined for specific tasks. The second phase, supervised fine-tuning, customizes the model for downstream applications by either refining the pre-trained weights or appending and training additional layers tailored to the target task [18].

While self-supervised pre-training and supervised fine-tuning are familiar techniques in the WHAR community [104], FMs and their large-scale pre-training are less thoroughly explored for WHAR tasks. Therefore, this chapter defines FMs in

the WHAR domain and explores fundamental concepts such as self-supervised pre-training, cross-dataset evaluation, and task-specific fine-tuning.

## 4.1 Foundation Models

Bommasani et al. define the term Foundation Model (FM) as "any model that is trained on broad data [...] that can be adapted [...] to a wide range of downstream tasks" [18]. However, this definition is very broad. Lawmakers of several countries have clarified the term by emphasizing the extreme size and risk potential [16] or the generality [43] of FMs. Nonetheless, the fundamental characteristics of FMs remain consistent across all definitions: self-supervised pre-training on large, diverse data, and adaptability through task-specific fine-tuning. FMs are either restricted to a single data modality (e.g., text [130] or images [42]) or offer extended generalizability as multi-modal models [18] that fuse data from multiple modalities (e.g., text and images [84]) to gain a deeper understanding of their application domains.

In the WHAR domain, however, data is not available in large volumes, signals differ between users, sensors, and sensor placements, and models are limited to a few tens of thousands of parameters to ensure wearable deployability (cf. Sections 2.5 and 2.6). To address these unique challenges, this work adapts Bommasani's definition to WHAR:

> *Inertial WHAR-FMs are neural networks pre-trained on inertial sensor signals from diverse sensor configurations and activity contexts, designed to be adaptable to WHAR tasks such as activity classification or signal forecasting.*

Figure 4.1 illustrates the workflow of WHAR-FMs. Given the variability in sensing devices, sensors, and activity domains, the homogenization of input signals is crucial for ensuring the generalizability of these models. Moreover, WHAR-FMs must accommodate the resource constraints of wearable devices to enable practical everyday applications.

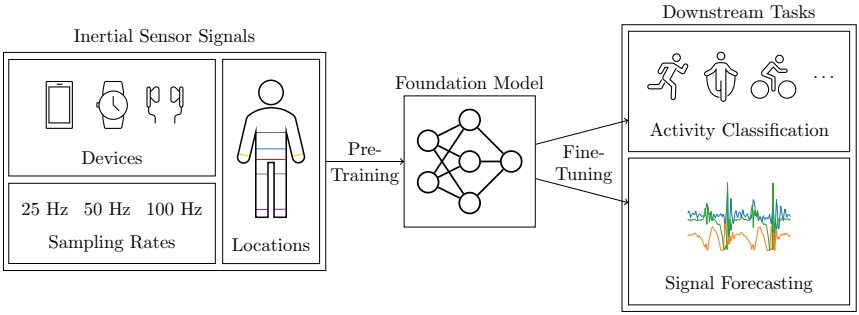



**Figure 4.1:** Workflow of WHAR-FMs - from sensor data acquisition through pre-training to fine-tuning for downstream activity classification or signal forecasting.

## 4.2 Self-Supervised Pre-Training

Self-Supervised Learning (SSL) and pre-training enable FMs to autonomously discover characteristic properties of unlabeled data [18]. Self-supervised pre-training objectives guide the learning process, but do not necessarily resemble the ultimate downstream goal of the models. Therefore, the design and selection of meaningful pre-training objectives is of great importance for pre-trained FMs to achieve generality.

Table 4.1 categorizes previous SSL approaches in WHAR according to their pre-training objectives, model architectures, downstream evaluation tasks, and the existence of cross-dataset validations. These SSL approaches aim to extract characteristic signal abstractions with CNNs, RNNs, or Transformers that facilitate their use in downstream tasks like classification, clustering, or signal forecasting.

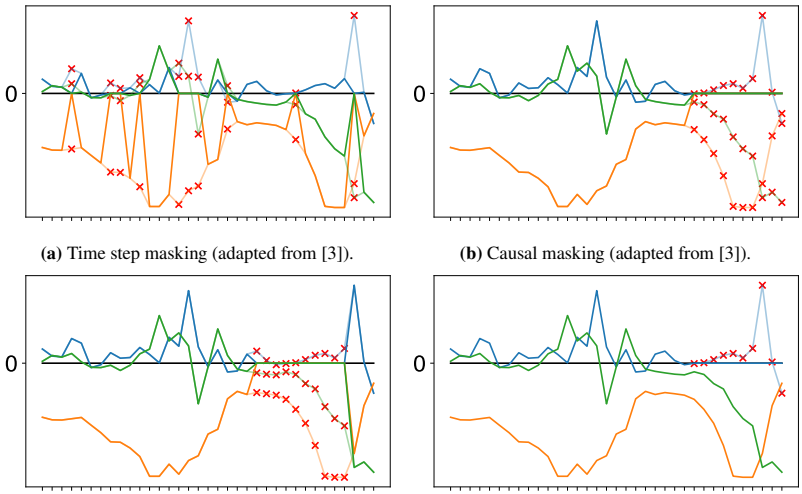
A simple pre-training objective is signal reconstruction, where the goal of a model $z$ is to accurately reconstruct the input signal $s$ [60]. Therefore, the output $z(s)$ of the model should match its input, i.e., $z(s) = s$. For instance, autoencoders use this reconstruction objective to project input signals into a representative latent space while aiming for perfect reconstruction [60].

Masking approaches learn to reconstruct signals with masked values. They involve nullifying specific sensor readings in $s$ to create a masked signal $\hat{s}$, which

**Table 4.1:** Pre-training objectives, network architectures, validated downstream tasks, and the existence of cross-dataset validation in self-supervised WHAR models. Italic references mark this work's publications.

| Pre-Training Objective(s) | Architecture | Downstream Task | Cross-Dataset | References |
|---|---|---|---|---|
| Contrastive Learning | CNN | Classification | No | [38, 77] |
| | CNN, RNN | Classification | No | [161] |
| Masking | CNN | Classification | No | [37] |
| | Transformer | Classification | No | [112] |
| | Transformer | Classification | Yes | [61, 164, 165] |
| Masking + Contrastive Learning | Transformer | Classification | Yes | [69] |
| Causal | CNN | Classification | No | [131] |
| | CNN, RNN | Classification | No | [142] |
| Causal + Contrastive Learning | CNN, RNN | Classification | No | [62] |
| Causal + Reconstruction | RNN | Clustering | No | [6] |
| Transformation Detection | CNN | Classification | No | [76] |
| | CNN | Classification | Yes | [134] |
| Transformation Detection | Transformer, CNN | Classification | Yes | [80] |
| + Contrastive Learning | CNN | Clustering | No | [8] |
| Masking or Causal | Dual-View Mixing | Signal Forecasting | Yes | *[3]* |
| Reconstruction | Multi-Resolution Wavelet-Attention | Classification | Yes | *[4]* |

is then fed into a model $z$ that infers the masked values based on the surrounding signal context [165], i.e., $z(\hat{s}) = s$. Masked Signal Modeling (MSM [3], Figure 4.2a) derives from Masked Language Modeling [39]; it nullifies the sensor readings at $n_{\text{mask}}$ randomly selected time steps. During pre-training, the random selection varies from mini-batch to mini-batch. This way, MSM teaches the model to reconstruct missing values from the broad signal context. Other masking approaches include nullifying the sensor readings of $n_{\text{mask}}$ consecutive time steps (span masking) [164, 165] (Figure 4.2c) or (partially) masking single sensor axes (spatial masking) [112, 131] (Figure 4.2d) and have been applied to input signals (e.g., [61]) or latent representations (e.g., [37]).




**(a)** Time step masking (adapted from [3]).

**(b)** Causal masking (adapted from [3]).

**(c)** Span masking (cf. [165]).

**(d)** Partial axis masking (cf. [131]).

**Figure 4.2:** Overview of different masking approaches.

A special form of masking is Causal Signal Modeling (CSM, Figure 4.2b), which is derived from Causal Language Modeling [130]. CSM learns to forecast future signal values based on previous signals from the same or different sensors by reconstructing the nullified sensor readings at the last $n_{\text{mask}}$ time steps. For instance, Haresamudram et al. [62] forecast multi-modal sensor signals from their

pasts. Rahimi et al. [131] forecast only the z-axis of acceleration based on the past acceleration in z-direction and the whole acceleration signals in x- and y-directions. Spathis et al. [142] use signals from one sensor (e.g., accelerometer) to predict signals from another modality (e.g., heart rate).

Contrastive learning approaches learn embeddings that maximize the similarity between similar (positive) pairs of input signals while minimizing the similarity between dissimilar (negative) pairs. For instance, Jain et al. [77] consider temporally aligned signals from sensors at different body locations as positive samples and temporally misaligned signals from different locations as negative samples. Deldari et al. [38] present a similar approach that identifies temporally aligned signals from different sensor modalities (e.g., sensor types) as positive samples, while temporally distant signals from the same modalities and random batch samples are negative. Wu et al. [161], on the other hand, compare input signals with automatically generated signal representations in a contrastive setup.

Transformation detection approaches apply random augmentations to input signals and learn to recognize the applied transformations in a multi-class learning environment. For instance, Saeed et al. [134] pre-train their model to detect eight signal transformations, including the addition of random noise to input signals, the scaling, rotation, vertical and horizontal flipping, permutation, the time-warping of signals, and the shuffling of the signal channels. Islam et al. [76] apply the same transformations to inertial sensor signals.

Some researchers even combine different pre-training objectives to derive more extensive embeddings [6, 8, 62, 69]. Apart from the pre-training objectives mentioned in Table 4.1, there are also other pre-training approaches in the WHAR area. For instance, some researchers leverage supervised transfer learning (e.g., [68]), others extract signal features with unsupervised autoencoders (e.g., [60]), or apply semi-supervised training objectives (e.g., [44, 45, 129, 147, 170]).

Inspired by transfer learning, cross-dataset validation evaluates the generalizability and transferability of pre-trained models to unseen domains. This method involves pre-training a model on one dataset and assessing the learned embeddings on one or more other datasets from different sources (e.g., [80]). Xu et al. [165]

even merge the unlabeled data from four datasets and evaluate the classification accuracy of the derived representations on each dataset individually. Similarly, Hong et al. [69] pre-train their model on three unlabeled datasets and present the classification accuracy on a fourth dataset. However, with only six publications that employ cross-dataset validation listed in Table 4.1 (plus two from this work), this evaluation strategy is rather uncommon in self-supervised WHAR research.

## 4.3 Task-Specific Fine-Tuning

Task-specific fine-tuning adapts a pre-trained FM to real-world downstream tasks, leveraging its general knowledge while tailoring it to specific applications (cf. Figure 4.1). A common fine-tuning workflow has the following steps [28, 156]:

1. Append additional, yet untrained layers that are specific to the downstream task to the pre-trained model (Figure 4.3).
2. Freeze the weights of the pre-trained layers to preserve their learned representations during fine-tuning.
3. Re-train the model on labeled data using supervised learning. During fine-tuning, only the newly added layers are trainable, allowing the pre-trained knowledge to remain intact while the model adapts to the new task.

After this initial adaptation, an optional refinement step involves (partially) unfreezing and retraining pre-trained layers to further align the model with the new task [28]. This step requires a small learning rate (e.g., 1e-5) to avoid catastrophic forgetting of pre-trained knowledge and to prevent overfitting [28].

An alternative strategy is surgical fine-tuning, which updates only subset of weights in the pre-trained model while others remain fixed [91]. This method is particularly useful for adapting models to changes in input or output distributions [91].

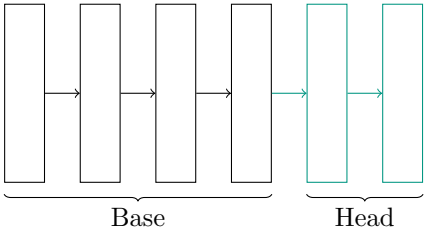



**Figure 4.3:** Task-specific fine-tuning by appending new layers to a pre-trained model. The base contains the frozen pre-trained layers, while the head consists of additional layers tailored to the downstream task.

## 4.4 Summary

Foundation models have the potential to revolutionize the development of novel WHAR approaches by leveraging unlabeled sensor data for pre-training and task-specific labeled data for fine-tuning. However, self-supervised pre-training requires vast amounts of training data and sophisticated pre-training strategies that capture the intrinsic characteristics of human inertial signals and activities. Due to the heterogeneity of WHAR datasets and the lack of standardized pre-training frameworks, large-scale pre-training remains challenging. To address these challenges, Chapter 7 presents a framework for harmonizing WHAR datasets and enabling large-scale, location-invariant pre-training of WHAR-FMs. Chapters 8 and 9 implement WHAR-FMs for signal forecasting and activity classification, respectively. Moreover, these chapters evaluate the significance of data volumes and their impact on recognizing and forecasting signals of different activities.

# Part II

# Innovative Signal Representations with Graph Neural Networks

# 5 Lightweight Activity Classification on the Graph-Level

GNNs for WHAR encode domain knowledge by modeling temporal relationships between sensor readings or spatial relationships between sensors and sensor channels as graphs (cf. Section 3.3). These graph representations enable the structured analysis of complex relationships within sensor signals beyond linear connections in time, up to the detection of compound activities. To ensure effective activity classification, the graphs must remain distinguishable across different activities while respecting the strict resource constraints of wearable devices. Based on *TinyGraphHAR: Enhancing Human Activity Recognition With Graph Neural Networks* [1], this chapter presents a signal-graph transformation that captures the temporal context in sensor signals, coupled with a lightweight GNN architecture for efficient graph-level activity recognition. This approach, which provides a novel perspective on interpreting sensor signals as graphs, achieves state-of-the-art classification performance while using fewer network parameters than comparable models.

## 5.1 Signal-Graph Transformation

WHAR applications typically process fixed-length signals windows $s$ (cf. Section 2.1). Building on this processing paradigm, this work converts signal windows into graphs using the following signal-graph transformation:

1. Apply a 1-level DWT with the Haar Wavelet to a signal $s \in \mathbb{R}^{2w \times f}$, but extract only the low-frequency approximation coefficients $cA \in \mathbb{R}^{w \times f}$.
2. Construct a graph from $cA$ that captures temporal relationships between the coefficients.

Step 1 smoothes $s$ by extracting its low-frequency approximation coefficients and omitting high-frequency components. Provided that the input signals were recorded at a sufficiently high sampling rate, extracting $cA$ is fine for WHAR tasks since human activities predominantly occur at a maximum frequency of 15 Hz [33]. Furthermore, the approximation coefficients have only half as many time steps as $s$, which helps keeping the graphs small.

Step 2 converts the approximation coefficients into graph vertices $v_i \in V$. Each $v_i$ contains a feature vector $x_i \in X$ that holds the $f$-dimensional $cA_i$ at time $t_i$ with $i \in \{1, \ldots, w\}$. Additionally, step 2 links the vertices $v_i$ to all vertices $v_j$ in their neighborhoods

$$N_k(v_i) = \{v_j \mid \max(1, i-k) \leq j \leq \min(i+k, w)\},\ k \in \mathbb{N}^+. \tag{5.1}$$

Self-loops $v_i \in N(v_i)$ ensure that GNNs include the features of the source vertices in the computation of their updated representations (cf. Equation 3.2). In edge cases where $i - k < 1$ or $i + k > w$, the neighborhoods get truncated and thus, are smaller than those of the inner vertices. The resulting signal graph $G$ reflects the time course of the input signal as $N_k$ defines the symmetric neighborhood around $v_i$ and includes all intermediate vertices between $v_{i-k}$ and $v_{i+k}$.

Figure 5.1 illustrates the signal-graph transformation with $N_2$ for a short signal window with $w = 5$ time steps and $f = 3$ distinct features.

To ensure high expressiveness and capacity of the graphs while keeping the number of edges small and not overloading the graphs with complex edge relationships that are difficult to interpret, this work compares neighborhoods up to $k = 3$:

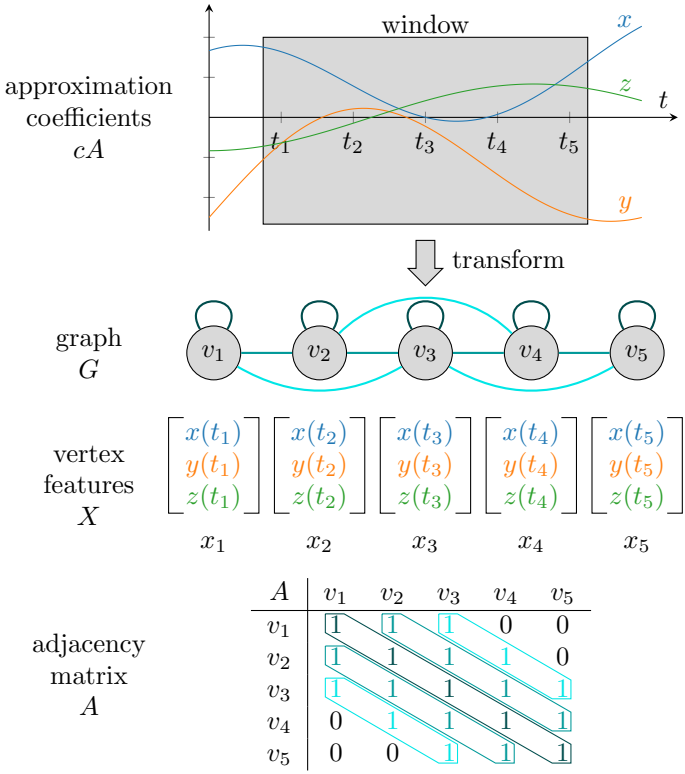



**Figure 5.1:** Signal-graph transformation for lightweight WHAR with GNNs (adapted from [1]). This example transforms the approximation coefficients $cA$ of a three-dimensional signal into a graph $G$ with $w = 5$ vertices. Each vertex $v_i \in V$ represents the features $x_i$ at time $t_i$ with $i \in \{1, \ldots, w\}$. The edges connect the vertices according to Equation 5.1.

$$N_1(v_i) = \{v_i, v_{i-1}, v_{i+1}\}, \tag{5.2}$$
$$N_2(v_i) = \{v_i, v_{i-1}, v_{i+1}, v_{i-2}, v_{i+2}\}, \tag{5.3}$$
$$N_3(v_i) = \{v_i, v_{i-1}, v_{i+1}, v_{i-2}, v_{i+2}, v_{i-3}, v_{i+3}\}. \tag{5.4}$$

For $k = 1$, only the main diagonal (self-loops) of the square adjacency matrix $A^{w \times w}$ and its first upper and lower secondary diagonals (direct temporal neighbors of $v_i$) are filled with ones; the other cells contain zeros. $k = 2$ additionally fills the second-order upper and lower secondary diagonals (indirect neighbors of $v_i$) with ones; $k = 3$ the third-order diagonals. After applying Step 1, the $cA$ windows are usually small with $32 \leq w \leq 256$ time steps. In turn, the resulting graphs are also small with $|V| = w$ vertices and highly sparse adjacency matrices. Furthermore, Equation 5.1 ensures that $A$ remains identical for all signal graphs as long as $k$ and $w$ do not change. Hence, this signal-graph transformation ensures the reusability of $A$ with low resource requirements through the one-time generation and storage of the adjacency matrix.

## 5.2 Model Design

Figure 5.2 presents a lightweight, graph-level GNN architecture for classifying signal graphs $G = (X, A)$. This model analyzes signal graphs in three steps:

1. Graph Processing,
2. Feature Extraction,
3. Downstream Classification.

The Graph Processing part analyzes a signal graph with a GGC and a GAT layer (cf. Section 3.2). The GGC expands the feature dimension $f$ of the vertex features $X \in \mathbb{R}^{w \times f}$ to $c$ channels and captures temporal relationships between the vertices through a GRU operation. It outputs an updated feature matrix $X \in \mathbb{R}^{w \times c}$. The

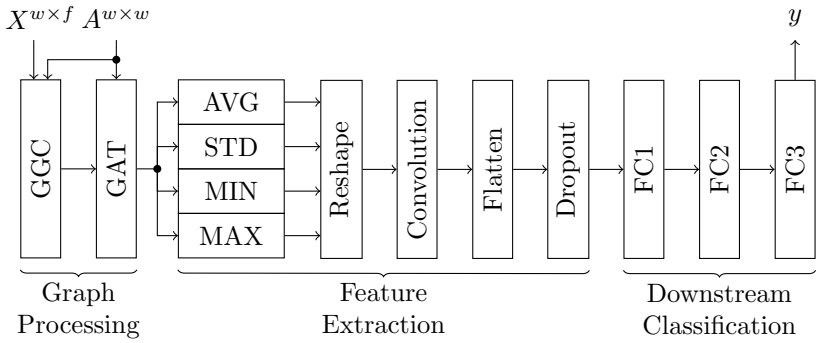



**Figure 5.2:** The lightweight GNN architecture for signal graph classification (adapted from [1]). The Graph Processing block leverages structural information of the input graphs $G = (X, A)$ to update the vertex features. The Feature Extraction block breaks the graph structure and incorporates global statistics to learn representative vectors from the vertex features only. The Downstream Classification block detects the performed activity $y \in Y$.

GAT layer refines these features by dynamically attending to neighboring vertices using $n_h$ attention heads. Each head learns unique attention weights that focus on different aspects of the graph's structure. As a result, the GAT layer transforms the vertex features $X$ from shape $(w, c)$ to $(w, n_h c)$.

During Feature Extraction, the model extracts patterns from $X$ that are most relevant for downstream activity classification. This step breaks the graph structure by aggregating the updated vertex features using $n_p = 4$ global pooling operations AVG, STD, MIN, and MAX (cf. Equations 3.9 to 3.12). This results in four $n_h c$-sized vectors that emphasize key properties of the updated vertices. The reshape operation transforms the pooled features from $(n_p, n_h c)$ to $(n_p, c, n_h)$ to be compatible with the subsequent 2D convolution. This convolution applies a kernel of size $(n_p, 1)$ to aggregate information along the $n_p$ dimension of the reshaped features. This operation effectively reduces the output shape to $(1, c, n_h)$ while preserving the channel and attention head dimension. Flattening prepares the output for downstream classification; dropout prevents overfitting and improves generalization by randomly deactivating $d\,\%$ of the neurons during training. In the end, the Feature Extraction block outputs $(n_h c, )$-shaped vectors.

The Downstream Classification block predicts the most probable activity $y \in Y$ with three subsequent fully-connected layers (FC1 to FC3). These layers gradually reduce the hidden dimensions from $(n_h c, )$ to the number of classifiable activities $|Y|$. FC1 and FC2 output $(\lfloor(|\text{previous layer}| + |Y|)/2\rfloor, )$-shaped vectors, while FC3 has $|Y|$ output neurons. Here, |previous layer| is the number of neurons in the preceding layer. For FC1, the previous layer is the dropout layer which outputs $(n_h c, )$-shaped representations; for FC2, the previous layer is FC1.

## 5.3 Implementation and Evaluation Details

The proposed signal-graph transformation and GNN architecture are highly configurable. Selecting appropriate hyperparameters is essential for achieving optimal model performance and ensuring wearable applicability. To assess the reliability and effectiveness of the lightweight GNN for human activity classification, this work trains it on four benchmark datasets. This section provides a comprehensive overview of the implementation and evaluation process by detailing the hyperparameters (Section 5.3.1), outlining the data preparation steps (Section 5.3.2), and describing the training setup (Section 5.3.3).

### 5.3.1 Hyperparameters

The GNN contains several customizable hyperparameters, such as the number of hidden channels or attention heads, that impact classification performance and model size. This work uses the values from Table 5.1 to ensure high classification performance while keeping the classifiers small with 2,470 to 8,080 trainable parameters only.

Moreover, the layers apply specific activation functions that enhance the extraction of meaningful signal patterns. The output layer FC3 uses the sigmoid activation for binary classification tasks and the softmax activation for multi-class classification. FC2 uses $\tanh$ activations. All other layers use LeakyReLU with $\alpha = 0.3$.

**Table 5.1:** Model configuration.

| Parameter | Value(s) |
|---|---|
| Channels $c$ | $\{12, 20\}$ |
| Attention Heads $n_h$ | 3 |
| Pooling Operations $n_p$ | 4 |
| Dropout Factor $d$ | $20\,\%$ |

The graphs also have configurable hyperparameters that define their structure (cf. Table 5.2). The window size and the number of features depend on the signals and sensors provided by the training datasets. The ideal neighborhood size must be determined experimentally. This work compares small neighborhoods up to $k = 3$.

**Table 5.2:** Graph configuration.

| Parameter | Value(s) |
|---|---|
| Window Size $w$ | $\mathbb{N}^+$ |
| Features $f$ | $\mathbb{N}^+$ |
| Neighborhood $N_k$ | $\{N_1, N_2, N_3\}$ |

### 5.3.2 Data Preparation

This work evaluates the GNN on four benchmark datasets: KU-HAR [141], PAMAP2 [133], UCI-HAR [10], and Daphnet [12]. These datasets represent a wide variety of daily living activities, sensing setups, and demographic structures (cf. Table 2.2). KU-HAR provides non-overlapping $3\,\mathrm{s}$ signal windows. The other datasets provide continuous sensor signals. This work splits those signals into windows of $5.12\,\mathrm{s}$ for PAMAP2, $2.56\,\mathrm{s}$ for UCI-HAR, and $1\,\mathrm{s}$ for Daphnet with a $50\,\%$ overlap. These durations are given by the respective publications.

Only for Daphnet, this work sticks with the window duration from [177] to keep the GNN comparable with state-of-the-art classifiers.

The signal-graph transformation (cf. Section 5.1) constructs graphs from signal windows. Since window lengths and sampling rates differ between datasets, the resulting graphs have different numbers of vertices, ranging from $|V| = w = 32$ for Daphnet to 256 for PAMAP2. Additionally, the datasets provide different types and quantities of sensors (cf. Section 2.5). Hence, the number of features $f$ per vertex also varies. Table 5.3 summarizes the graph dimensions per dataset along with the durations of the respective signal windows in seconds.

**Table 5.3:** Vertex feature dimensions $X \in \mathbb{R}^{w \times f}$ and window durations per dataset.

| Dataset | $w$ | $f$ | Duration |
|---|---|---|---|
| Daphnet | 32 | 9 | 1 s |
| UCI-HAR | 64 | 6 | 2.56 s |
| KU-HAR | 150 | 6 | 3 s |
| PAMAP2 | 256 | 18 | 5.12 s |

Table 5.4 lists the space requirements in kB for storing the GNNs as TensorFlow Lite models with 32 bit weights per dataset and channel count $c$. The datasets differ in their feature counts $f$ (cf. Table 5.3) and the number of activities $|Y|$ (cf. Table 2.2). The storage requirements increase with increasing $c$, $f$, and $|Y|$. The 12-channel models are approximately 30 % smaller than their 20-channel counterparts. Nevertheless, the 20-channels models still require less than 60 kB of storage space, which is acceptable for modern wearables. These sizes confirm the lightness of the proposed GNN. Weight pruning can reduce the space requirements even further while keeping the recognition rates high [59].

Table 5.5 compares the space requirements in kB for storing the graph's adjacency matrices from the four datasets. The table distinguishes between sparse matrix representations in the COO format (cf. Section 3.1) and binary dense matrices. The sparse sizes include an overhead of 0.928 kB that defines the COO format.

**Table 5.4:** Model sizes per dataset and channel count $c$ in kB.

| Dataset | $c = 12$ | $c = 20$ |
|---|---|---|
| Daphnet | 35.2 | 51.0 |
| UCI-HAR | 37.0 | 53.4 |
| KU-HAR | 42.4 | 60.3 |
| PAMAP2 | - | 59.1 |

Since COO matrices store the positions and values of the occupied matrix cells, the storage requirements increase with increasing neighborhood sizes. In contrast, the storage requirements for the dense matrices remain constant regardless of the neighborhood configuration as each cell is saved individually. Despite the COO overhead, the sparse format requires only up to $2.095\,\mathrm{kB}$ to store the largest adjacency matrix with the most complex neighborhood $N_3$. The largest COO matrix requires similar storage space as the smallest dense matrix. However, since the storage requirements of the dense matrices increase quadratically, the sparse representation is clearly the preferred choice for mobile applications. Compared to the model sizes, the memory requirements for the one-time storage of the sparse adjacency matrices is negligible. Thus, the sparse adjacency matrices do not hinder the wearable applicability of the GNN.

The KU-HAR dataset contains labeled sensor signals, while Daphnet, UCI-HAR, and PAMAP2 also provide mappings between sensor signals and users. To evaluate the GNN, this work randomly splits KU-HAR ten times into training ($70\,\%$), validation ($15\,\%$), and test ($15\,\%$) data and reports the average classification performance on the test sets. For the other datasets, which include user-related information, this work employs Leave-One-Subject-Out (LOSO) cross-validation. In LOSO, signals from one subject (i.e., a user) serve as the test set, while signals from the remaining users are randomly split into training and validation sets. Adhering KU-HAR's 70/15/15 rule, this work uses $70/85$ of the remaining signals for training and $15/85$ for validation. LOSO repeats this splitting operation so that each user is tested once and returns the average classification performance across all test users.

**Table 5.5:** Comparison of space requirements in kB for storing sparse adjacency matrices in COO format with different neighborhoods versus dense matrices. Dense matrices require the same space regardless of neighborhood size, as they store the entire matrix.

| | Sparse | | | Dense |
|---|---|---|---|---|
| Dataset | $N_1$ | $N_2$ | $N_3$ | $\{N_1, N_2, N_3\}$ |
| Daphnet | 1.095 | 1.117 | 1.125 | 2.080 |
| UCI-HAR | 1.216 | 1.249 | 1.263 | 8.256 |
| KU-HAR | 1.498 | 1.571 | 1.603 | 45.150 |
| PAMAP2 | 1.912 | 2.045 | 2.095 | 131.328 |
| Overhead | | 0.928 | | 0 |

### 5.3.3 Training Setup

The cross-entropy loss guides the learning process [29, 53]. The weights are updated with the Adam optimizer [85], learning rate 0.001, and gradient norm clipping to prevent exploding gradients during training [125]. Furthermore, this work applies early stopping with patience 35 to terminate the training when the validation loss stagnates, thereby reducing the risk of overfitting.

During training, the GNN processes small batches of 32 graphs. To account for statistical instabilities, this work trains the models five times per split using different random seeds $\in \{0, 1, 2, 3, 4\}$.

## 5.4 Performance Analysis

This work analyzes the classification accuracies and macro F1-scores (cf. Equations 2.10 and 2.11) of the lightweight GNN on four benchmark datasets: KU-HAR, PAMAP2, UCI-HAR, and Daphnet. Section 5.4.1 analyzes the GNN's average macro-F1 scores and accuracies for $c \in \{12, 20\}$ channels and $N_k \in \{N_1, N_2, N_3\}$ per dataset. Section 5.4.2 compares the best performing configurations to the published results of state-of-the-art classifiers that also aim

to reduce resource consumption by keeping the number of network parameters small. However, since KU-HAR is a relatively new dataset, the number of papers presenting resource-efficient solutions for it is still small. Despite that, this evaluation includes KU-HAR due to its comprehensive coverage of 18 daily activities performed by 90 individuals (cf. Table 2.2). To further assess the lightness of the proposed GNN, Section 5.4.3 analyzes its computational complexity and inference time on a smartwatch.

### 5.4.1 Comparison of Model and Graph Configurations

#### KU-HAR.

For KU-HAR, the GNN achieves accuracies of up to 0.955 and macro F1-scores of up to 0.953 (cf. Table 5.6). These high recognition rates, coupled with small standard deviations, demonstrate the effectiveness and consistency of the proposed approach. Both channel configurations perform best with the $N_2$ neighborhood, though the differences across neighborhood sizes are relatively small. The larger 20-channel models outperform the smaller 12-channel models by up to 0.017 points, suggesting that increased model capacity leads to better predictions. Additionally, the similarity between accuracies and macro F1-scores indicates a well-balanced performance across activities.

Figure 5.3 presents the confusion matrix for the best GNN configuration (20 channels, $N_2$). The numbers are averaged across all seeds and test splits. The confusion matrix confirms the high scores from Table 5.6 as only its diagonal entries have high values while the other cells are close to 0. The GNN distinguishes dynamic movements such as walk, run, and jump almost perfectly and even masters the distinction between similar movements with different nuances, such as walk, walk-backward, and walk-circle. In contrast, the distinction between static activities, such as stand, sit or lay, is more difficult. However, the frequency of misclassifications of static activities is only in the single-digit range, with a maximum number of twelve false classifications when distinguishing between stand and sit.

**Table 5.6:** The average accuracies and macro F1-scores for different channels $c$ and neighborhoods $N_k$ on KU-HAR (adapted from [1]). $\mu$: average scores (**best**), $\sigma$: standard deviations.

| | | Accuracy | | Macro F1 | |
|---|---|---|---|---|---|
| $c$ | $N_k$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| 12 | $N_1$ | 0.939 | 0.012 | 0.935 | 0.022 |
| 12 | $N_2$ | **0.941** | 0.016 | **0.936** | 0.010 |
| 12 | $N_3$ | 0.937 | 0.020 | 0.935 | 0.024 |
| 20 | $N_1$ | 0.952 | 0.014 | 0.950 | 0.024 |
| 20 | $N_2$ | **0.955** | 0.022 | **0.953** | 0.016 |
| 20 | $N_3$ | 0.953 | 0.013 | 0.952 | 0.012 |

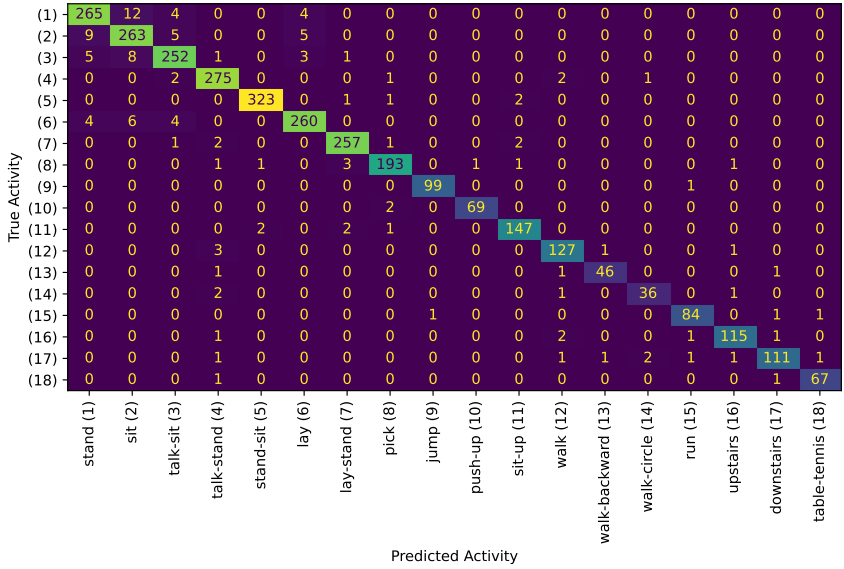



**Figure 5.3:** The average confusion matrix of the best performing GNN configuration on KU-HAR (20 channels, $N_2$).

### PAMAP2.

Table 5.7 presents the GNN's average performance on PAMAP2. Since this dataset provides $f = 18$ features per time step (cf. Table 5.3) and the GGC layer requires the number of channels $c$ to be greater than $f$ (cf. Section 3.2), this work only considers $c = 20$ for PAMAP2 and omits the smaller $c = 12$ configuration. The GNN achieves its best performance for graphs with the $N_1$ neighborhood. The performance across different neighborhood sizes varies slightly, with the macro F1-scores differing by up to 0.028 and accuracies by up to 0.014 points. The standard deviations range from 0.103 to 0.368, indicating less consistency compared to KU-HAR. This variability can be attributed to inter-user differences: users perform activities differently, not all users completed every activity (e.g., the ninth user only performed rope jumping), and the duration of the activities also varies.

**Table 5.7:** The average accuracies and macro F1-scores for different channels $c$ and neighborhoods $N_k$ on PAMAP2 (adapted from [1]). $\mu$: average scores (**best**), $\sigma$: standard deviations.

| | | Accuracy | | Macro F1 | |
|---|---|---|---|---|---|
| $c$ | $N_k$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| 20 | $N_1$ | **0.800** | 0.103 | **0.756** | 0.331 |
| 20 | $N_2$ | 0.790 | 0.318 | 0.728 | 0.368 |
| 20 | $N_3$ | 0.786 | 0.182 | 0.732 | 0.366 |

Figure 5.4 presents the confusion matrix for the best GNN configuration (20 channels, $N_1$). The numbers are averaged across all seeds and all LOSO splits. The confusion matrix confirms the imbalance in PAMAP2's activity distribution, as the average number of graphs per activity in the test sets ranges from 19 for rope jumping to 100 for ironing. In general, the matrix aligns well with the scores from Table 5.7, with the largest values appearing along its main diagonal, while most other cells remain close to 0. However, the GNN confuses some static movements, such as sit and stand, and also struggles to distinguish similar dynamic movements, such as downstairs, walk, and nordic walking.

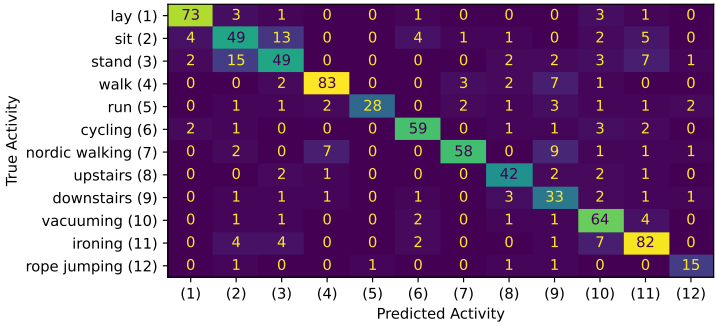



**Figure 5.4:** The average confusion matrix of the best performing GNN configuration on PAMAP2 (20 channels, $N_1$).

### UCI-HAR.

For UCI-HAR, the GNN achieves accuracies of up to 0.9 and macro F1-scores of up to 0.902 (cf. Table 5.8). Across both channel configurations, $N_1$ produces the best results. With improvements of 0.007 points in macro F1-score and 0.008 points in accuracy, $c = 20$ performs only marginally better than $c = 12$. The differences between the best and worst $N_k$ for each channel configuration are below 0.006 points; the standard deviation is also quite low at a maximum of 0.061. This demonstrates the GNN's consistent classification performance across different graph configurations and random seeds.

Figure 5.5 presents the confusion matrix for the best GNN configuration (20 channels, $N_1$). The numbers are averaged across all seeds and all LOSO splits. The high values along the main diagonal combined with the small numbers in the off-diagonal cells confirm the high accuracy and macro F1-score from Table 5.8. The GNN rarely confuses dynamic activities (walk, upstairs, downstairs), whereas the differentiation of static activities (sit, stand, lay) is more challenging. Notably, the GNN never confuses dynamic activities with static activities. This observation underscores its strong feature extraction capabilities.

**Table 5.8:** The average accuracies and macro F1-scores for different channels $c$ and neighborhoods $N_k$ on UCI-HAR (adapted from [1]). $\mu$: average scores (**best**), $\sigma$: standard deviations.

| | | Accuracy | | Macro F1 | |
|---|---|---|---|---|---|
| $c$ | $N_k$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| 12 | $N_1$ | **0.893** | 0.040 | **0.894** | 0.049 |
| 12 | $N_2$ | 0.891 | 0.053 | 0.893 | 0.054 |
| 12 | $N_3$ | 0.887 | 0.028 | 0.890 | 0.030 |
| 20 | $N_1$ | **0.900** | 0.039 | **0.902** | 0.041 |
| 20 | $N_2$ | 0.896 | 0.043 | 0.898 | 0.048 |
| 20 | $N_3$ | 0.895 | 0.055 | 0.897 | 0.061 |

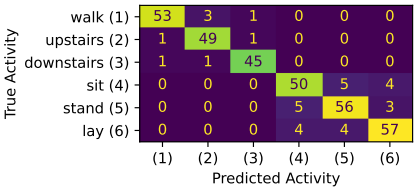



**Figure 5.5:** The average confusion matrix of the best performing GNN configuration on UCI-HAR (20 channels, $N_1$).

### Daphnet.

For Daphnet, the GNN consistently achieves high accuracies close to 0.89 across all configurations (cf. Table 5.9). However, the average accuracy exceeds the average macro F1-score by 0.311 points, indicating that the GNN struggles to recognize one of the two classes in Daphnet's imbalanced class distribution.

The GNN with 12 channels performs similarly to the GNN with 20 channels. In both cases, $N_1$ achieves a notably higher macro F1-score than $N_2$ and $N_3$; particularly for the 20-channel configuration. In contrast, $N_2$ yields slightly higher accuracies than the other neighborhood configurations. Nevertheless, this work considers $N_1$ as the best configuration for Daphnet, as the accuracy differences

across different neighborhoods are minimal and the macro F1-scores provide a more balanced evaluation of class-specific performance.

**Table 5.9:** The average accuracies and macro F1-scores for different channels $c$ and neighborhoods $N_k$ on Daphnet (adapted from [1]). $\mu$: average scores (**best**), $\sigma$: standard deviations.

| | | Accuracy | | Macro F1 | |
|---|---|---|---|---|---|
| $c$ | $N_k$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| 12 | $N_1$ | 0.885 | 0.034 | **0.583** | 0.234 |
| 12 | $N_2$ | **0.887** | 0.058 | 0.577 | 0.490 |
| 12 | $N_3$ | 0.886 | 0.064 | 0.570 | 0.175 |
| 20 | $N_1$ | 0.887 | 0.052 | **0.598** | 0.232 |
| 20 | $N_2$ | **0.888** | 0.024 | 0.572 | 0.202 |
| 20 | $N_3$ | **0.888** | 0.025 | 0.554 | 0.183 |

Figure 5.6 presents the confusion matrix for the best GNN configuration (20 channels, $N_1$). The numbers are averaged across all seeds and all LOSO splits. Daphnet includes two classes freeze and no freeze, which reflect the presence or absence of freezing of gait in Parkinson's patients during walking [12]. On average, the GNN correctly classifies 83 of the 345 graphs that represent freezing of gait and 3,076 of the 3,217 graphs from healthy activities. These results reinforce the findings from Table 5.9, as they highlight the GNN's difficulty in identifying the underrepresented freeze class, which has roughly nine times fewer test samples than the no freeze class.

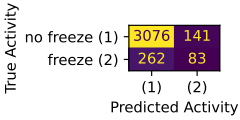



**Figure 5.6:** The average confusion matrix of the best performing GNN configuration on Daphnet (20 channels, $N_1$).

### 5.4.2 Comparison with State-of-the-Art Classifiers

#### KU-HAR.

Figure 5.7 compares the parameter counts, the F1-scores, and the accuracies of the best GNNs per channel configuration with literature. The CNN approaches with arithmetic optimization [33] or circulant matrices [140] present their best results for a certain test set; the Transformer approach [41] includes extensive data augmentation that boosts the model performance.

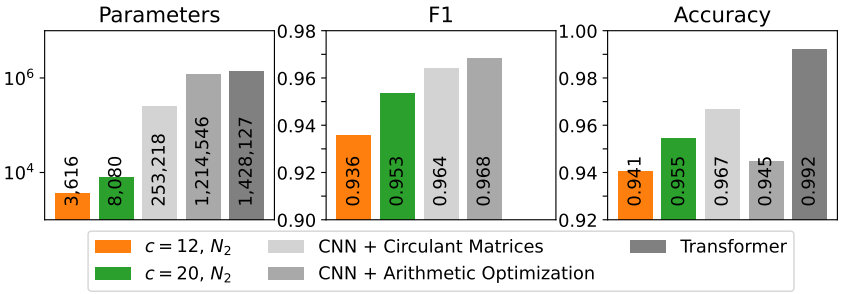



**Figure 5.7:** Comparison of network parameters, F1-scores, and accuracies between the best graph and GNN configurations and state-of-the-art classifiers on KU-HAR (adapted from [1]).

The GNNs are two to three orders of magnitude smaller than the comparison models. Despite that, the 20-channel GNN achieves an F1-score that can keep up with both CNN approaches. The CNN with arithmetic optimization outperforms this GNN by only 0.015 points, while the gap between the GNN and the CNN with circulant matrices is even smaller. In terms of accuracy, the 20-channel GNN surpasses the CNN with arithmetic optimization by 0.01 points and even the 12-channel GNN scratches the score of the arithmetic-optimized CNN. However, the large Transformer model, with its 1.4 million parameters and data augmentation, exceeds the GNNs in accuracy. Nonetheless, these results highlight the effectiveness of GNNs for WHAR tasks.

### PAMAP2.

Figure 5.8 compares the parameter counts, the F1-scores, and the accuracies of the 20-channel GNN with DeepConvLSTM [17, 122] and TinyHAR [177]; two state-of-the-art activity classifiers. The scores of the comparison approaches are taken from [177], which also presents a smaller version of DeepConvLSTM called DeepConvLSTM_0.25 that matches TinyHAR's parameter count. Unfortunately, [177] does not report accuracy scores. Hence, this work only compares the macro F1-scores of the classifiers.

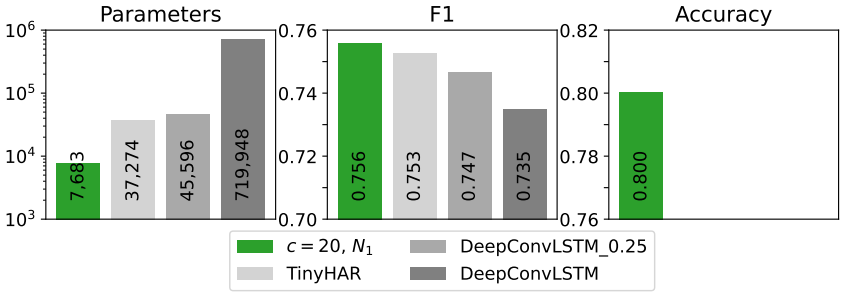



**Figure 5.8:** Comparison of network parameters, F1-scores, and accuracies between the best graph and GNN configurations and state-of-the-art classifiers on PAMAP2 (adapted from [1]).

With a modest improvement of 0.003 points, the GNN slightly outperforms TinyHAR on PAMAP2, but demonstrates greater improvements of up to 0.021 points over the DeepConvLSTM models. Furthermore, the GNN uses only 20.6 % of TinyHAR's parameters, which is the smallest comparison model. These results further confirm the applicability of GNNs to wearable activity classification.

### UCI-HAR.

Figure 5.9 compares the parameter counts, the F1-scores, and the accuracies of the best GNNs per channel configuration with literature. In contrast to this work, which employs a fine-grained LOSO cross-validation across all test users, the literature approaches predsim [148], LSTM-CNN [162], and CNN (Model B) [162] evaluate their models on a fixed selection of test users. Consequently,

the comparison models do not fully capture the variability in individual movement styles, which can significantly influence performance metrics. For instance, the average macro F1-score of the 20-channel GNN is 0.192 points higher than its score for the 9th UCI-HAR user, whereas the score for the 24th user exceeds the GNN's average by 0.092 points.

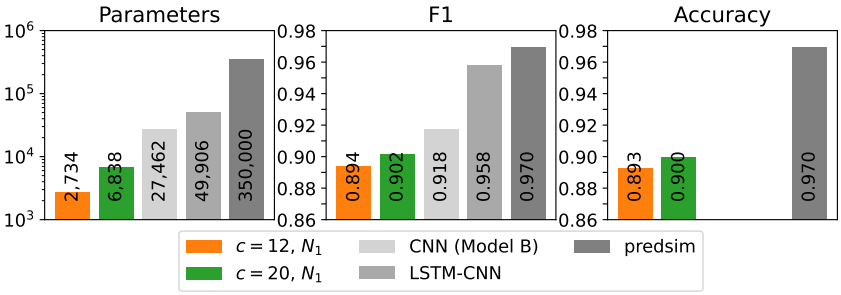



**Figure 5.9:** Comparison of network parameters, F1-scores, and accuracies between the best graph and GNN configurations and state-of-the-art classifiers on UCI-HAR (adapted from [1]).

Overall, the GNNs are up to two orders of magnitude smaller than the comparison models. Despite being four to ten times smaller, the GNNs perform similarly to CNN (Model B), which is the smallest comparison model. LSTM-CNN and predsim outperform the GNNs by 0.056 to 0.068 points, but require at least seven times more parameters to achieve these gains. This comparison underscores the potential of GNNs for resource-constrained WHAR.

### Daphnet.

Figure 5.10 compares the parameter counts, the F1-scores, and the accuracies of the best GNNs per channel configuration with TinyHAR, DeepConvLSTM_0.25, and DeepConvLSTM. Again, this work compares the macro F1-scores of the GNNs with the results from [177], which does not report accuracy scores.

The GNNs outperform DeepConvLSTM by 0.083 points for $c = 20$ and by 0.068 points for $c = 12$, despite requiring less than $1.5\,\%$ of DeepConvLSTM's trainable parameters. The 20-channel GNN surpasses DeepConvLSTM_0.25 by

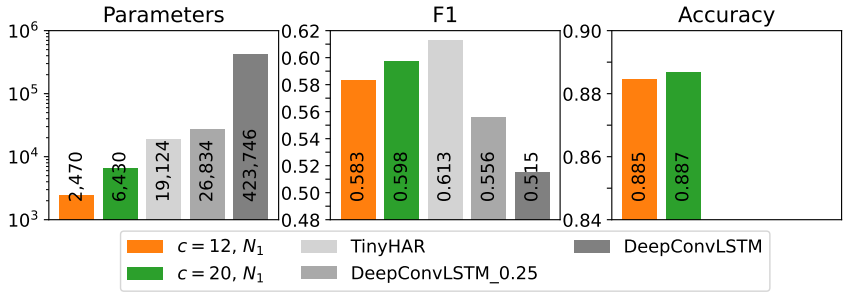



**Figure 5.10:** Comparison of network parameters, F1-scores, and accuracies between the best graph and GNN configurations and state-of-the-art classifiers on Daphnet (adapted from [1]).

0.042 points while being four times smaller. The 12-channel model, although even smaller, still outperforms DeepConvLSTM_0.25 by 0.027 points. TinyHAR outperforms both GNNs by 0.015 points for $c = 20$ and 0.03 points for $c = 12$ but is approximately three times larger than the 20-channel GNN. These results highlight the potential of GNNs for further advancements in lightweight gait freeze detection.

### 5.4.3 Efficiency Analysis

To evaluate the lightness and real-world applicability of the proposed GNNs, this work measures their inference times on a Fossil Gen 5 Carlyle HR smartwatch. This smartwatch features the Qualcomm Snapdragon Wear 3100 platform with four 1.2 GHz ARM Cortex-A7 cores and 1 GB of LPDDR3 RAM clocked at 400 MHz. Figure 5.11 shows the average inference times of the best GNNs per channel configuration and dataset in milliseconds over 1000 runs. To further analyze the computational complexity of the models, this figure also presents the estimated MMACs (Millions of Multiply-Accumulate Operations) of the models after TensorFlow Lite conversion.

The average inference times and MMACs vary based on the graph configurations and the number of output neurons, which correspond to the classifiable activities.

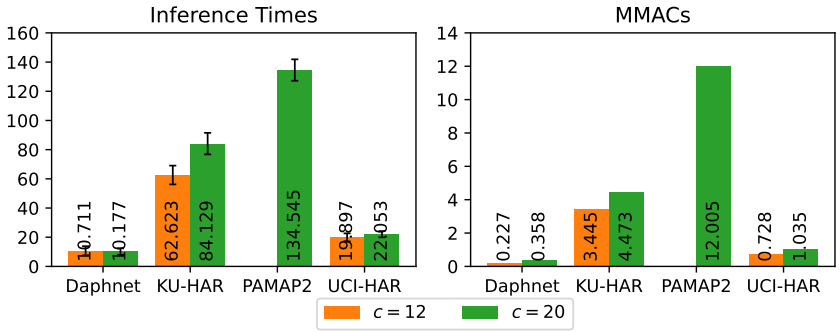



**Figure 5.11:** The average inference times in ms over 1000 runs and MMACs of the best GNN configurations with 12 and 20 channels on UCI-HAR, PAMAP2, KU-HAR, and Daphnet after TensorFlow Lite conversion.

Datasets with larger input graphs, such as KU-HAR and PAMAP2 (cf. Table 5.3), have longer inference times and require more MMACs than datasets with smaller feature matrices and fewer labels. As expected, the larger 20-channel models require more time to process graphs than the 12-channel models. This effect increases with the size of the input graphs and is most noticeable for KU-HAR. For Daphnet with its small $32 \times 9$ feature matrices, the differences between the model configurations are negligible. Overall, the inference times per graph represent only a small fraction of the durations of their respective signal windows (cf. Table 5.3), demonstrating the lightness and real-world applicability of the GNNs, even on a smartwatch with limited memory and processing power.

## 5.5 Summary

This chapter presented a novel signal-graph transformation that constructs structured graphs from multivariate sensor signals. This transformation converts each time step in a signal window into a graph vertex and connects them using temporal edges that capture relationships between adjacent and distant sensor readings. By specifying vertex neighborhood sizes, the graphs can be tailored to different sensing and activity domains. For fixed window and neighborhood sizes, the

transformation yields graphs with constant, reusable adjacency matrices that are highly sparse, as each vertex typically has a small neighborhood relative to the overall graph size. This sparsity and reusability enable efficient storage of the matrices, making the approach well-suited for deployment on wearable devices.

Additionally, this chapter introduced a shallow GNN architecture for WHAR that is designed to run efficiently on devices with limited computational resources. Unlike traditional activity classifiers that treat sensor data as flat time-series, this GNN analyzes the interactions between sensor readings and sensor placements in a more structured way. It achieves high classification rates across four benchmark datasets with varying activity types, users, and sensor setups, while maintaining fast inference times on a smartwatch with limited processing power. These results demonstrate the potential of graph-based activity classifiers as lightweight, efficient alternatives to state-of-the-art approaches.

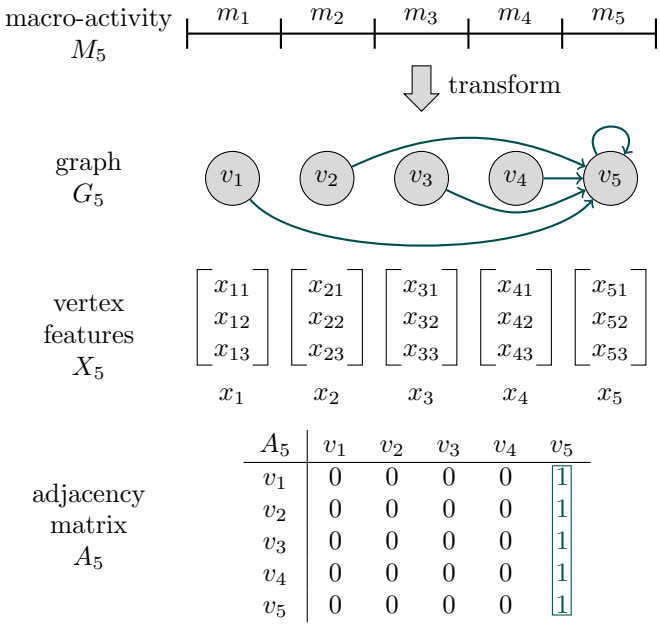
# 6 Sequential Activity Prediction on the Edge-Level

Beyond the classification of short signal windows, recognizing long-running activity sequences composed of consecutive micro-activities poses a key challenge in ubiquitous and pervasive WHAR (cf. Section 2.6). To determine whether consecutive signal windows belong to a common macro-activity (cf. Section 2.1), this work frames sequential activity prediction as a graph link prediction problem (cf. Section 3.2). The proposed solution is based on the activity-graph transformation and the GNN architecture introduced in *Detection and Validation of Macro-Activities in Human Inertial Signals Using Graph Link Prediction* [2]. This approach analyzes two aspects of micro-activity sequences: whether the latest micro-activity logically follows the preceding micro-activity sequence (link prediction) and whether the current sequence constitutes a complete macro-activity (graph validation). Results on two benchmark datasets show that the proposed graph link prediction approach is both effective and well-suited for recognizing sequential activities in wearable sensor signals.

## 6.1 Activity-Graph Transformation

According to Section 2.1, sequences of human micro-activities $m_t$ with $t \in \{1, \ldots, T\}$ form macro-activities $M_T$. The analysis of such micro-activity sequences with GNNs requires a meaningful activity-graph transformation that reflects the relationships and the core properties of each micro-activity in the

sequence. This work builds a micro-activity graph $G_T$ from a micro-activity sequence by converting each $m_t \in M_T$ into a graph vertex $v_t \in V_T$ and connecting each $v_t$ to the last vertex $v_T$ in time. The resulting graph $G_T$ has a highly sparse adjacency matrix $A_T$ which contains ones only in the $T$-th column, while all other cells remain zero. Figure 6.1 illustrates the proposed activity-graph transformation for a short macro-activity $M_5$ with $T = 5$ micro-activities and $f' = 3$ distinct features per micro-activity.




| $A_5$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| $v_1$ | 0 | 0 | 0 | 0 | 1 |
| $v_2$ | 0 | 0 | 0 | 0 | 1 |
| $v_3$ | 0 | 0 | 0 | 0 | 1 |
| $v_4$ | 0 | 0 | 0 | 0 | 1 |
| $v_5$ | 0 | 0 | 0 | 0 | 1 |

**Figure 6.1:** Activity-graph transformation for the prediction of sequential activities with GNNs (adapted from [2]). This example converts a macro-activity $M_T$ composed of $T = 5$ sequential micro-activities into a graph $G_T$ with five vertices. Each vertex $v_t \in V_T$ encodes the $f' = 3$-dimensional feature vector $x_t \in \mathbb{R}^{f'}$ of a micro-activity $m_t$ with $t \in \{1, \ldots, T\}$. The edges connect each vertex to the last vertex in time, i.e., the $T$-th vertex.

Since each $m_t$ represents a signal window $s_t \in \mathbb{R}^{w \times f}$, the unprocessed flattened feature vector of each micro-activity contains $wf$ elements, which can potentially

be a large number (cf. Table 5.3). To reduce the dimensions of the vertex feature vectors $x_t$ to a smaller $f'$, this work applies a UMAP [108] to $s_t$, resulting in the vertex feature matrix $X_T \in \mathbb{R}^{T \times f'}$. However, other feature reduction techniques, such as learned autoencoders or Principal Component Analysis (PCA) are also feasible [53].

Attaching a freshly observed micro-activity $m_{T+1}$ to an existing graph $G_T$ requires adjustments to the graph's vertex list, its feature matrix, and its adjacency matrix:

1. Append $v_{T+1}$ to the vertex list $V_T$: $V_{T+1} = V_T \| v_{T+1}$.
2. Append $x_{T+1}$ to the feature matrix $X_T$: $X_{T+1} = X_T \| x_{T+1}$
3. Expand the dimensions of the adjacency matrix $A_T$ by one row and one column; set all values in the new $T+1$-th column to one and replace the values in the $T$-th column with zeros.

These steps result in the updated graph $G_{T+1}$, along with the updated representations $V_{T+1}$, $X_{T+1}$, and $A_{T+1}$. Figure 6.2 illustrates the attachment step for the adjacency matrix from Figure 6.1.

| $A_5$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| $v_1$ | 0 | 0 | 0 | 0 | 1 |
| $v_2$ | 0 | 0 | 0 | 0 | 1 |
| $v_3$ | 0 | 0 | 0 | 0 | 1 |
| $v_4$ | 0 | 0 | 0 | 0 | 1 |
| $v_5$ | 0 | 0 | 0 | 0 | 1 |

append $v_6$

| $A_6$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ |
|---|---|---|---|---|---|---|
| $v_1$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $v_2$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $v_3$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $v_4$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $v_5$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $v_6$ | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 6.2:** Appending a newly observed micro-activity $m_{T+1}$ to the graph $G_T$ increases the dimensions of its adjacency matrix by one. The updated adjacency matrix $A_{T+1}$ contains ones in the $T+1$-th column and zeros in the $T$-th column.

This activity-graph transformation constructs micro-activity graphs that do not explicitly preserve the chronological order of activities as each vertex is connected only to the final vertex in time, $v_T$. However, temporal order is essential for

accurate sequential activity prediction. To address the absence of inherent temporal structure in the micro-activity graphs, the GNN introduced in the following section incorporates positional encodings and an attention mechanism.

## 6.2 Model Design

Figure 6.3 presents a GNN architecture for the detection and validation of sequential activities using graph link prediction. This model processes activity graphs $G_T = (X_T, A_T)$ in three steps, with the second and third steps executed in parallel:

1. Graph Embedding,
2. Link Prediction,
3. Graph Validation.

The Graph Embedding block updates the feature matrix $X^{T \times f'}$ with Positional Encodings (PE, [149]) and a GAT layer (cf. Section 3.2). PE enriches the vertex feature vectors with positional information that enable the model to capture temporal relationships within a micro-activity sequence. The GAT layer changes the vertex feature dimension from $f'$ features to $c$ channels and weights the updated vertex features by dynamically attending to the features of neighboring vertices using $n_h$ attention heads. As a result, the GAT layer transforms the vertex features $X$ from shape $(T, f')$ to $(T, n_h c)$. The attention mechanism only affects the features $x_T$ of the last vertex in time $v_T$, as this is the only vertex with incoming edges (cf. Figure 6.1). Thus, the updated $x_T$ represents the whole graph $G_T$, i.e., the whole micro-activity sequence. Finally, Last Vertex Pooling (LVP, cf. Equation 3.13) extracts and outputs the graph embedding $E(G_T) = x_T \in \mathbb{R}^{n_h c}$.

The Link Prediction block checks whether a newly observed micro-activity $m_{T+1}$ with feature vector $x_{T+1} \in \mathbb{R}^{f'}$ and the $T$ preceding micro-activities span a common micro-activity sequence ($y_L = 1$) or not ($y_L = 0$). To do this, the link predictor concatenates $E(G_T)$ with the feature vector $x_{T+1}$ of the new activity

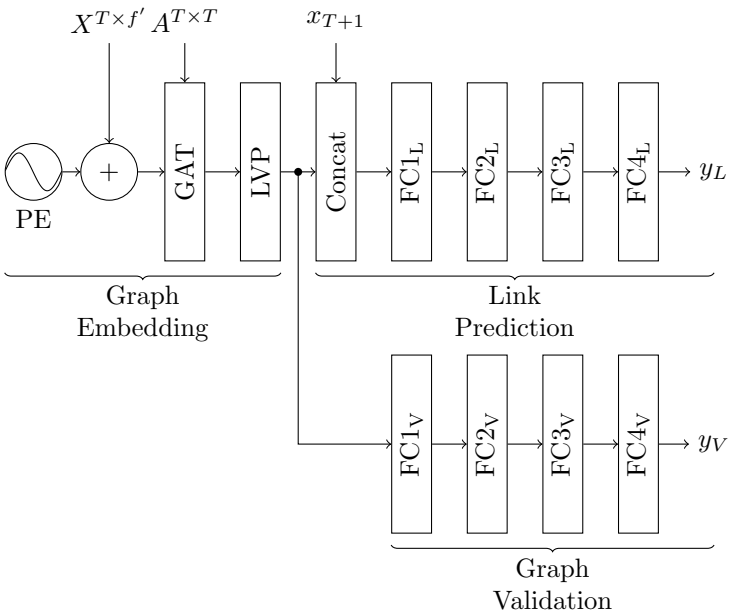



**Figure 6.3:** The GNN architecture for sequential activity prediction (adapted from [2]). The Graph Embedding block leverages Positional Encodings (PE) and structural information of the input graph $G_T = (X_T, A_T)$ to update and to extract the features of the last vertex $x_T \in X_T$. Link Prediction concatenates the graph embedding with the feature vector $x_{T+1}$ of the next micro-activity $m_{T+1}$ to check whether $m_{T+1}$ is part of the preceding micro-activity sequence $M_T$ or not. Graph Validation uses the graph embedding to check whether $G_T$ represents a complete macro-activity or not.

and processes the concatenated vector with four subsequent fully-connected layers ($\mathrm{FC1_L}$ to $\mathrm{FC4_L}$). The first three layers output $c$-sized vectors. Since link prediction is a binary classification task, $\mathrm{FC4_L}$ outputs a single decimal number with values greater than 0.5 referring to $y_L = 1$ and values less than 0.5 referring to $y_L = 0$.

The Graph Validation block checks whether a micro-activity graph $G_T$ represents a complete macro-activity $M_T$ ($y_V = 1$) or not ($y_V = 0$). To do this, the graph validator processes $E(G_T)$ with four subsequent fully-connected layers ($\mathrm{FC1_V}$ to $\mathrm{FC4_V}$). The first three layers keep $E(G_T)$'s dimensions unchanged; the last layer

outputs a single decimal number with values greater than 0.5 corresponding to $y_V = 1$ and values less than 0.5 corresponding to $y_V = 0$.

## 6.3 Implementation and Evaluation Details

Efficient training of the proposed GNN for sequential activity prediction requires suitable training data. However, WHAR datasets typically provide only signal-activity pairs without information about micro-activity sequences and macro-activities (e.g., [133, 141]). Therefore, several data preparation steps are necessary to enable sequential activity prediction with neural networks, including the generation of micro-activity embeddings and sequences. This section provides a comprehensive overview of the implementation and evaluation process by detailing graph and model hyperparameters (Section 6.3.1), describing the data preparation steps (Section 6.3.2), and outlining the training setup (Section 6.3.3).

### 6.3.1 Hyperparameters

The GNN and its input graphs contain several customizable hyperparameters that directly affect the recognition rates of sequential activities (cf. Table 6.1). $f'$ determines the feature dimensionality of the micro-activities after applying a UMAP (cf. Section 6.1). The GAT layer expands $f'$ to $c$ channels and learns $n_h$ attention matrices to develop a broad understanding of the micro-activities.

**Table 6.1:** Graph and model configuration.

| Parameter | Value(s) |
|---|---|
| Features $f'$ | $\{5, 15, 25\}$ |
| Channels $c$ | $2f'$ |
| Attention Heads $n_h$ | 5 |

Moreover, the layers apply specific activation functions that help learning meaningful activity patterns. Since Graph Validation and Link Prediction are binary classification tasks (cf. Section 6.2), their output layers $FC4_V$ and $FC4_L$ leverage the sigmoid activation function. All other layers use ReLU.

### 6.3.2 Data Preparation

This work evaluates the GNN for sequential activity prediction on two community benchmark datasets: Cooking [9, 88] and TT-Strokes [159].

#### The Cooking Dataset.

The Cooking dataset [9, 88] provides three-axis accelerometer signals from the left hip, the right arm, and both wrists during the preparation of three cooking recipes (macro-activities). Each time-series represents a segment of a recipe and includes up to six distinct cooking steps (micro-activities), though the temporal boundaries of these steps are not annotated. The dataset defines eight cooking actions cut (0), mix (1), open (2), peel (3), pour (4), put (5), take (6), and wash (7), but the sequence of these micro-activities within each recipe is not explicitly provided. To address this limitation, this work derives seven micro-activity sequences from these with micro-activities: 60567056705, 67056056705, 67056705605, 630565624, 656246305, 656305624, and 63056305630561. The first three micro-activity sequences represent the preparation of a sandwich, the next three sequences correspond to the preparation of cereals, and the last one refers to the preparation of a fruit salad.

#### The TT-Strokes Dataset.

The TT-Strokes dataset [159] contains three-axis accelerometer, gyroscope, and magnetometer signals measured at the racket-holding wrist during table tennis. Each time-series contains up to 15 repetitions of a single table tennis stroke type. This dataset contains signals of eight important table tennis strokes (micro-activities); however, it does not provide macro-activity labels. Thus, the handbook [118] serves as an external source of information that presents an extensive list of table

tennis exercises of which 55 exercises (macro-activities) are playable with the provided strokes backhand (BH) block (0), BH drive (1), BH loop (2), BH push (3), forehand (FH) block (4), FH drive (5), FH loop (6), and FH push (7): 0, 1, 2, 3, 4, 5, 6, 7, 02, 11, 12, 22, 24, 32, 33, 36, 37, 55, 56, 66, 73, 76, 77, 032, 111, 116, 213, 300, 302, 315, 330, 355, 373, 512, 555, 600, 662, 700, 0044, 1230, 1256, 2226, 2266, 3377, 3460, 5674, 6662, 7632, 72626, 76767, 662551, 3266666, 51624073, 222666156, and 7373202646.

### Deriving Micro-Activity Embeddings with UMAPs.

The activity-graph transformation converts micro-activity sequences into activity-graphs $G_T = (V_T, A_T)$, where each vertex corresponds to a micro-activity window (cf. Section 6.1). However, these graphs lack vertex features $X_T$, which are essential for graph-based learning. To generate meaningful vertex representations, this work employs UMAPs; a dimensionality reduction technique that yields more discriminative embeddings than alternatives like PCA [108].

This work splits the time-series from the TT-Strokes and Cooking datasets into non-overlapping $100\,\mathrm{ms}$-windows with $w = 10$ time steps and $f$ features ($f = 9$ for TT-Strokes, $f = 12$ for Cooking). It then embeds each micro-activity window into a lower-dimensional space using UMAPs with $f' \in \{5, 15, 25\}$ components.

The Cooking dataset introduces an additional challenge due to the presence of multiple micro-activity labels per window. To resolve this ambiguity, this work samples the final micro-activity embeddings from Gaussian distributions $N(\mu(m), \sigma)$, where $\mu(m)$ denotes the average UMAP embedding of all windows associated with micro-activity class $m$ and $\sigma$ is a fixed standard deviation, as follows:

1. Fit a seeded UMAP [108] with $f'$ components on the windows. This step reduces the feature dimensions per window from $wf$ values to $f'$ features.

2. Label the UMAP embeddings of each window with their respective micro-activity labels. If a window has multiple micro-activity labels, duplicate its embedding and assign the individual micro-activity labels to each duplicated embedding.

3. Calculate the average UMAP embeddings $\mu(m)$ per micro-activity label $m$.
4. Fit Gaussian distributions $N(\mu(m), \sigma)$ with a fixed standard deviation $\sigma$ per micro-activity $m$.
5. Generate 30 feature matrices $X_T$ per macro-activity $M_T$ by iterating over their micro-activities $m_t \in M_T$ and drawing unique micro-activity embeddings from the corresponding Gaussian distributions $N(\mu(m_t), \sigma)$.

For consistency, this work applies this data generation procedure to both datasets. Table 6.2 lists the average similarities between the mean UMAP embeddings (Step 3) of $n$ distinct micro-activity classes according to the following equation:

$$\text{similarity} = \frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d[i,j]}, \tag{6.1}$$

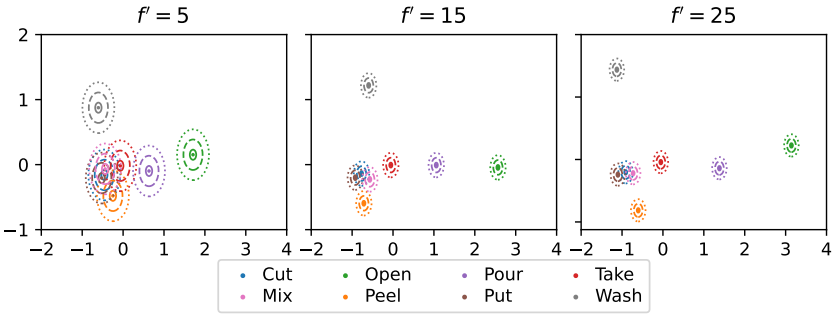
Here, $d[i,j]$ is the Euclidean distance between the mean UMAP embeddings of two micro-activities $\mu(m_i)$ and $\mu(m_j)$. Higher similarity scores indicate greater overlap between features of different micro-activity classes. The larger the feature dimension $f'$, the better the distinguishability between classes. However, larger feature vectors also increase the computational effort of the GNN.

**Table 6.2:** Average micro-activity similarity per dataset (adapted from [2]). Higher values indicate greater similarity and thus, worse distinguishability between micro-activities.

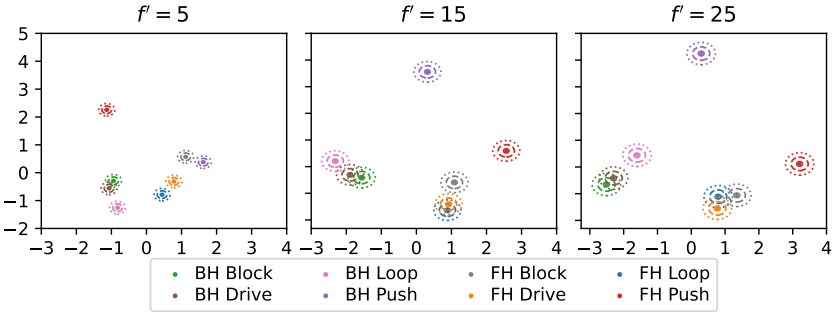
| Dataset | UMAP components $f'$ | | |
|---|---|---|---|
| | 5 | 15 | 25 |
| Cooking | 1.7235 | 1.0429 | 0.8657 |
| TT-Strokes | 0.5071 | 0.3024 | 0.2322 |

Figure 6.4 plots the two-dimensional PCA of the mean UMAP embeddings from Step 3 for both datasets and varying values of $f'$. The PCA is used solely for visualization. The circles around the dots indicate the standard deviations

$\sigma \in \{0.05, 0.15, 0.25\}$. These plots confirm that larger $f'$ improve the distinguishability between micro-activity classes. However, even with $f' = 25$, some micro-activities still have overlapping distributions.




**(a)** Cooking dataset (adapted from [2]).




**(b)** TT-Strokes dataset (adapted from [2]).

**Figure 6.4:** PCAs of the mean UMAP embeddings (seed 1) for two Cooking datasets. Dots: centroids, solid/dashed/dotted circles: standard deviations $\sigma \in \{0.05, 0.15, 0.25\}$.

### Generating Positive and Negative Samples.

Training a machine learning model for graph validation and link prediction requires positive and negative samples. Therefore, the pure existence of macro-activities is not sufficient to validate micro-activity sequences or to predict their

subsequent micro-activities. This work derives positive and negative samples from macro-activities as follows:

1. For each complete macro-activity $M_T = (m_1, \ldots, m_T)$, extract all prefix sequences $P(M_T) = \{M_{T'} \mid T' \in \{1, \ldots, T\}\}$. Prefix sequences $M_{T'} \in P(M_T)$ form prefix graphs $G_{T'} = (X_{T'}, V_{T'})$.
2. For each prefix sequence $M_{T'} \in P(M_T)$, extract all possible successive micro-activities $m_{T'+1}$ with $M_{T'} \| m_{T'+1} \in P(M_T)$ and all impossible sequences with $M_{T'} \| m_{T'+1} \notin P(M_T)$.

Prefix sequences $M_{T'}$ with $T' \neq T$ represent incomplete micro-activity sequences and carry the graph validation label $y_V = 0$. In contrast, sequences with $T' = T$ represent complete macro-activities with the validation label $y_V = 1$.

A link $m_{T'+1}$ is possible if the concatenated sequence $M_{T'+1} = M_{T'} \| m_{T'+1}$ forms a prefix of the macro-activity $M_T$, i.e., $M_{T'+1} \in P(M_T)$. Possible links have the link prediction label $y_L = 1$. If the concatenated sequence is not a prefix of $M_T$, i.e., $M_{T'+1} \notin P(M_T)$, the link is impossible and has the label $y_L = 0$.

As an example, Table 6.3 lists the prefix sequences $M_{T'}$ of the table tennis exercise 1230 along with its possible and impossible links $m_{T'+1}$. Since 11, 1256, and the single-activity sequences are also valid macro-activities in the table tennis dataset, multiple $m_{T'+1}$ are possible for $T' \in \{0, 1, 2\}$.

**Table 6.3:** The prefix sequences $M_{T'} \in P(M_T)$ of the table tennis exercise $M_T = 1230$ along with its possible and impossible links $m_{T'+1}$.

| $T'$ | $M_{T'}$ | Possible $m_{T'+1}$ | Impossible $m_{T'+1}$ |
|---|---|---|---|
| 0 | () | $\{0, 1, 2, 3, 4, 5, 6, 7\}$ | $\emptyset$ |
| 1 | $(1)$ | $\{1, 2\}$ | $\{0, 3, 4, 5, 6, 7\}$ |
| 2 | $(1, 2)$ | $\{3, 5\}$ | $\{0, 1, 2, 4, 6, 7\}$ |
| 3 | $(1, 2, 3)$ | $\{0\}$ | $\{1, 2, 3, 4, 5, 6, 7\}$ |
| 4 | $(1, 2, 3, 0)$ | $\emptyset$ | $\{0, 1, 2, 3, 4, 5, 6, 7\}$ |

#### Data Splits for Model Training.

This work randomly splits the graphs $G_T$ of complete macro-activities $M_T$ into $60\,\%$ train, $20\,\%$ validation, and $20\,\%$ test sets. To avoid data leakage, prefix graphs $G_{T'}$ belong to the same sets as their complete graphs $G_T$. Since macro-activities vary in length, the number of prefix graphs per complete graph varies. Consequently, the actual split sizes differ from the 60/20/20 split on the complete graphs. This data preparation strategy results in 284,916 graphs for Cooking and 853,062 graphs for TT-Strokes. Table 6.4 lists the number of valid ($y_V = 1$) and invalid ($y_V = 0$) graphs as well as the number of samples with possible ($y_L = 1$) and impossible ($y_L = 0$) links.

**Table 6.4:** Class distributions of Cooking and TT-Strokes datasets (adapted from [2]).

| Dataset | $y_V = 1$ | $y_V = 0$ | $y_L = 1$ | $y_L = 0$ |
|---|---|---|---|---|
| Cooking | 19,614 | 265,302 | 51,084 | 233,832 |
| TT-Strokes | 688,326 | 164,736 | 213,840 | 639,222 |

### 6.3.3 Training Setup

The GNN for sequential activity prediction is a multi-output model that computes graph validation and link prediction scores with two separate network branches. To account for the large imbalance in the graph labels (cf. Table 6.4), weighted binary cross-entropy losses [29, 53] with balanced class weights

$$w_y = \frac{n}{|Y| n_y}, \tag{6.2}$$

guide the learning process of both modalities. Here, $n$ is the total number of samples in the training set and $n_y$ is the number of samples with class $y \in Y$. The weights are updated with the Adam optimizer [85], learning rate 0.001, and gradient norm clipping [125]. Furthermore, this work applies early stopping with

patience 10 to terminate the training when the validation loss stagnates, thereby reducing the risk of overfitting.

During training, the GNN processes graphs in batches of 128. To account for statistical instabilities, this work repeats data generation and training three times with different random seeds $\in \{1, 2, 3\}$.

## 6.4 Performance Analysis

This work evaluates the GNN's link prediction and graph validation capabilities in terms of accuracies and macro F1-scores (cf. Equations 2.10 and 2.11) on two benchmark datasets. It compares different vertex feature configurations and analyzes the impact of positional encodings in the graph embedding block of the GNN (cf. Figure 6.3) on the performance metrics. The presented scores are averaged over three runs.

### Cooking.

Figure 6.5 illustrates the performance metrics for graph validation and link prediction on the Cooking dataset with different standard deviations $\sigma \in \{0.05, 0.15, 0.25\}$ and UMAP embedding sizes $f' \in \{5, 15, 25\}$. Furthermore, the charts present the effects of adding and omitting the positional encoding. The following two paragraphs analyze the performance of the GNN with positional encodings, while the final paragraph discusses performance without them.

In general, the accuracies and F1-scores improve with increasing $f'$ and decreasing $\sigma$. For graph validation, the accuracies range from 0.968 for $f' = 5$ and $\sigma = 0.25$ to 1.0 for $f' = 25$ and $\sigma = 0.05$. The differences in the macro F1-scores are slightly larger with values ranging from 0.882 to 1.0 for the same configurations. With accuracies ranging from 0.854 to 0.998 and macro F1-scores lying between 0.648 and 0.997, these observations are also valid for link prediction. However, the gaps between the best and worst configurations are much greater for link prediction than for graph validation. Possible reasons for these trends are the large

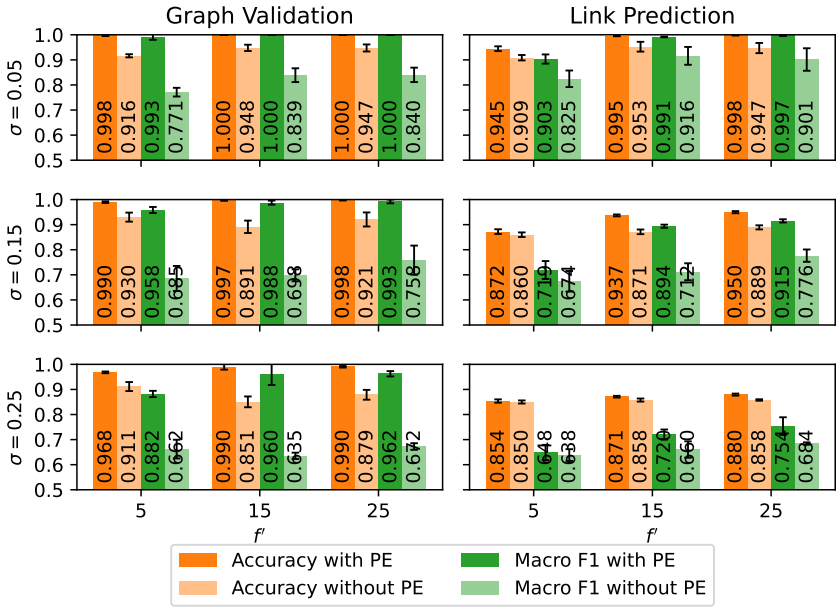



**Figure 6.5:** Graph validation and link prediction results for different standard deviations $\sigma$ and varying UMAP embedding sizes $f'$ on the Cooking dataset (adapted from [2]). The plots compare accuracies and macro F1-scores with and without positional encodings.

similarities between UMAP embeddings (cf. Table 6.2), the high class imbalance (cf. Table 6.4), and the similarities of macro-activities (cf. Section 6.3.2), which complicate the prediction of possible links and the validation of micro-activity sequences in the cooking domain.

Remarkably, the GNN consistently achieves the same or even better scores with $f' = 25$ as with $f' = 5$ and the next smaller $\sigma$. For instance, the macro F1-score for $f' = 25$ and $\sigma = 0.25$ is 0.035 points higher than the macro F1-score for $f' = 5$ and $\sigma = 0.15$ in link prediction. The same effects are observable with graph validation, but the differences are smaller due to the already higher recognition rates. Thus, the positive impact of larger embedding dimensions $f'$ on the classification performance surpasses the negative impact of larger standard deviations $\sigma$ for both classifiers.

Positional encodings significantly boost the performance of the GNN. For graph validation, macro F1-score improvements reach up to 0.325 ($f' = 15, \sigma = 0.25$), while link prediction sees gains of up to 0.182 ($f' = 15, \sigma = 0.15$). Accuracy also increases substantially, with improvements of up to 0.139 for graph validation and 0.066 for link prediction under the same configurations. These results highlight the importance of incorporating sequential information into the GNN architecture for the accurate detection and validation of micro-activity sequences; especially since the underlying graph structure does not inherently capture temporal dependencies between consecutive micro-activities.

### TT-Strokes.

Figure 6.6 illustrates the performance metrics for graph validation and link prediction for TT-Strokes with different standard deviations $\sigma$ and UMAP embedding sizes $f'$. Once again, the charts compare the impact of positional encodings.

The graph validation and link prediction results on the TT-Strokes dataset follow the same trends observed in the Cooking dataset. The metrics improve with increasing $f'$ and decreasing $\sigma$, with $f'$ having a stronger impact on performance. Again, positional encodings lead to massive performance gains. However, overall scores on TT-Strokes are significantly higher, resulting in smaller performance differences between the best and worst GNN and data configurations.

With positional encodings, the graph validation F1-scores range from 0.905 ($f' = 5$, $\sigma = 0.25$) to 1.0 ($f' = 25$, $\sigma = 0.05$); the accuracies range from 0.942 to 1.0. With macro F1-scores ranging from 0.875 to 1.0 and accuracies ranging from 0.913 to 1.0, the differences are again more pronounced for link prediction. These results indicate that the link prediction task is more challenging and complex than graph validation.

Notably, even the GNNs that operate on graphs with $f' = 15$ vertex features outperform their $f' = 5$ counterparts with the next smaller $\sigma$ in link prediction and graph validation across the board. This observation aligns well with the numbers from Table 6.2, which state that the micro-activities in the TT-Strokes dataset are easier to distinguish than the micro-activities in the Cooking dataset.

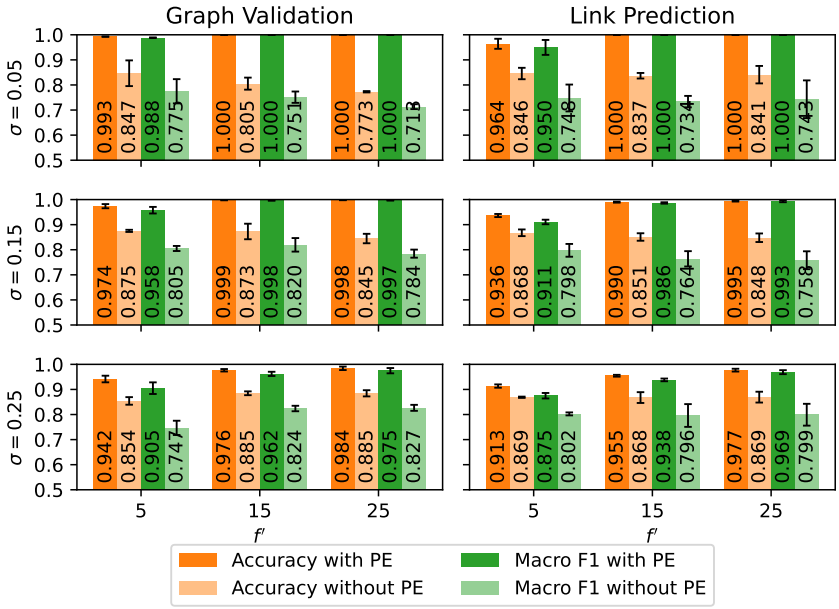



**Figure 6.6:** Graph validation and link prediction results for different standard deviations $\sigma$ and varying UMAP embedding sizes $f'$ on the TT-Strokes dataset (adapted from [2]). The plots compare accuracies and macro F1-scores with and without positional encodings.

Without positional encodings, the GNN shows performance degradations of up to 0.287 in the macro F1-scores and 0.227 in the accuracies during graph validation with $f' = 25$ and $\sigma = 0.05$. The link prediction scores decrease by up to 0.266 for the F1-score and 0.163 for the accuracy with $f' = 15$ and $\sigma = 0.05$. Once again, these results demonstrate the importance of temporal information for sequential activity prediction.

## 6.5 Summary

This chapter redefined sequential activity recognition as a graph link prediction problem. It introduced a novel activity-graph transformation that constructs expandable micro-activity graphs from sequences of signal windows, where each vertex represents one micro-activity and connects to the most recent vertex. Therefore, the latest vertex gathers information from all previous micro-activities during GNN processing, enabling tasks such as sequence validation (graph validation) and future activity prediction (link prediction). Although the resulting adjacency matrices are sparse, they are only reusable for graphs of the same size, meaning activity sequences must have equal length. To support efficient sequence analysis, the transformation applies UMAP to reduce the dimensionality of signal windows before assigning them as vertex features. However, practical deployment in wearable applications demands a lighter feature extraction method that retains key signal characteristics.

Additionally, this chapter presented a GNN architecture that determines whether a micro-activity graph forms a complete sequence (graph validation) or requires additional steps to complete the associated macro-activity (link prediction). To address the lack of labeled micro- and macro-activity sequences in existing WHAR datasets, this chapter also presented a method for generating such labels. Experiments showed that the GNN accurately detects and validates long-running sequential activities. Link prediction remains more challenging than graph validation, since many macro-activities share similar prefix sequences. Future work could explore classifiers that recognize both micro- and macro-activities or detect nested and combined activity patterns, enabling broader applications with greater impact on users' daily lives. Such advancements, however, rely on well-labeled training datasets that capture a diverse range of real-world scenarios.

# Part III

# Advanced Signal Generalization with Foundation Models

# 7 Preparation Steps for Effective Pre-Training

State-of-the-art WHAR models are typically trained on small benchmark datasets that contain signals from only a few users performing a limited set of activities (e.g., [1, 177]). As a result, these models are highly specialized and constrained by the narrow scope of their training data and objectives. Examples include the GNN-based models from Part II, which focus on resource-efficient activity classification from single datasets (cf. Chapter 5) and the detection of composite activities (cf. Chapter 6). In contrast, Foundation Models (FMs) aim to develop a general understanding of their application domain that enables a wide range of downstream applications. However, FMs require large and diverse datasets to identify broadly applicable data patterns (cf. Chapter 4). In the WHAR domain, such datasets are rare due to the variability of activities, users, and sensing devices (cf. Section 2.5). Based on the approaches proposed in *Inertial Signal Forecasting With Foundation Model Techniques* [3] and *Enhancing Sensor-Based Human Activity Recognition With Multiresolution Wavelet-Attention* [4], this chapter introduces a universal data preparation strategy. This strategy enables efficient pre-training of WHAR-FMs, laying the groundwork for more versatile and scalable activity recognition models.

## 7.1 Harmonizing Heterogeneous Signals

WHAR-FMs derive general signal understanding from (unlabeled) training data in a self-supervised fashion (cf. Section 4.1). However, publicly available WHAR

datasets typically lack depth, covering only a few users, activities, or samples (cf. Section 2.5). To facilitate pre-training of WHAR-FMs, signal fusion from diverse sources is essential. Unfortunately, benchmark datasets vary significantly in sensor types, placements, and activities. Therefore, selecting (Section 7.1.1) and aligning (Section 7.1.2) comprehensive datasets is crucial to ensure signal compatibility between different data sources for effective pre-training.

### 7.1.1 Dataset Selection

To achieve general signal understanding, WHAR-FMs require comprehensive pre-training datasets that encompass a wide range of activities, users, and sensors. However, common benchmark datasets typically include only a limited set of sensor signals from a few users and activities (cf. Table 2.2). Hence, this work presents the following three dataset selection criteria that streamline the development of foundation models for WHAR:

1. "Each sensor location should contain signals from at least one accelerometer and one gyroscope." [3]
2. "Each location should be present in at least two datasets." [3]
3. "The datasets should provide raw sensor signals." [3]

Accelerometers and gyroscopes are among the most common sensors in WHAR (cf. Table 2.2). These sensors provide a detailed view on human motions by detecting the acceleration and angular velocity of selected body parts. Additional sensors, such as magnetometers or pressure sensors, can further improve WHAR applications [158]. However, using more sensors also leads to increased resource consumption during signal collection and processing. Hence, Criterion 1 focuses solely on accelerometers and gyroscopes.

The location of sensors on the human body significantly influences the distributions of the collected signals. For instance, wrist signals from activities that

mainly involve arm movements differ considerably from those captured at the ankle (cf. Figure 7.1). Since proper validation of machine learning models requires test data with a distribution similar to the training data [53], Criterion 2 ensures that each sensor location is present in multiple datasets.

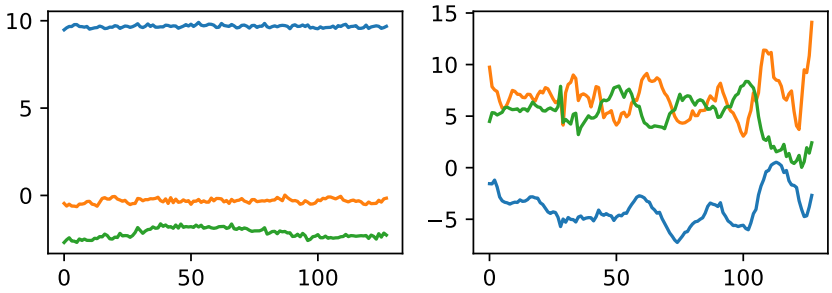
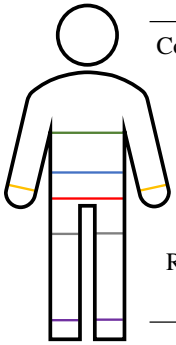



**Figure 7.1:** Three-axis accelerometer signals recorded at the ankle (left) and wrist (right) during ironing (signals from [133]). The ankle plot shows low-variance signals that are nearly constant, indicating minimal movement, while the wrist plot features significant changes in acceleration over time.

Pre-processing can alter the distribution of the collected signals, with different pre-processing methods impacting the data in various ways. For instance, normalization modifies the range and statistical properties of sensor readings [70], while filtering suppresses certain frequency ranges [111]. To ensure compatibility across datasets, Criterion 3 mandates the availability of raw sensor signals.

This work identifies eight datasets that fulfill these requirements (cf. Figure 7.2). mHealth's chest data and DLA's wrist data are excluded as they only provide acceleration signals, while RealWorld's head and upper arm signals are ignored since these locations are not present in any other dataset. Due to the lack of standardized sensor location names in literature, this work categorizes them into six common positions: ankle, chest, hip, pocket, waist, and wrist. In this grouping, ankle includes signals from the ankle or shin, pocket represents signals from the pocket, upper leg, or thigh, and wrist encompasses signals from the left or right wrist, as well as the forearm.



| Dataset | Ankle | Chest | Hip | Pocket | Waist | Wrist |
|---|---|---|---|---|---|---|
| Complex HAD [138, 139] | - | - | - | 1 | - | 1 |
| DLA [92] | 1 | - | 1 | - | - | - |
| DSADS [15] | - | 1 | - | 2 | - | 2 |
| FLAAP [87] | - | - | - | - | 1 | - |
| mHealth [13] | 1 | - | - | - | - | 1 |
| PAMAP2 [133] | 1 | 1 | - | - | - | 1 |
| RealWorld (HAR) [146] | 1 | 1 | - | 1 | 1 | 1 |
| USC-HAD [172] | - | - | 1 | - | - | - |
| $\sum$ | 4 | 3 | 2 | 4 | 2 | 6 |

**Figure 7.2:** The standardized sensor placements on the human body (left) and the number of six-axis sensors at these locations (right) (adapted from [3]).

Moreover, the datasets differ in the number of activities (cf. Table 2.2), their types (e.g., daily living vs. sports), and their granularity (e.g., walk vs. walk circle). Like the sensor location names, the activity labels are not standardized either. Thus, this work generalizes the activity labels across dataset boundaries. The datasets share 40 common activities, making the combined dataset significantly more comprehensive than the individual datasets. The general activity labels are: basketball, cleaning, cross trainer, crouching, cycling, downstairs, drink, driving, dusting, eating, elevator, face washing, frontal arm elevation, hand washing, handwriting, ironing, jump, laptop, laundry, lay, nordic walking, relax, rowing, rubbing, run, sit, situp, sleep, smoke, soccer, stand, stepper, sweeping, talk, teeth brush, tv, upstairs, vacuuming, waist bends, and walk.

### 7.1.2 Signal Alignment

Research groups collect WHAR datasets with different sensing devices from different manufacturers in real-world or laboratory environments. Therefore, each dataset has different properties (Table 7.1). To address these discrepancies and to facilitate the development of WHAR-FMs, this work applies several signal

alignment strategies. In addition to the generalization of the location and activity labels (cf. Section 7.1.1), these strategies involve the standardization of units, sampling rates, and window sizes.

**Table 7.1:** Properties of the eight selected datasets (adapted from [3]). |L|: number of individual sensor locations, SI: original units of acceleration (A) and angular velocity (G), SR: original sampling rate in Hz, |W|: number of windows after alignment (per sensor location |W(L)|). For additional information, see Table 2.2.

| Dataset | SI (A) | SI (G) | SR | \|W\| | \|L\| | \|W(L)\| |
|---|---|---|---|---|---|---|
| Complex HAD [138, 139] | $\mathrm{m/s^2}$ | $\mathrm{rad/s}$ | 50 | 22.298 | 2 | 11.149 |
| DLA [92] | $\mathrm{g}$ | $\mathrm{deg/s}$ | 100 | 36.530 | 2 | 18.265 |
| DSADS [15] | $\mathrm{m/s^2}$ | $\mathrm{rad/s}$ | 25 | 177.080 | 5 | 35.416 |
| FLAAP [87] | $\mathrm{m/s^2}$ | $\mathrm{rad/s}$ | 100 | 10.142 | 1 | 10.142 |
| mHealth [13] | $\mathrm{m/s^2}$ | $\mathrm{deg/s}$ | 50 | 10.458 | 2 | 5.229 |
| PAMAP2 [133] | $\mathrm{m/s^2}$ | $\mathrm{rad/s}$ | 100 | 61.209 | 3 | 20.403 |
| RealWorld (HAR) [146] | $\mathrm{m/s^2}$ | $\mathrm{rad/s}$ | 50 | 258.580 | 5 | 51.716 |
| USC-HAD [172] | $\mathrm{g}$ | $\mathrm{deg/s}$ | 100 | 20.768 | 1 | 20.768 |

### Units.

The selected datasets capture signals using various measurement units. Accelerometers (A) record signals in $\mathrm{m/s^2}$ or $\mathrm{g}$, while gyroscopes (G) measure in $\mathrm{rad/s}$ or $\mathrm{deg/s}$. To harmonize the input distribution, this work converts all accelerometer readings to $\mathrm{m/s^2}$ and all gyroscope readings to $\mathrm{rad/s}$ as follows:

$$\mathrm{A}\left[\mathrm{m/s^2}\right] = \mathrm{A}\left[\mathrm{g}\right] * g_n, \tag{7.1}$$

$$\mathrm{G}\left[\mathrm{rad/s}\right] = \mathrm{G}\left[\mathrm{deg/s}\right] * \pi/360, \tag{7.2}$$

where $g_n = 9.806\,65\,\mathrm{m/s^2}$ represents the standard acceleration of gravity [30].

### Sampling Rate.

The sampling rate affects the temporal resolution of a signal, with higher sampling rates capturing more details about the signal's variations. The selected datasets collect signals at varying sampling rates, with most of them using frequencies of $50\,\mathrm{Hz}$ or higher. To maintain consistent temporal resolution in the input distribution of the consolidated dataset, this work resamples all signals to $50\,\mathrm{Hz}$ with a cubic spline interpolation [35], requiring only the signals from DSADS to be upsampled, while four datasets are downsampled. This rate effectively captures human motion, as most of the activities fall within the 0 to $15\,\mathrm{Hz}$ range [33]. Additionally, resampling to $50\,\mathrm{Hz}$ enables ubiquitous applications of WHAR-FMs, as this lower sampling rate requires less processing resources and storage compared to higher frequencies.

### Windowing.

Following the considerations outlined in [10], this work divides the signals into $2.56\,\mathrm{s}$ windows with a $50\,\%$ overlap. With the sampling rate of $50\,\mathrm{Hz}$ and dataset selection criterion 1, each location-specific window $(w, f)$ contains $w = 2.56\,\mathrm{s} * 50\,\mathrm{Hz} = 128$ data points with $f = 6$ features from a three-axis accelerometer and a three-axis gyroscope. The |W(L)| column in Table 7.1 lists the total number of windows per dataset and location. For each dataset, the windows are evenly distributed across its available sensor locations. Table 7.2 summarizes the total number of windows collected across all eight datasets, highlighting the varying amounts of data per sensor location.

**Table 7.2:** Total number of windows per sensor location.

| Ankle | Chest | Hip | Pocket | Waist | Wrist |
|---|---|---|---|---|---|
| 95.613 | 107.535 | 39.033 | 133.697 | 61.858 | 159.329 |

## 7.2 Location-Invariant Pre-Training

Training FMs typically requires large volumes of data (cf. Chapter 4). However, WHAR datasets are often limited in size, featuring only a few available sensor locations (cf. Figure 7.2) and less than 35k windows per sensor location (cf. Table 7.2). To address this challenge, this work introduces location-invariant pre-training as an efficient approach for neural networks to develop general knowledge about WHAR signals. Table 7.3 compares important properties of location-invariant processing with multi-location processing.

**Table 7.3:** Differences between location-invariant and multi-location processing of WHAR signals.

| Aspect | Location-Invariant Processing | Multi-Location Processing |
|---|---|---|
| Input Dimensions | $(w, f)$ per location $l \in L$ | $(w, \|L\|f)$ |
| Training Corpus | $\|L\|$ samples per window (one per $l$) | 1 sample per window (stacked features from all $l$) |
| Sensor Arrangement | Dynamic: support varying sensor placements | Static: require fixed, pre-defined sensor placements |
| Learned Representations | Location-independent | Learn correlations between fixed locations |
| Adaptability to New Locations | High | Limited due to fixed sensor placements |
| Flexibility for Downstream Tasks | High | Limited due to fixed sensor placements |

Location-invariant pre-training enhances the training of WHAR-FMs by treating $(w, f)$-shaped signals from various body locations $l \in L$ as separate input streams, even when a dataset encompasses multiple locations. This technique effectively addresses the limited availability of WHAR signals by increasing the number of individual training samples by a factor of $|L|$ (number of available locations)

without relying on artificial data augmentation. The pre-training objectives remain identical to multi-location WHAR-FMs (cf. Section 4.2).

As a result, location-invariant pre-training learns highly flexible and universally applicable signal representations that are independent of the sensor arrangement on the human body. This adaptability allows models to seamlessly integrate new sensor locations without requiring modifications to the model architecture or additional data collection for already existing training samples. Moreover, location-invariant processing simplifies the merging of datasets with different available sensor locations, as only the number and types of sensors at each location are predetermined, while the sensor locations can be adjusted dynamically.

In contrast to multi-location models with pre-defined and fixed sensor arrangements that stack signals from all available locations into $(w, |L|f)$-shaped samples (e.g., [1, 177]), location-invariant pre-training cannot identify correlations between sensor locations. Nevertheless, downstream tasks benefit from the location-invariant processing of pre-trained WHAR-FMs, as each task can define its own subset of required sensor placements and flexibly merge the single-location signal representations without relying on a predefined sensor arrangement.

Moreover, location-invariant pre-training enables Leave-One-Dataset-Out cross-validation (LODO), where samples from $n - 1$ datasets are randomly split into training and validation sets, while the $n$-th dataset serves as the unseen test set (cf. Figure 7.3). Dataset selection criterion 2 ensures that this test set always contains a subset of the locations that are present during training.

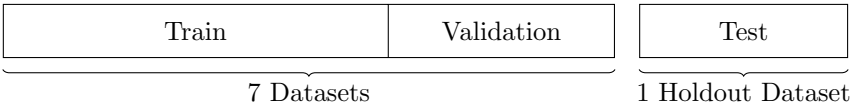



**Figure 7.3:** Leave-One-Dataset-Out (LODO) pre-training split for eight available datasets (adapted from [3]).

## 7.3 Summary

This chapter addressed the heterogeneity of publicly available WHAR datasets by proposing a reusable signal alignment strategy. This approach involves selecting compatible datasets, aligning signal representations across different sensing setups, and standardizing activity and sensor location labels. Using this method, this work consolidated eight datasets that include accelerometer and gyroscope signals from six sensor locations across 40 daily living, sports, and household activities into a unified corpus for training WHAR-FMs. Researchers can apply the same strategy to additional datasets to further expand the training corpus in the future.

Additionally, this chapter introduced a location-invariant approach for the self-supervised pre-training of WHAR-FMs. This method allows the models to process input signals regardless of the sensor placement on the human body, enabling them to develop a general understanding of human activity signals with diverse characteristics. By eliminating the need to define sensor locations in advance, location-invariant pre-training enhances the flexibility of both the pre-trained models and the pre-training process itself. To support a wide range of downstream applications, the chapter also introduced multi-location fine-tuning, which adapts the learned representations to task-specific multi-location sensing setups.

# 8 Signal Forecasting with Foundation Models

Accurate signal forecasting can enhance activity classification tasks while reducing the need for additional data collection efforts [79]. It also facilitates the detection of signal anomalies [173]. However, sensor signals can vary significantly across individuals, sensor locations, and activities (cf. Figures 2.6 and 7.1). To address this variability, this chapter introduces a WHAR-FM for signal forecasting that develops a comprehensive understanding of human activity signals from diverse data collection environments through local and global signal analysis. This Dual-View FM, initially presented in *Inertial Signal Forecasting With Foundation Model Techniques* [3], treats signals from various sensor locations on the human body equally, establishing it as a location-invariant model for signal forecasting. A comprehensive analysis of its pre-training and fine-tuning performances against state-of-the-art architectures demonstrates the model's effectiveness in delivering accurate signal forecasts. Additionally, this chapter explores the impact of sensor placement meta-information and pre-train data volumes on the forecasts.

## 8.1 Model Design

Signal forecasting models aim to predict future signal curves based on previous time steps (cf. Section 2.4). A recent model for time-series forecasting is TSMixer (Time-Series Mixer) [26]. The core of TSMixer is the Mixer Layer (cf. Figure 8.1), which processes input time-series $(w, f)$ with $w$ time steps and $f$ features per time step in two steps. During time-mixing, the Mixer Layer transforms the

time-series with a Multi-Layer Perceptron (MLP) along the time axis $w$. During feature-mixing, the Mixer Layer applies a second MLP along the feature axis $f$ of the intermediate representation. Appendix A.1 provides a high-level overview of the complete TSMixer architecture.

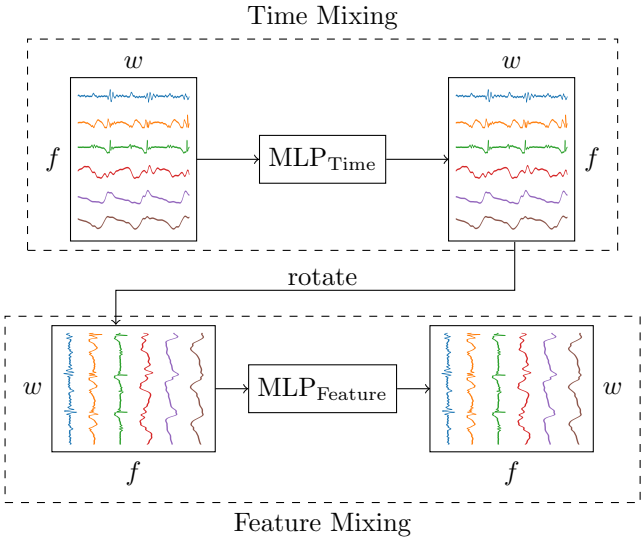


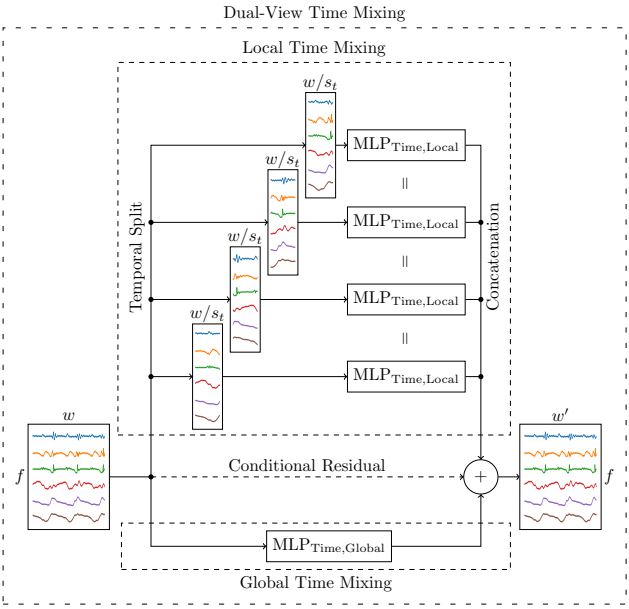

**Figure 8.1:** Simplified view of the Mixer Layer (adapted from [26]). The signal from [13] is a $(w, f)$-shaped time-series with $w$ time steps and $f = 6$ features (blue, orange, green: three-axis acceleration; red, purple, brown: three-axis angular velocity).

Unlike complex architectures such as Transformers, TSMixer's MLPs rely solely on fully-connected, dropout, and normalization layers, making this mixing approach lightweight and attractive for wearable signal forecasting. However, the fully-connected layers fuse inputs globally across the entire time or feature axis, which can be problematic for human activity signals with fast, abrupt changes (cf. signals in Figure 8.1). In such cases, global time mixing may overlook important spikes, while global feature mixing may suppress distinct characteristics from different sensors.

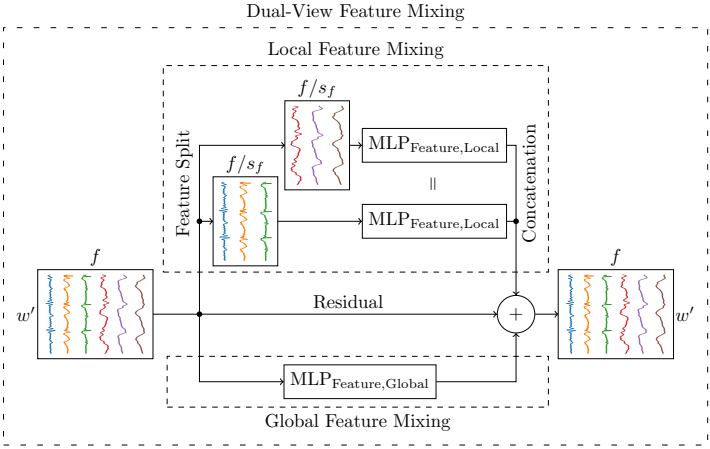
To overcome these limitations, this work presents Dual-View FM, a neural network for the forecasting of wearable sensor signals. In addition to TSMixer's global view on the signals (Global Time and Feature Mixing), Dual-View FM introduces Local Time and Feature Mixing operations, which divide input signals into smaller chunks to capture more focused context and finer-grained structures within them. Figure 8.2 illustrates these Dual-View Mixing operations. Local Time Mixing (cf. Figure 8.2a) splits the time axis into $s_t$ equally-sized, non-overlapping chunks with $w \mod s_t = 0$ and processes them separately with an MLP. Hence, Local Time Mixing inherently models the chronological order of the time chunks, whereas Global Time Mixing does not consider such temporal information. Local Feature Mixing (cf. Figure 8.2b) splits the feature axis into $s_f$ chunks with $f \mod s_f = 0$. In contrast to Global Feature Mixing, which fuses information across all sensors, Local Feature Mixing partitions signals by feature modality, e.g., into three-axis accelerometer and gyroscope signals, thereby enhancing the model's ability to capture sensor-specific characteristics.

Dual-View Mixing leverages four MLPs. $\text{MLP}_{\text{Time,Local}}$ processes the local time chunks, $\text{MLP}_{\text{Feature,Local}}$ processes the local feature chunks, $\text{MLP}_{\text{Time,Global}}$ operates on the entire time axis of the input signal, and $\text{MLP}_{\text{Feature,Global}}$ operates on the entire feature axis. Each MLP contains three sequential layers: a fully-connected layer (FC1), a dropout layer, and a second fully-connected layer (FC2). During feature mixing, FC1 expands the feature dimension from $f$ to $h$ and FC2 projects it back to $f$. During time mixing, FC1 expands the time dimension from $w$ to $h$; FC2 either projects it back to $w$ (for signal reconstruction) or to $w'$ (for signal forecasting). To reduce memory requirements and treat each chunk equally, the local MLPs share their weights across all chunks along their respective signal dimensions.

Residual connections (or skip connections) mitigate the risk of information loss during network propagation and enable the training of deeper models [53, 64]. To prevent shape mismatches when summing the input signal with the locally and globally mixed time-series, the residual connection in Dual-View Time Mixing is applied only during signal reconstruction tasks.




**(a)** Local Time Mixing splits the time axis $w$ in $s_t = 4$ chronological chunks; Global Time Mixing mixes the entire time axis.




**(b)** Local Feature Mixing splits the feature axis $f$ in $s_f = 2$ sensor-specific chunks (three-axis acceleration and three-axis angular velocity); Global Feature Mixing mixes the entire feature axis.

**Figure 8.2:** Dual-View FM's Local and Global Time and Feature Mixing (adapted from [3]).

From a high-level perspective, Dual-View FM takes an input signal $s \in \mathbb{R}^{w \times f}$ along with meta-information about the sensor location $l \in L$ and passes them through a large base and an interchangeable head that outputs the forecast signal $s' \in \mathbb{R}^{w' \times f}$ (cf. Figure 8.3). Both components are surrounded by Reversible Instance Normalization (RevIN, [83]), a technique for normalizing and denormalizing input signals per sample. RevIN ensures that the base and head layers treat signals from different sensor locations, manufacturers, and users equally and independently of their input distributions. Additionally, layer normalizations (Norm, [11]) stabilize intermediate signal representations.

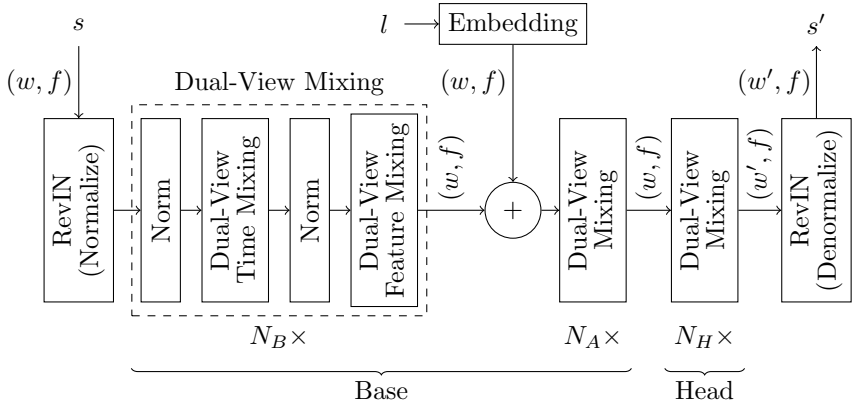



**Figure 8.3:** High-level architecture of Dual-View FM (adapted from [3]). The Dual-View Time Mixing and Dual-View Feature Mixing operations are detailed in Figure 8.2.

The base processes the input signal $s$ with $N_B$ Dual-View Mixing blocks. In parallel, it projects the sensor location $l$ into an embedding of matching shape $(w, f)$. The base then enriches the mixed signal by adding the location embedding (+) and further processes the result with $N_A$ additional Dual-View Mixing blocks that preserve signal shape.

The head applies $N_H$ Dual-View Mixing blocks to further refine the enriched signal. It either outputs a signal $s'$ with unchanged shape $(w, f)$ for signal reconstruction or alters the dimensions to $(w', f)$ for signal forecasting.

## 8.2 Location-Invariant Pre-Training

This work pre-trains Dual-View FM with location-invariant pre-training (cf. Section 7.2). Therefore, the model does not fuse the signals of several positions. Instead, each training sample represents the acceleration and angular velocity at a single body location $l \in L$. This flexible approach enables Dual-View FM to obtain broad knowledge about WHAR signals from a large signal corpus, regardless of the sensor placement on the human body.

Masked Signal Modeling (MSM) and Causal Signal Modeling (CSM) serve as independent pre-training objectives (cf. Section 4.2). Both objectives nullify $n_{\text{mask}}$ time steps in the training samples and train the FM to predict the masked sensor readings. MSM aims for accurate signal reconstruction ($s = s'$), while CSM teaches the model to forecast the next $n_{\text{mask}}$ time steps, producing output signals $s' \in \mathbb{R}^{w' \times f}$, where $w' = n_{\text{mask}}$. This change in the time dimension happens in the head (cf. Figure 8.3); the base always keeps the dimensions unchanged.

This work uses a Leave-One-Dataset-Out approach (LODO, cf. Figure 7.3) to pre-train Dual-View FM on the harmonized signals from eight WHAR datasets (cf. Section 7.1). These datasets cover a wide range of activities, users, sensors, and sensor placements. To ensure the generality of the pre-trained FMs, LODO selects one dataset for testing and splits the seven remaining datasets into $80\,\%$ training and $20\,\%$ validation data, where both sets contain signals from all datasets, activities, and sensor locations.

## 8.3 Location-Invariant Fine-Tuning

The downstream goal of Dual-View FM is signal forecasting. By design, CSM teaches the model to predict $n_{\text{mask}}$ future values. Therefore, CSM models do not require changes to their architecture to fulfill the downstream task. The MSM objective, on the other hand, deviates from the downstream task. To enable signal forecasting with pre-trained MSM models, this work replaces MSM's signal

reconstruction head with a randomly initialized forecasting head that contains $N_H$ Dual-View Mixing blocks that ultimately change the input dimension from $(w, f)$ to $(w', f)$, where $w' = n_{\text{mask}}$.

To fine-tune the models, this work follows the two-step scheme from Section 4.3. Fine-tuning first freezes the weights of the layers in the base and the RevIN layers and retrains only the head weights with a learning rate of 1e-3. Then, it unfreezes all weights and adjusts the whole models using the tiny learning rate 1e-5.

Fine-tuning happens on the eighth dataset which has not been part of the pre-training (cf. Section 8.2). This work splits this dataset into 60 % training, 20 % validation, and 20 % test data. Figure 8.4 illustrates the relationship between the pre-training and fine-tuning data splits. Like pre-training, fine-tuning is location-invariant. As a result, the fine-tuned models analyze signals from different body positions independently.

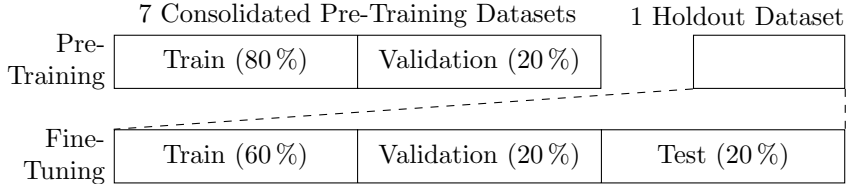



**Figure 8.4:** Relationship between pre-training (top) and fine-tuning (bottom) data splits (adapted from [4]). Pre-training uses seven consolidated datasets for model fitting, while fine-tuning is conducted on the eighth holdout dataset.

## 8.4 Implementation and Evaluation Details

The proposed Dual-View FM architecture is highly configurable, supporting various hyperparameters and pre-training objectives. To assess its accuracy in signal forecasting tasks, this work pre-trains several model configurations on seven benchmark datasets and fine-tunes them on an additional eighth dataset. This section explains the steps for evaluating the role of location embeddings

(Section 8.4.1) and provides details about hyperparameter configurations (Section 8.4.2) as well as the setups for pre-training and fine-tuning with different objectives (Section 8.4.3).

### 8.4.1 Relevance of Location Embeddings

To evaluate the importance of location information for signal forecasting with location-invariant FMs, this work compares location-aware (LA) models, which incorporate location embeddings (cf. Figure 8.3), with location-unaware models (LU) that omit the location input $l$. Accordingly, both the MSM and CSM models are trained in two configurations: with and without location information. The respective setup applies to both pre-training and fine-tuning.

### 8.4.2 Hyperparameters

Dual-View FM contains several customizable hyperparameters, including the number of Dual-View Mixing blocks in base ($N_B$, $N_A$) and head ($N_H$), the number of splits for local time and feature mixing ($s_t$, $s_f$), the hidden dimension $h$, and the dropout factor $d$, which is only applied to the base layers. The number of time steps to be masked ($n_{\text{mask}}$) is also customizable. This work uses the hyperparameter configuration from Table 8.1. These hyperparameters ensure that CSM and MSM models contain roughly the same number of trainable parameters during pre-training (454,615 parameters for CSM, 451,673 parameters for MSM).

The MSM and CSM configurations only differ in the number of Dual-View Mixing blocks in their respective heads. CSM's signal forecasting head uses $N_H = 3$ blocks, while MSM's signal reconstruction head uses $N_H = 1$ block.

Each Dual-View Mixing block in the forecasting head halves the width of the time axis. Given a signal window with shape $(w, f) = (128, 6)$ (cf. Section 7.1.2), the first block outputs $w' = w/2 = 64$ time steps, the second block outputs $w'' = w'/2 = 32$ time steps, and the third block outputs $w''' = w''/2 = 16 = n_{\text{mask}}$

**Table 8.1:** Model configuration (adapted from [3]).

| Parameter | CSM | MSM |
|---|---|---|
| $N_B$ | 2 | 2 |
| $N_A$ | 2 | 2 |
| $N_H$ | 3 | 1 |
| $s_t$ | 4 | 4 |
| $s_f$ | 2 | 2 |
| $h$ | cf. Equation 8.1 | |
| $d$ | 20 % | 20 % |
| $n_{\text{mask}}$ | 16 | 16 |

time steps. The Dual-View Mixing blocks in the reconstruction head and the base, on the other hand, output signal representations whose shapes match their input shapes.

The hidden dimension $h$ defines the projection dimension of the FC1's in the MLPs. This work uses the following rule to select the hidden dimension $h_i$ of the current Dual-View Mixing block $i$ based on the output length $w_{i-1}$ of its predecessor:

$$h_i = 2w_{i-1}, \tag{8.1}$$

where $w_0 = w = 128$.

### 8.4.3 Training Setup

The training parameters differ between pre-training and fine-tuning (Table 8.2). Both training steps repeat the training and data splitting three times with varying random seeds and leverage early stopping with a patience of 10 to reduce overfitting. To account for different training set sizes and training objectives, the batch sizes and learning rates differ between pre-training and fine-tuning.

**Table 8.2:** Pre-training and fine-tuning parameters of Dual-View FM.

| Parameter | Pre-Training | Fine-Tuning |
|---|---|---|
| Batch Size | 256 | $\{32, 128\}$ |
| Patience | 10 | 10 |
| Learning Rate | 1e-3 | $\{$1e-3, 1e-5$\}$ |
| Seeds | $\{1, 2, 3\}$ | $\{1, 2, 3\}$ |

Pre-training uses a batch size of 256 to facilitate training on the large pre-training corpus (cf. Section 8.2). Fine-tuning uses batch size 32 if the fine-tuning dataset contains less than 100k windows (cf. Table 7.1) or 128 if it is larger.

During pre-training and during fine-tuning with fixed base and RevIN weights (cf. Section 8.3), the learning rate is 1e-3. Moreover, this work uses learning rate 1e-5 to further refine the fine-tuned models without fixed weights. Both training steps leverage the Adam optimizer [85].

Dual-View FM leverages a custom loss function that prioritizes key signal characteristics like magnitude and shape [3]:

$$\text{MAE/CD-Loss} = 0.5\,\text{MAE} + 0.5\,\text{CD}, \tag{8.2}$$

where MAE is defined in Equation 2.14 and the Cosine Distance (CD) $\in [0, 2]$ is a reformulation of the cosine similarity (Equation 2.16) as a loss function:

$$\text{CD} = 1 - \text{CS}. \tag{8.3}$$

Smaller CD values indicate greater similarity.

## 8.5 Performance Analysis

This work compares the pre-training (Section 8.5.1) and fine-tuning (Section 8.5.2) performance of Dual-View FM with TSMixer [26], a Transformer architecture [149], and TimeGPT-1 [48]. TSMixer analyzes time-series separately in their time and feature dimensions but adopts a single-view approach that only considers global relationships within signals (cf. Figure 8.1). Therefore, comparing Dual-View FM with TSMixer offers valuable insights into the effectiveness of the proposed dual-view approach (cf. Figure 8.2). Transformers, on the other hand, excel at capturing complex relationships within time-series and are the core of many recent FMs, such as GPT [130], BERT [39], or TimeGPT-1 [48]. Thus, Transformers provide a strong baseline for evaluating WHAR-FMs. TimeGPT-1 is a closed-source, production-ready FM for general time-series forecasting that leverages a Transformer architecture [48, 120]. TimeGPT-1 has been pre-trained on time-series from several domains, including finance, health care, sensor data from the internet of things, and more [48].

To ensure consistency between the models, TSMixer and the Transformer model use the same high-level architecture as Dual-View FM (cf. Figure 8.3) but replace the Dual-View Mixing blocks with Mixer Layers (cf. Figure 8.1) or Transformer Encoders [149], respectively. Appendix A.1 provides details on the architectures and specifications of the comparison models. With 98,212 (MSM) and 83,764 (CSM) weights, the pre-trained TSMixer is roughly five times smaller than Dual-View FM. With 447,950 (CSM) and 508,318 (MSM) trainable parameters, the pre-trained Transformer models have a similar size as the proposed Dual-View FM. The comparison models also use the pre-training and fine-tuning policies from Sections 8.2 and 8.3.

### 8.5.1 Pre-Training

This work evaluates the pre-training performance of location-aware (LA) and location-unaware (LU) Dual-View FMs (cf. Section 8.4.1), LU TSMixer, and LA

Transformer using a Leave-One-Dataset-Out approach (cf. Section 7.3). The models are repeatedly pre-trained on seven pre-training datasets with varying random seeds (cf. Table 8.1). Sections 8.5.1.1 and 8.5.1.2 analyze the performance of the pre-trained CSM and MSM models separately. The presented scores are the Mean Absolute Errors (MAEs, Equation 2.14) and Root Mean Squared Errors (RMSEs, Equation 2.15) on an eighth holdout dataset, averaged across three random seeds. Additionally, Section 8.5.1.3 investigates the impact of different amounts of pre-training data on Dual-View FM's generalizability.

#### 8.5.1.1 Causal Signal Modeling (CSM)

Table 8.3 presents the average MAEs and RMSEs of the Dual-View FMs, LU TSMixer, and LA Transformer for the CSM pre-training task and compares them with TimeGPT-1 on an eighth holdout dataset.

The location-unaware Dual-View FMs achieve the best overall performance on the CSM task, outperforming their location-aware counterparts in 7 out of 8 holdout datasets. Although the performance differences are relatively small, they are consistent across these datasets. These results suggest that incorporating location information into the model architecture has only a minor effect on the CSM performance of pre-trained Dual-View FMs.

TSMixer ranks third on the CSM task, with MAEs and RMSEs $7.21\,\%$ and $5.76\,\%$ higher than those of the LU Dual-View FMs. These results demonstrate the clear benefit of incorporating local views alongside TSMixer's global mixing strategy in fully-connected forecasting models.

The Transformer performs considerably worse than both Dual-View FMs and TSMixer. Nevertheless, its improvements in MAE over TimeGPT-1, which has been pre-trained on a much broader and more versatile dataset, underscore the value of domain-specific pre-training data in time-series forecasting tasks.

Both Dual-View FMs consistently outperform TimeGPT-1 in MAE across all holdout datasets. On average, the LU Dual-View FMs achieve improvements of

**Table 8.3:** Average pre-training performance of the location-unaware (LU) Dual-View FMs, their location-aware (LA) counterparts, TSMixer, Transformer, and TimeGPT-1 [48] on the CSM pre-training task; **best**, 2nd best, *3rd best* scores per holdout dataset (adapted from [3]). The last two rows contain the average scores over all datasets and the deltas between the best models and the other models.

**(a)** Mean Absolute Error (MAE).

| Dataset | LU | LA | TSMixer | Transformer | TimeGPT-1 |
|---|---|---|---|---|---|
| Complex HAD | **0.451** | 0.466 | *0.488* | 0.521 | 0.783 |
| DLA | 0.458 | **0.455** | *0.505* | 0.544 | 0.788 |
| DSADS | **0.590** | 0.592 | *0.632* | 0.670 | 0.958 |
| FLAAP | 0.578 | *0.581* | **0.576** | 0.649 | 0.700 |
| PAMAP2 | **0.666** | 0.668 | *0.718* | 0.865 | 1.031 |
| RealWorld (HAR) | **0.654** | 0.659 | *0.698* | 0.939 | 1.069 |
| USC-HAD | **0.592** | 0.595 | 0.656 | *0.632* | 1.015 |
| mHealth | **0.749** | 0.750 | *0.830* | 0.927 | 1.082 |
| Average | **0.592** | 0.596 | *0.638* | 0.718 | 0.928 |
| Delta | - | 0.004 | 0.046 | 0.126 | 0.336 |

**(b)** Root Mean Squared Error (RMSE).

| Dataset | LU | LA | TSMixer | Transformer | TimeGPT-1 |
|---|---|---|---|---|---|
| Complex HAD | **1.094** | 1.126 | *1.176* | 1.212 | 1.283 |
| DLA | *2.110* | 2.097 | 2.239 | 2.323 | **1.815** |
| DSADS | **1.588** | 1.594 | 1.676 | 1.708 | *1.674* |
| FLAAP | 1.414 | *1.415* | **1.399** | 1.552 | 1.514 |
| PAMAP2 | **2.121** | 2.123 | 2.244 | 2.505 | *2.173* |
| RealWorld (HAR) | **1.609** | 1.618 | *1.692* | 2.085 | 1.939 |
| USC-HAD | **1.508** | 1.514 | 1.664 | *1.572* | 2.351 |
| mHealth | **2.152** | 2.156 | *2.343* | 2.683 | 2.917 |
| Average | **1.700** | 1.705 | *1.804* | 1.955 | 1.958 |
| Delta | - | 0.005 | 0.104 | 0.255 | 0.258 |

0.336 in MAE (36.21 %) and 0.258 in RMSE (13.18 %). The largest performance gains are observed on the USC-HAD dataset, where the LU Dual-View FMs outperform TimeGPT-1 by 0.423 and 0.843 in MAE and RMSE, respectively. These results highlight the critical role of domain-specific FMs for signal forecasting.

Overall, the CSM results show that fully-connected models with both global and local time and feature mixing operations outperform models with global views only, as well as Transformer-based architectures, including the state-of-the-art TimeGPT-1, on signal forecasting tasks.

#### 8.5.1.2 Masked Signal Modeling (MSM)

Table 8.4 compares the average MAEs and RMSEs of both Dual-View FMs, the LU TSMixer, and the LA Transformer for the MSM objective. MSM pre-trains the models for signal reconstruction and data imputation. Therefore, the closed-source TimeGPT-1, which is specifically designed for signal forecasting, cannot serve as a comparison model for the MSM task.

All models accurately reconstruct sensor signals with relatively small errors. In general, the MAEs and RMSEs observed in the MSM task are lower than those in the CSM task (cf. Table 8.3), indicating that reconstructing signals with missing features is less challenging than forecasting based on historical data.

The Transformer achieves the best overall performance, with the Dual-View FMs trailing by 0.1 points in MAE and 0.531 points in RMSE. Nevertheless, the Dual-View FMs still outperform TSMixer by up to 0.086 points in MAE and 0.224 points in RMSE, which underscores the benefit of incorporating both local and global time and feature perspectives in the processing.

Consistent with the findings from the CSM task, the location-unaware Dual-View FMs slightly surpass their location-aware counterparts on average. These results further demonstrate that meta-information about sensor placement on the human body has negligible impact on the pre-training performance of Dual-View FMs.

**Table 8.4:** Average pre-training performance of the location-unaware (LU) Dual-View FMs, their location-aware (LA) counterparts, TSMixer, and Transformer on the MSM pre-training task; **best**, 2nd best, *3rd best* scores per holdout dataset (adapted from [3]). The last two rows contain the average scores over all datasets and the deltas between the best models and the other models.

**(a)** Mean Absolute Error (MAE).

| Dataset | LU | LA | TSMixer | Transformer |
|---|---|---|---|---|
| Complex HAD | *0.153* | 0.147 | 0.265 | **0.040** |
| DLA | 0.177 | *0.200* | 0.263 | **0.050** |
| DSADS | *0.159* | 0.137 | 0.247 | **0.069** |
| FLAAP | *0.128* | 0.122 | 0.178 | **0.033** |
| PAMAP2 | 0.192 | *0.209* | 0.276 | **0.080** |
| RealWorld (HAR) | 0.182 | *0.190* | 0.272 | **0.074** |
| USC-HAD | 0.180 | *0.186* | 0.240 | **0.052** |
| mHealth | 0.244 | *0.245* | 0.366 | **0.216** |
| Average | 0.177 | *0.180* | 0.263 | **0.077** |
| Delta | 0.100 | 0.103 | 0.186 | - |

**(b)** Root Mean Squared Error (RMSE).

| Dataset | LU | LA | TSMixer | Transformer |
|---|---|---|---|---|
| Complex HAD | 0.867 | 0.821 | 1.174 | **0.297** |
| DLA | 1.472 | *1.517* | 1.711 | **0.709** |
| DSADS | *0.975* | 0.918 | 1.247 | **0.349** |
| FLAAP | *0.756* | 0.718 | 0.861 | **0.283** |
| PAMAP2 | *1.484* | 1.461 | 1.671 | **0.755** |
| RealWorld (HAR) | 0.998 | *1.050* | 1.171 | **0.430** |
| USC-HAD | 1.043 | *1.054* | 1.195 | **0.425** |
| mHealth | 1.481 | **1.457** | 1.759 | *1.507* |
| Average | *1.134* | 1.125 | 1.349 | **0.594** |
| Delta | 0.540 | 0.531 | 0.755 | - |

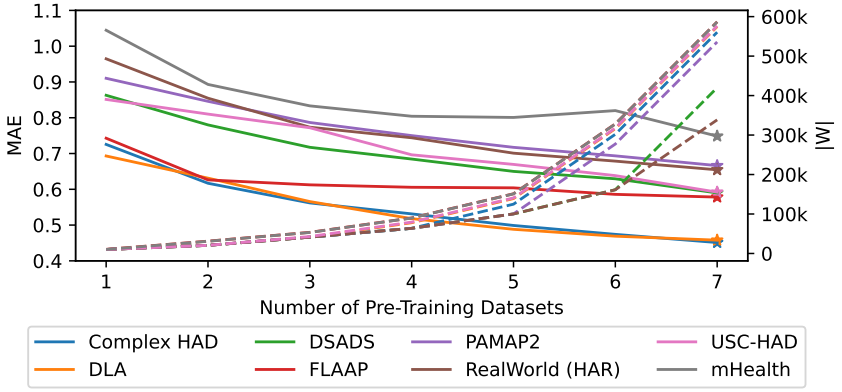
#### 8.5.1.3 Dataset Size

To assess the impact of training data volume on the generalizability of Dual-View FM, this work sorts the pre-training datasets by their number of windows |W| in ascending order (cf. Table 7.1), successively pre-trains the location-unaware model with increasing numbers of datasets, and analyzes the resulting changes in MAE on the holdout datasets. Figure 8.5 plots the MAEs (solid lines) of the Dual-View FMs for both pre-training objectives per holdout dataset against the number of pre-training datasets (horizontal axis) and the corresponding total number of training samples (dashed lines). Stars mark the lowest MAE per holdout dataset, representing the best performing data configurations.

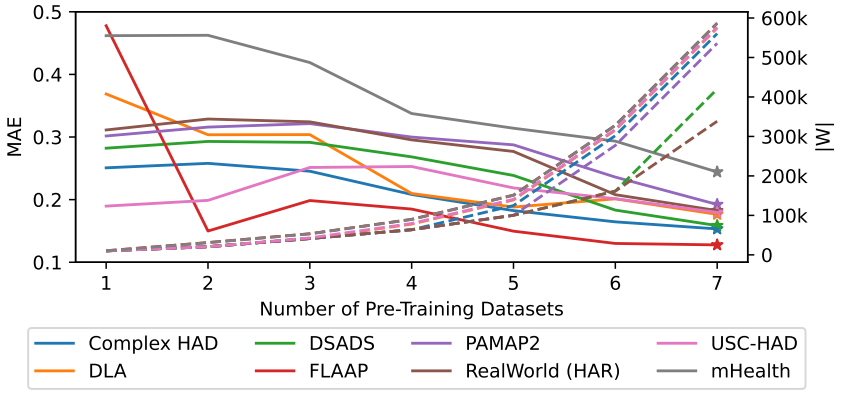
Despite differences in sensing setups between the training and test datasets, the generalizability of the models on unseen holdout datasets consistently improves with increasing amounts of pre-training data. The best performing models are those trained on all seven pre-training datasets (cf. Tables 8.3 and 8.4). The CSM models achieve an average MAE improvement of 0.257 when comparing full pre-training on all seven pre-training datasets to training on the smallest datasets only, with RealWorld (HAR) showing the largest absolute gain of 0.31. For the MSM objective, the average MAE improvement is 0.154. Here, the FLAAP datasets achieves the most pronounced gain of 0.35.

Some performance drops appear as the number of pre-training datasets increases. The most notable example for CSM is mHealth, where the addition of the sixth pre-training dataset (here: DSADS) worsens the MAE by 0.019 points. For MSM, the holdout datasets that observe the largest drops are USC-HAD and FLAAP, where adding the third pre-training dataset (here: Complex HAD) worsens the MAE by 0.053 and 0.049, respectively. FLAAP's performance recovers from this misery after adding the fourth and fifth dataset. USC-HAD, however, requires all seven pre-training datasets to finally achieve improvements over pre-training on the smallest dataset only.

These results confirm that additional pre-training data improves the generalizability of Dual-View FM on unseen holdout datasets even when the signals originate




**(a)** CSM (adapted from [3]).




**(b)** MSM.

**Figure 8.5:** MAEs of Dual-View FMs on unseen holdout datasets plotted against the number of pre-training datasets (1 to 7) sorted by size. The forecasting performance improves as the number of pre-training datasets increases. Solid lines represent MAEs for CSM and MSM objectives, while dashed lines indicate the total training samples. Stars denote the optimal data configurations with the lowest MAEs.

from different sensing setups. However, such cross-domain differences can significantly impact model generalization when training data is limited. This limitation can be effectively mitigated by scaling up the volume and diversity of pre-training data.

### 8.5.2 Fine-Tuning

Fine-tuning adapts the pre-trained models to the forecasting of the next $n_{\text{mask}}$ time steps using their respective holdout datasets. Section 8.5.2.1 compares the performance of fine-tuned Dual-View FMs against fine-tuned TSMixers and Transformers with different pre-training objectives. To resolve the mismatch between the MSM objective and the forecasting task, the reconstruction heads of the MSM models are replaced with freshly initialized forecasting heads identical to those used by the CSM models. Additionally, Section 8.5.2.2 analyzes the forecasting performance at the activity level, offering insights into the unique challenges associated with signals from different activity types. TimeGPT-1 is excluded from this evaluation as this closed-source model cannot be fine-tuned locally with this work's custom loss function (cf. Equation 8.2).

#### 8.5.2.1 Comparison with State-of-the-Art Forecasters

Table 8.5 compares the average MAEs and RMSEs of the fine-tuned models across all holdout datasets per pre-training objective. With average MAEs of 0.54 and average RMSEs of 1.598, the Dual-View FMs with CSM objective generate signal forecasts with the highest quality. Both variants outperform the best TSMixer by approximately $7.06\,\%$ (MAE) and $6\,\%$ (RMSE) and the best Transformer by $12.34\,\%$ (MAE) and $7.31\,\%$ (RMSE).

In general, the CSM models outperform their MSM counterparts, which is expected as the CSM pre-training objective already matches the fine-tuning goal. Despite the gap between MSM and the fine-tuning task, both Dual-View FMs with MSM objective achieve similar scores as TSMixer with CSM objective and even

**Table 8.5:** Average fine-tuning performances across all holdout dataset; **best**, 2nd best, *3rd best* scores (adapted from [3]).

| Pre-Training Objective | Metric | Dual-View FM | | TSMixer | Transformer |
|---|---|---|---|---|---|
| | | LU | LA | LU | LA |
| CSM | MAE | **0.540** | **0.540** | *0.581* | 0.616 |
| | RMSE | **1.598** | **1.598** | *1.700* | 1.724 |
| MSM | MAE | 0.601 | 0.602 | 0.703 | 0.781 |
| | RMSE | 1.716 | 1.718 | 1.957 | 2.178 |

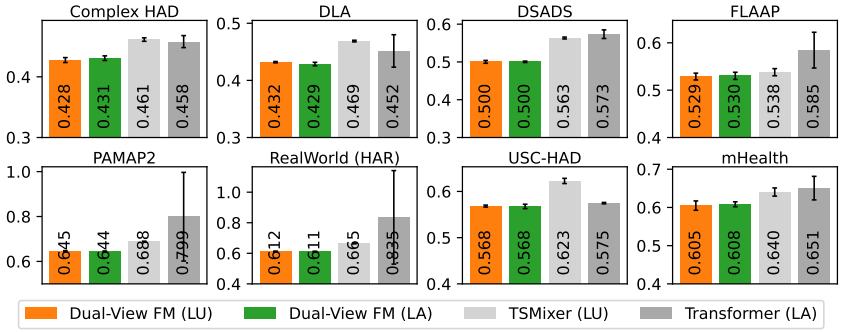
surpass the Transformers in MAE and RMSE. These results underscore the great signal forecasting capabilities of Dual-View FM and emphasize the importance of pre-training objectives that suit the downstream use case.

A comparison between the fine-tuning results and the CSM pre-training scores from Table 8.3 reveals that fine-tuning improves the MAEs and RMSEs of the location-unaware Dual-View FMs by an average of 0.052 and 0.102 points, corresponding to relative gains of $8.78\,\%$ and $6\,\%$, respectively. These improvements highlight the importance of context-specific fine-tuning to optimally handle the diversity of sensor signals across different sensing setups.

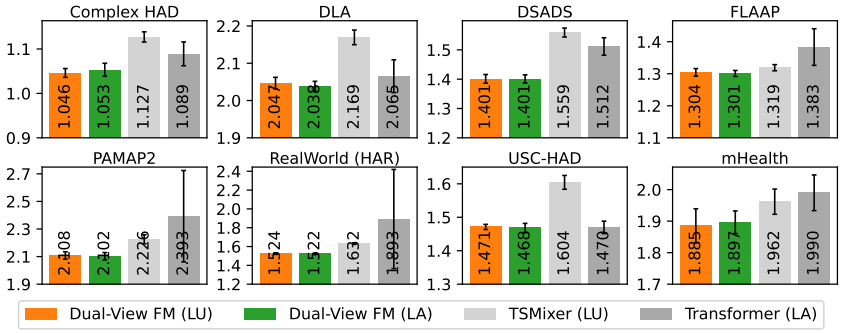
Figure 8.6 compares the MAEs and RMSEs of the fine-tuned models with CSM pre-training objective per dataset as bar charts. These charts confirm the findings from Table 8.5: Dual-View FM forecasts signals with higher accuracy than TSMixer or Transformer across all datasets.

#### 8.5.2.2 Activity-Level Performance

As discussed in Section 7.1.1, the eight datasets share 40 common activity labels. Figure 8.7 illustrates the average MAEs and RMSEs of the fine-tuned, location-unaware Dual-View FMs with CSM objective for each activity. These metrics are averaged across all test datasets.




**(a)** MAE (adapted from [3]).




**(b)** RMSE.

**Figure 8.6:** Comparison of the fine-tuned CSM models per dataset and metric.

The performance metrics per activity correlate with the variability in their signals. Static activities with small standard deviations in their signals are generally easier to forecast than dynamic activities with large or spontaneous changes. Basketball, downstairs, jump, nordic walking, run, soccer, and upstairs are the activities with the largest MAEs and RMSEs. With standard deviations between 1.7 and 4.7, their signals are also among those with the highest variabilities. Thus, activities with larger signal variances are harder to forecast.

Figure 8.8 compares true walking and running signals with the forecasts of a fine-tuned, location-unaware Dual-View FM with CSM pre-training objective. These

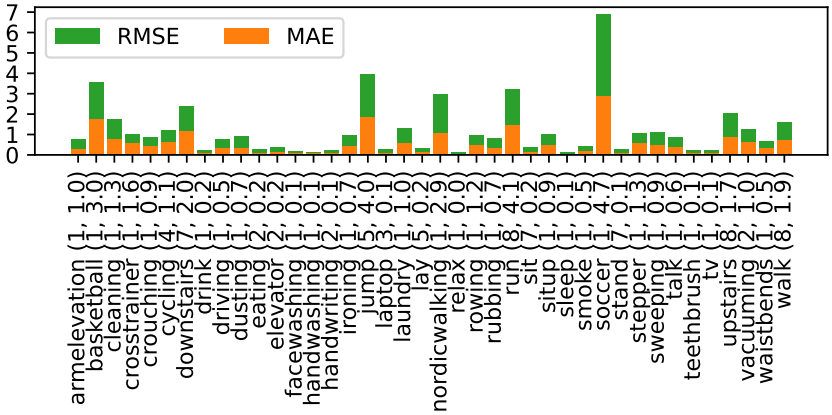


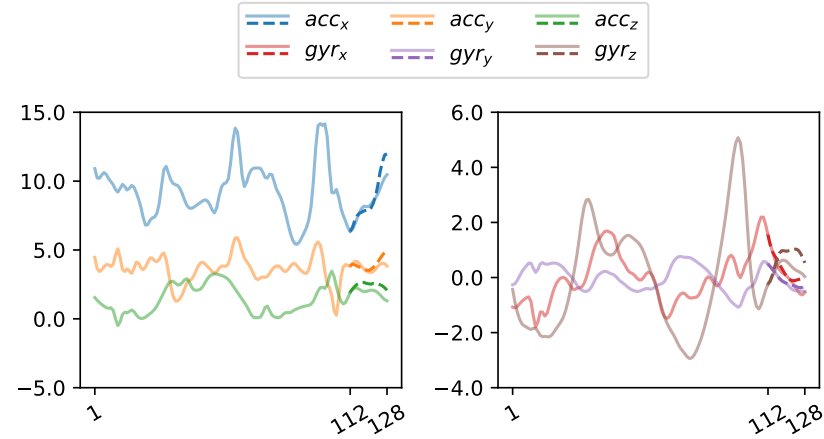

**Figure 8.7:** Average MAEs and RMSEs of the fine-tuned, location-unaware Dual-View FMs with CSM objective per general activity $g$ (adapted from [3]). The labels on the horizontal axis are "$g$ (number of datasets that contain $g$, standard deviation of $g$'s signals)".

examples represent unseen test samples with median forecast loss within their respective activity classes. The plots confirm that the forecasts closely align with the true signal curves.

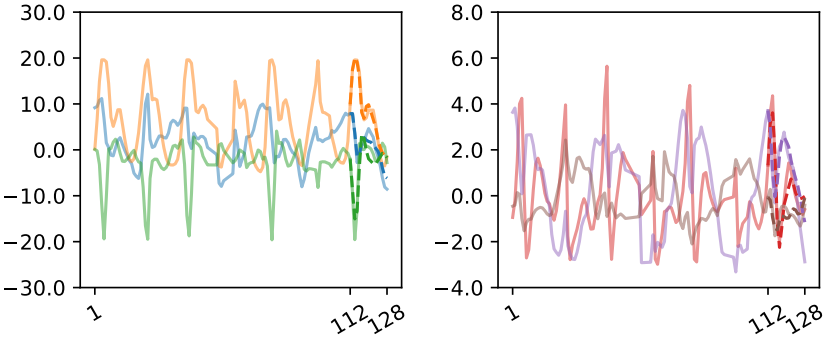
## 8.6 Summary

This chapter introduced Dual-View FM, a location-invariant foundation model that forecasts sensor signals by capturing both short- and long-term dependencies within signal windows through local and global mixing operations. The model leverages pre-training on a diverse corpus of seven consolidated datasets with varying characteristics and applies fine-tuning on an independent eighth holdout dataset. Dual-View FM consistently outperforms state-of-the-art models in time-series forecasting tasks. Increasing the size of the pre-training corpus improves its generalizability and robustness on unseen data. Fine-tuning enhances the model's accuracy even further. However, forecasting signals with abrupt changes or large deviations remains more challenging than predicting static activities. Although

signal patterns differ across body locations, incorporating metadata about sensor placement into the training process does not improve forecasting performance.




**(a)** Walk (location: wrist, test loss: 0.18).




**(b)** Run (location: chest, test loss: 0.51).

**Figure 8.8:** Example forecasts of unseen walking and running signals from the fine-tuned, location-unaware Dual-View FMs with CSM objective. The left plots show true three-dimensional accelerations (acc, semi-transparent) and their forecasts (dashed), while the right plots show true three-dimensional angular velocities (gyr, semi-transparent) and their forecasts (dashed). The selected signals represent test samples with median forecast loss within their respective activity classes.

# 9 Activity Classification with Foundation Models

Activity classification is arguably the most important task in WHAR. Beyond achieving high recognition rates across a broad range of activities, WHAR classifiers must also meet strict device constraints to be viable for wearable deployment. FMs, however, often rely on large numbers of trainable parameters [18]. Even Dual-View FM, which has demonstrated the great potential of WHAR-FMs for signal forecasting, uses approximately 450k parameters (cf. Chapter 8). This raises the question of how WHAR-FMs can be further reduced in size to support practical use in wearable systems.

To address this challenge, this chapter introduces MRWA-FM, the Multi-Resolution Wavelet-Attention Foundation Model for human activity classification from *Enhancing Sensor-Based Human Activity Recognition With Multiresolution Wavelet-Attention* [4]. This second FM architecture combines deep learning principles with the Discrete Wavelet Transform (DWT, cf. Section 2.2), a signal processing technique which is well-suited for capturing abrupt changes in human activity signals via time-localized signal analysis. However, instead of using literature-defined Wavelets, MRWA-FM learns task-specific Wavelet filters that enable the extraction of characteristic features of human motion.

To enhance the flexibility of the learned Wavelets and to compensate for limited training data, MRWA-FM uses the same location-invariant pre-training strategy as Dual-View FM (cf. Sections 7.2 and 8.2). Building on evidence that multi-sensor fusion improves classification performance [122, 158], this chapter also presents a lightweight fine-tuning approach for multi-location downstream classification.

A comprehensive evaluation demonstrates the effectiveness of MRWA-FM as a lightweight feature extractor for human activity classification that improves the performance of three state-of-the-art downstream classifiers.

## 9.1 Model Design

Figure 9.1 presents MRWA-FM's architecture, which enhances input signals $s$ via Multi-Resolution Wavelet-Attention. MRWA-FM incorporates learnable Wavelet filters with multi-head cross-attention between $s$ and their Wavelet coefficients to automatically detect and amplify important signal components. This architecture provides two processing paths with shared layer weights: the location-invariant pre-training path and the multi-location fine-tuning path.

### Pre-Training Path.

Location-invariant pre-training (Figure 9.1a) learns to reconstruct single-location signals $s \in \mathbb{R}^{w \times f}$ by correlating $s$ with their Wavelet coefficients as follows.

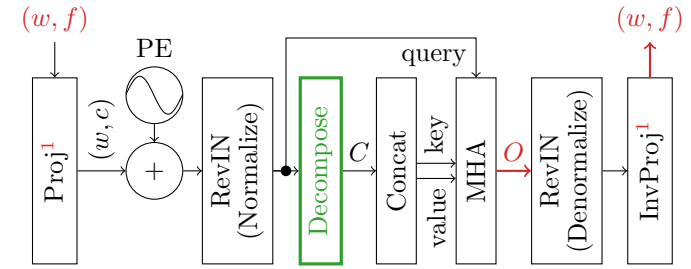
The projection layer Proj[1] is a fully-connected layer that adjusts the $f$-dimension of $s$ to $c$ hidden channels according to the equation

$$s' = sW^{1,\,\text{Proj}} + b^{1,\,\text{Proj}}, \tag{9.1}$$

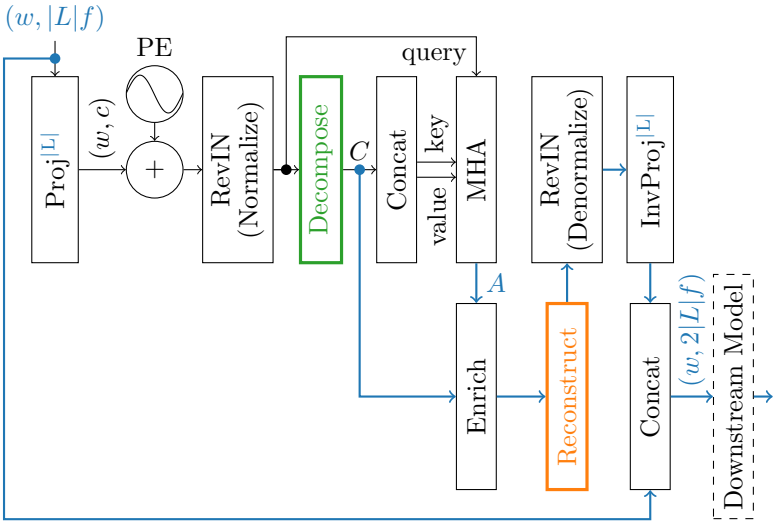
where $W^{1,\,\text{Proj}} \in \mathbb{R}^{f \times c}$ is a learnable weight matrix, $b^{1,\,\text{Proj}} \in \mathbb{R}^{c}$ is a learnable bias vector, and $s' \in \mathbb{R}^{w \times c}$ is the modified hidden representation. This simplified equation omits the activation function of the layer and the input batch size.

Positional Encodings (PE, [149]) capture the sequential order of the hidden representation $(w, c)$ in the $w$-dimension; RevIN [83] normalizes the hidden representations per sample.

Decompose applies a $J$-level Wavelet decomposition (cf. Section 2.2) to $(w, c)$ that produces $J+1$ Wavelet coefficients $C = (cD_1, cD_2, \ldots, cD_J, cA_J)$ (cf. Figure 9.2). The detail coefficients $cD_j$ with $j \in \{1, \ldots, J\}$ have shapes $(w/2^j, c)$;




**(a)** Location-invariant pre-training path (adapted from [4]).




**(b)** Multi-location fine-tuning path with a downstream model (adapted from [4]).

**Figure 9.1:** Architecture of MRWA-FM (adapted from [4]). The data propagation differs between location-invariant pre-training and multi-location fine-tuning.

the approximation coefficients $cA_J$ have shape $(w/2^J, c)$. Concat concatenates the coefficients along their first axes into a single tensor $(w, c)$, which serves as both the key and the value for the subsequent multi-head attention layer (MHA).

MHA applies cross-attention between these key-value pairs and the normalized input signals, enabling the model to identify patterns and relationships between the hidden representations of the input signals and their time-frequency bands.

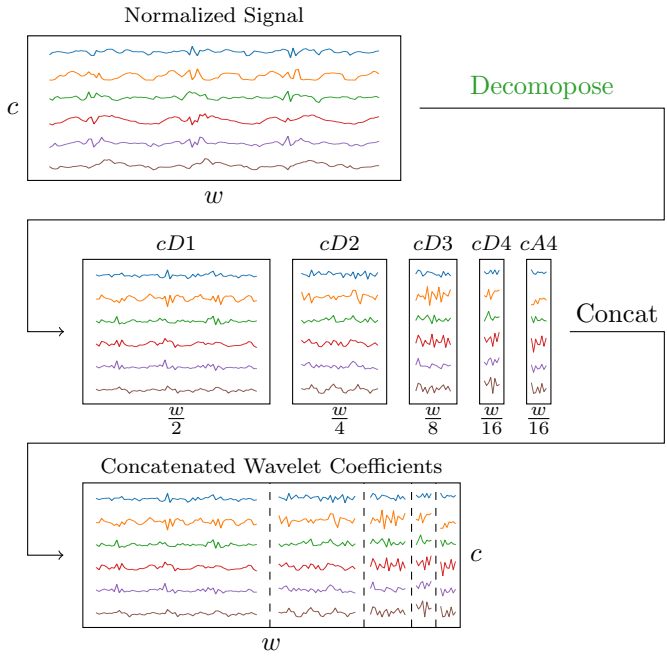



**Figure 9.2:** Decomposition of a normalized signal from [138] into Wavelet coefficients ($cD1$, $cD2$, $cD3$, $cD4$, and $cA4$) using a learned, four-level Wavelet transform, followed by concatenation of the coefficients (cf. Decompose and first Concat layer in Figure 9.1).

MHA has two outputs: the actual output tensor $O \in \mathbb{R}^{w \times c}$ and the attention scores $A \in \mathbb{R}^{n_h \times w \times w}$, where $n_h$ is the number of attention heads. During pre-training, the model passes $O$ to the denormalization layer.

The inverse projection layer InvProj[1] is another fully-connected layer that changes the hidden feature dimension $c$ back to $f$ according to the equation

$$s = s' W^{1,\,\text{InvProj}} + b^{1,\,\text{InvProj}}, \tag{9.2}$$

where $W^{1,\,\text{InvProj}} \in \mathbb{R}^{c \times f}$ is a learnable weight matrix and $b^{1,\,\text{InvProj}} \in \mathbb{R}^{f}$ is a learnable bias vector.

### Fine-Tuning Path.

Multi-location fine-tuning (Figure 9.1b) leverages the pre-learned correlations to emphasize important signal components in multi-location signals $\hat{s} \in \mathbb{R}^{w \times |L|f}$. Both the switch from location-invariant to multi-location processing and the signal enrichment require certain modifications to the processing path.

The Proj$^{|L|}$ layer enables the processing of multi-location signals $\hat{s}$. This layer replaces $W^{1,\,\text{Proj}}$ in Equation 9.1 with a multi-location weight matrix

$$W^{|L|,\,\text{Proj}}_{i,j} = W^{1,\,\text{Proj}}_{i \bmod |L|,j} \in \mathbb{R}^{|L|f \times c}, \quad i,j \in \{1,\ldots,|L|f\},\{1,\ldots,c\} \tag{9.3}$$

that repeats $W^{1,\,\text{Proj}}$ $|L|$ times along its $f$-dimension. With this modification, Proj$^{|L|}$ fuses signals from $|L|$ locations into the same $(w,c)$-shaped hidden space as Proj$^{1}$, while re-using the pre-trained weights at each location. Since fully-connected layers sum weighted inputs from all neurons in the previous layer, the multi-location setup changes the magnitude of the hidden space. However, the RevIN layers counteract this magnitude change as they normalize the hidden representations per sample.

The Enrich layer weights the decomposed Wavelet coefficients $C$ according to the attention scores $A$ as illustrated in Figure 9.3 and passes the enriched coefficients to the Wavelet reconstruction layer. The enrichment steps are as follows:

(E1) Compute the average $\overline{A} \in \mathbb{R}^{w \times w}$ of $A \in \mathbb{R}^{n_h \times w \times w}$ over $n_h$:

$$\overline{A} = \frac{1}{n_h} \sum_{i=1}^{n_h} A_i. \tag{9.4}$$

$\overline{A}$'s first $w$-axis represents the time steps of the MHA query (the normalized input signals), while its second $w$-axis represents specific positions in the MHA keys (the Wavelet coefficients).

(E2) Partition the second $w$-axis of $\overline{A}$ into $J+1$ segments. The first $J$ segments $\overline{A}_j \in \mathbb{R}^{w \times \frac{w}{2^j}}$ with $j \in \{1,\ldots,J\}$ represent the detail coefficients

$cD_j$. The final segment $\overline{A}_{J+1} \in \mathbb{R}^{w \times \frac{w}{2^J}}$ corresponds to the approximation coefficients $cA_J$.

(E3) Aggregate each $\overline{A}_i$ with an aggregation function agg $\in \{\max, \text{mean}, \text{sum}\}$ to obtain scalars $a_i \in \mathbb{R}$:

$$a_i = \text{agg}(\overline{A}_i), \quad i \in \{1, \ldots, J+1\}. \tag{9.5}$$

(E4) Weight the Wavelet coefficients $c_i \in C$ by normalized $a_i$:

$$c_i' = c_i a_i \frac{J+1}{\sum_{k=1}^{J+1} a_k}, \quad i \in \{1, \ldots, J+1\}. \tag{9.6}$$

This normalization ensures that the weighted coefficients $c_i'$ maintain their relative importance across different Wavelet levels.

During fine-tuning, the denormalization layer processes the enriched and reconstructed signal representations instead of the MHA output $O$.

The InvProj$^{|L|}$ converts the $(w, c)$-shaped hidden representations into multi-location outputs. This layer replaces $W^{1,\,\text{InvProj}}$ and $b^{1,\,\text{InvProj}}$ in Equation 9.2 with modified $W^{|L|,\,\text{InvProj}}$ and $b^{|L|}$. To this end, this layer repeats the pre-trained weight matrix $W^{1,\,\text{InvProj}}$ of InvProj$^1$ |L| times along its $f$-dimension:

$$W_{i,j}^{|L|,\,\text{InvProj}} = W_{i,j \bmod |L|}^{1,\,\text{InvProj}} \in \mathbb{R}^{c \times |L| f}, \quad i, j \in \{1, \ldots, c\}, \{1, \ldots, |L| f\}. \tag{9.7}$$

Since InvProj$^1$'s bias term has the length of the output feature dimension $f$, $b^{1,\,\text{InvProj}}$ also requires adjustments to fit the multi-location setting:

$$b_j^{|L|,\,\text{InvProj}} = b_{j \bmod |L|}^{1,\,\text{InvProj}}, \quad j \in \{1, \ldots, |L| f\}. \tag{9.8}$$

Additionally, fine-tuning introduces a residual connection that concatenates the enriched output of InvProj$^{|L|}$ with the multi-location input signal $\hat{s}$ and forwards the concatenated signal to a downstream model. This concatenation preserves the

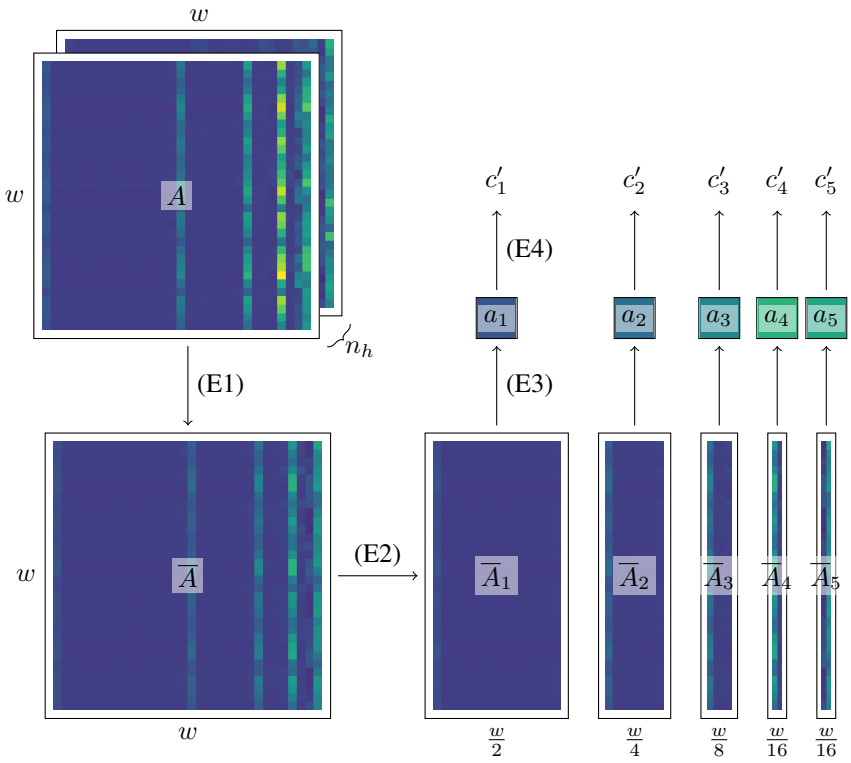



**Figure 9.3:** Internal processing of the attention matrix within the Enrich layer, resulting in the weighted Wavelet coefficients $c'_i$. Here, enrichment step (E3) uses the max aggregation function.

full information from $\hat{s}$, positioning the enriched signal as an additional source of information for the downstream model.

Figure 9.4 illustrates the steps between Wavelet decomposition and reconstruction for a three-level Wavelet transform. The multi-channel Wavelet filters $H$, $G$, $\tilde{H}$, and $\tilde{G}$ contain $c$ filters $h, g, \tilde{h}$, and $\tilde{g}$ (cf. Section 2.2). The decomposition and reconstruction layers implement Wavelet filters as depthwise convolutions [29]. Unlike standard convolutions in neural networks, which combine information across all input channels, depthwise convolutions operate independently on each

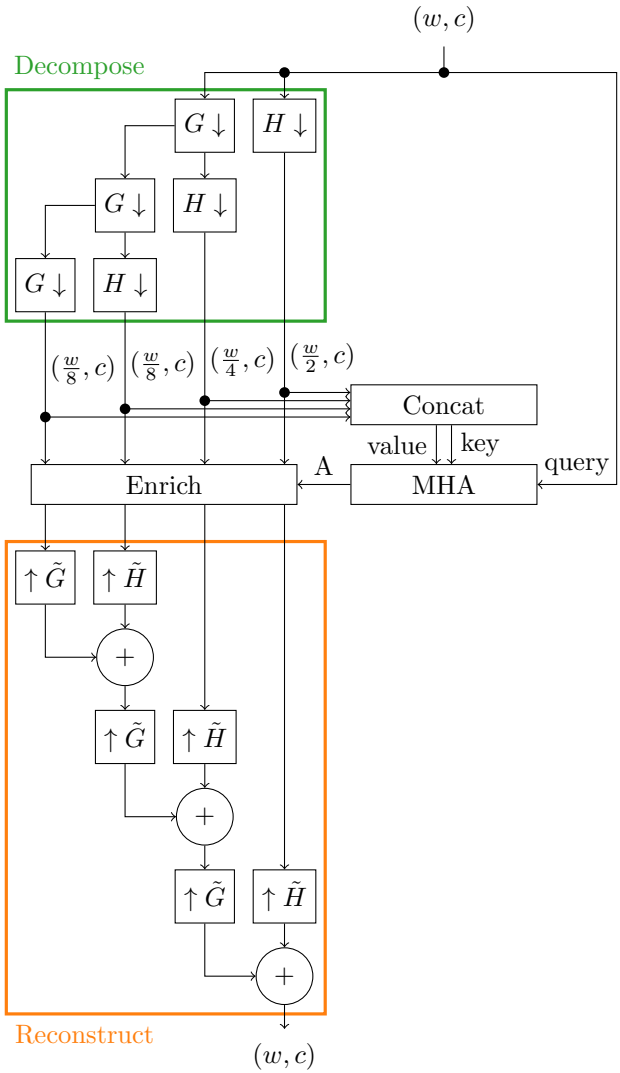



**Figure 9.4:** Illustration of the interactions between MRWA's decomposition, attention, enrichment, and reconstruction layers for a three-level Wavelet transform (adapted from [4]).

channel. This behavior mirrors the DWT, which also processes each channel separately without fusing information between them, making depthwise convolutions a natural fit for simulating Wavelet operations in neural networks.

Figure 9.5 compares the effects of different aggregation functions agg in the enrichment step (E3) on MRWA's outputs for different Wavelet levels $J$ and Wavelet filter sizes $K$. The matrices on the right visualize the strength of the aggregated and normalized attention coefficients $a_i$ for each Wavelet coefficient. For $J = 4$ with max aggregation, the colors of $a_i$ match those shown in Figure 9.3. Since the sum function adds up all values in $\overline{A}_i$, this aggregation function favors longer Wavelet coefficients from lower levels (e.g., $cD_1$, $cD_2$) which represent higher frequency bands. Max aggregation favors higher-level coefficients (e.g., $cD_J$, $cA_J$) which represent lower frequency bands. Mean aggregation has similar tendencies as max, but does not suppress the high-frequency signal components quite as strong. The plots on the left compare the original input signal with signals enriched using different aggregation functions. Since the sum aggregation favors high-frequency components, its enriched signals fluctuate significantly more than the signals that leverage mean or max aggregation during enrichment. Moreover, due to the similarity of the normalized $a_i$ from mean and max aggregations, their enriched signals are also very similar.

## 9.2 Location-Invariant Pre-Training

This work pre-trains MRWA-FM using LODO cross-validation with the same location-invariant 80/20-split of seven pre-training datasets as Dual-View FM (cf. Sections 7.2 and 8.2). The pre-training objective, however, differs from Dual-View FM. MRWA-FM applies the signal reconstruction objective (cf. Section 4.2) to learn WHAR-tailored Wavelet filters and attention weights that allow for the perfect reconstruction of input signals.

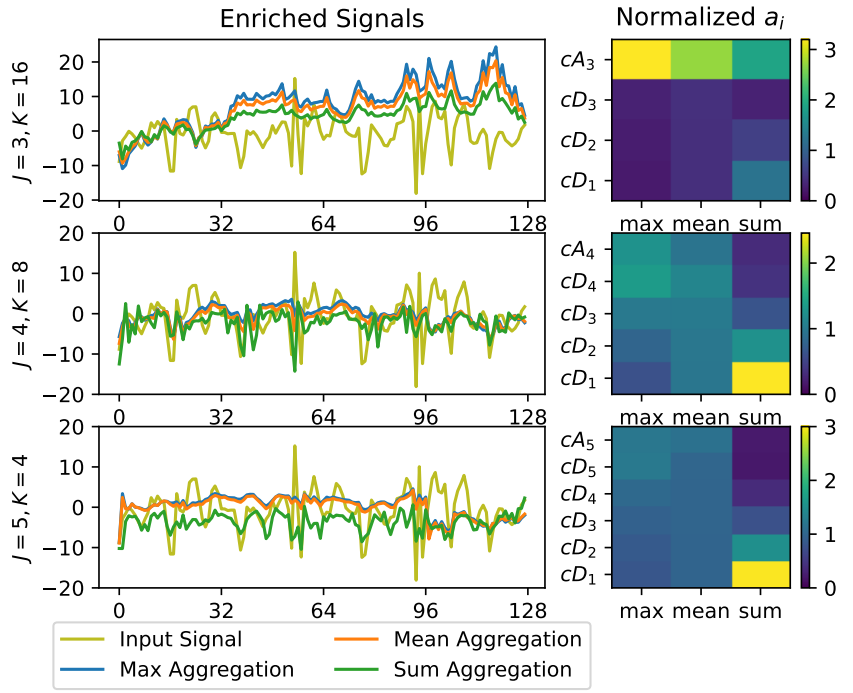



**Figure 9.5:** The effects of different aggregation functions agg $\in$ {max, mean, sum} in the Enrich layer on MRWA-FM's outputs (left) and the corresponding attention coefficients $a_i$ (enrichment step (E3), right) for different Wavelet levels $J$ and filter sizes $K$ (adapted from [4]; signal from [138]). The sum aggregation prefers low-level Wavelet coefficients (e.g., $cD_1$, $cD_2$) and thus, emphasizes high-frequency components in the enriched signals; max and mean aggregations favor high-level coefficients (e.g., $cD_J$, $cA_J$).

## 9.3 Multi-Location Fine-Tuning

The downstream goal of MRWA-FM is activity classification. Hence, the downstream model in Figure 9.1b is an activity classifier. Since neural networks can be seamlessly combined with other neural networks and optimized together as long as their input and output shapes match, this work examines how a preceding MRWA-FM impacts the performance of three state-of-the-art human activity classifiers: TinyHAR [177], DeepConvLSTM [17, 122], and GRU [165].

To this end, this work attaches the downstream classifiers to pre-trained MRWA-FMs and fine-tunes the joint models on the eighth WHAR dataset which has not been part of the pre-training. Fine-tuning uses the same 60/20/20-split on this eighth dataset as Dual-View FM (cf. Section 8.3). However, since human activity classification benefits significantly from incorporating sensor signals from multiple on-body locations [122, 158], MRWA-FM performs multi-location fine-tuning. This process integrates signals from all sensor locations available in the eighth dataset to enable comprehensive activity classification (cf. fine-tuning path in Section 9.1).

During fine-tuning, this work freezes all pre-trained MRWA-FM layers and only trains the downstream classifiers with a learning rate of 1e-3. For comparison, this work also retrains the InvProj and MHA layers to analyze their impact on overall classification performance.

## 9.4 Implementation and Evaluation Details

The proposed MRWA-FM architecture learns WHAR-tailored Wavelet filter banks with individual filter sizes. The filter size defines the number of Wavelet decomposition levels, i.e., the granularity of the Wavelet analysis, and thus, directly affects the parameter count of MRWA-FM. Furthermore, pre-training MRWA-FM requires a sophisticated pre-training objective that regularizes the Wavelet filter banks to reproduce proper Wavelets. Section 9.4.1 presents possible model configurations. Section 9.4.2 explains the pre-training and fine-tuning objectives.

### 9.4.1 Hyperparameters

MRWA-FM contains several customizable hyperparameters, including the maximum Wavelet decomposition level $J$, the Wavelet filter size $K(J)$, the number of hidden channels $c$, the key dimension $kd$ and the number of attention heads $n_h$ of the MHA layer, and the aggregation function agg of the Enrich layer which

is only relevant during fine-tuning. Table 9.1 lists the values of these hyperparameters. During location-invariant pre-training, MRWA-FM processes signals $(w, f) = (128, 6)$ from single sensor locations (cf. Section 7.1.2). During multi-location fine-tuning, the model uses all $|L|$ locations available in a dataset as inputs $(w, f) = (128, 6\,|L|)$ (cf. Table 7.1). The number of locations, however, has no effect on the model's hyperparameters.

**Table 9.1:** Model configuration.

| Parameter | Value |
|---|---|
| $J$ | $\{3, 4, 5\}$ |
| $K(J)$ | $\lfloor w/2^J \rfloor \in \{16, 8, 4\}$ |
| $c$ | 64 |
| $kd$ | $\{16, 32, 64\}$ |
| $n_h$ | 2 |
| agg | {max, mean, sum} |

Since MRWA-FM precedes a downstream classifier, the combined model must meet the resource constraints of wearable devices. In its largest configuration with $J = 3$, $K(J) = 16$, and $kd = 64$, MRWA-FM learns 35,206 weights during pre-training (cf. Table 9.2a). During multi-location fine-tuning, the number of MRWA-FM weights increases to up to 38.302 for the largest configuration with $|L| = 5$ sensor locations (cf. Table 9.2d). Given that current wearable devices can easily handle models with up to 100k parameters [166], MRWA-FM leaves sufficient capacity for flexible downstream classifiers.

### 9.4.2 Training Setup

The training parameters during pre-training und fine-tuning are almost identical (cf. Table 9.3). Only the batch sizes differ to account for varying train set sizes (cf. Sections 9.2 and 9.3). Both training paths leverage the Adam optimizer [85].

**Table 9.2:** The number of MRWA-FM's weights per hyperparameter configuration and number of sensor locations $|L|$.

**(a)** $|L| = 1$ (required for multi-location fine-tuning of FLAAP and USC-HAD and for location-invariant pre-training; adapted from [4]).

| $K(J)$ | $kd = 16$ | $kd = 32$ | $kd = 64$ |
|---|---|---|---|
| 4(5) | 9.574 | 17.862 | 34.438 |
| 8(4) | 9.830 | 18.118 | 34.694 |
| 16(3) | 10.342 | 18.630 | 35.206 |

**(b)** $|L| = 2$ (required for multi-location fine-tuning of Complex HAD, DLA, and mHealth).

| $K(J)$ | $kd = 16$ | $kd = 32$ | $kd = 64$ |
|---|---|---|---|
| 4(5) | 10.348 | 18.636 | 35.212 |
| 8(4) | 10.604 | 18.892 | 35.468 |
| 16(3) | 11.116 | 19.404 | 35.980 |

**(c)** $|L| = 3$ (required for multi-location fine-tuning of PAMAP2).

| $K(J)$ | $kd = 16$ | $kd = 32$ | $kd = 64$ |
|---|---|---|---|
| 4(5) | 11.122 | 19.410 | 35.986 |
| 8(4) | 11.378 | 19.666 | 36.242 |
| 16(3) | 11.890 | 20.178 | 36.754 |

**(d)** $|L| = 5$ (required for multi-location fine-tuning of DSADS and RealWorld (HAR)).

| $K(J)$ | $kd = 16$ | $kd = 32$ | $kd = 64$ |
|---|---|---|---|
| 4(5) | 12.670 | 20.958 | 37.534 |
| 8(4) | 12.926 | 21.214 | 37.790 |
| 16(3) | 13.438 | 21.726 | 38.302 |

**Table 9.3:** Pre-training and fine-tuning parameters of MRWA-FM.

| Parameter | Pre-Training | Fine-Tuning |
|---|---|---|
| Batch Size | 256 | 64 |
| Patience | 10 | 10 |
| Learning Rate | 1e-3 | 1e-3 |
| Seeds | $\{1, 2, 3\}$ | $\{1, 2, 3\}$ |

During pre-training, the goal of MRWA-FM is to learn Wavelets that perfectly reconstruct input signals $s$. Therefore, $s$ and the reconstructed output signals $s'$ must match and the learned filter banks must form proper Wavelets. The following pre-training loss function ensures both properties:

$$\text{loss} = \text{MSE}(s, s') + \text{rfft}_{\text{reg}} + \text{O}_{\text{reg}} + \text{sum}_{\text{reg}}, \tag{9.9}$$

where the regularization terms

$$\text{rfft}_{\text{reg}} = \sum \text{MAE}(|\text{rfft}(G)|, |\text{rfft}(B)| * \sqrt{2}), \tag{9.10}$$

$$\text{O}_{\text{reg}} = \sum \text{MAE}(G * G, \delta), \tag{9.11}$$

$$\text{sum}_{\text{reg}} = \sum \text{MAE}(\sum G, \sqrt{2}) + \sum \text{MAE}(\sum H, 0) \tag{9.12}$$

force $G$, $H$, $\tilde{G}$, and $\tilde{H}$ to form an orthogonal Wavelet filter bank when $H$, $\tilde{G}$, and $\tilde{H}$ are derived from an orthogonal $G$ according to Equations 2.5, 2.6, and 2.7. Equation 9.10 ensures the low-pass property of the learned decomposition low-pass filter $G$ by comparing its absolute real-valued Fourier coefficients $|\text{rfft}|$ with the Fourier coefficients of a third-order low-pass Butterworth filter $B$ with cutoff frequency at half the sampling rate. Equation 9.11 ensures the orthogonality of $G$, where $G * G$ denotes the convolution of $G$ with itself and $\delta$ is the Kronecker delta function (cf. Equation 2.3). Equation 9.12 ensures that $G$ preserves the signal energy and that $H$ isolates high-frequency details effectively.

Figure 9.6 shows examples of learned Wavelet filter banks with varying filter sizes $K$ and decomposition levels $J$, along with their corresponding frequency responses. The plots demonstrate that MRWA-FM learns valid filter banks that retain key properties of orthogonal Wavelets, including the expected relationships between decomposition and reconstruction filters as well as their high- and low-pass characteristics (cf. Equations 2.3 to 2.7). Although the learned Wavelets are more irregular and asymmetric than literature-defined Wavelets such as Haar [57] or Daubechies [34], they are particularly well-suited for WHAR tasks due to their data-driven optimization.

During fine-tuning, this work optimizes MRWA-FM jointly with the downstream classifiers TinyHAR [177], DeepConvLSTM [17, 122], and GRU [165] using the categorical cross-entropy loss [29, 53].

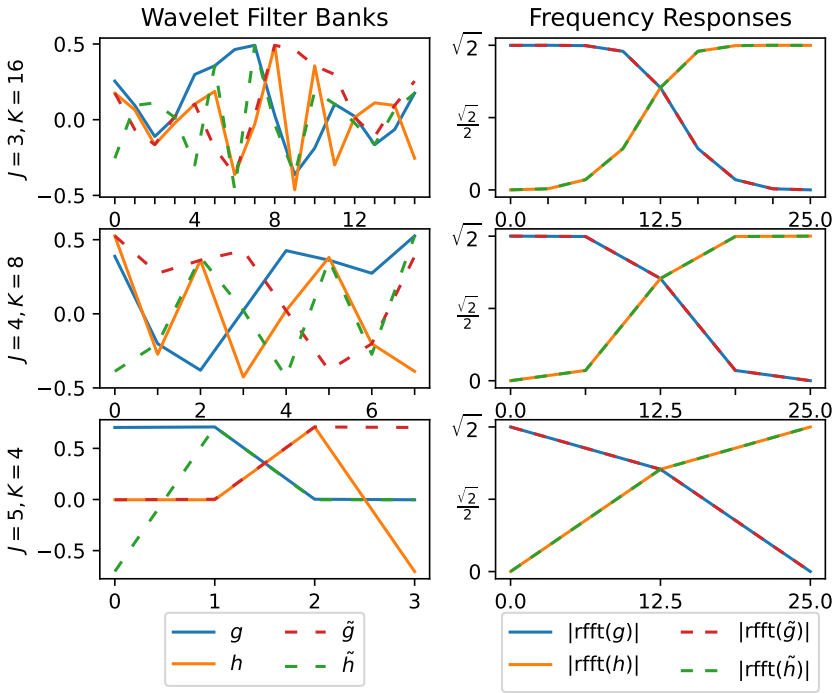



**Figure 9.6:** Learned Wavelet filter banks (left) and their frequency responses (right) for different Wavelet levels $J$ and Wavelet filter sizes $K$ (adapted from [4]).

## 9.5 Performance Analysis

To evaluate the impact of MRWA-FM on downstream classification, Section 9.5.1 compares the macro F1-scores and accuracies (cf. Equations 2.10 and 2.11) of three state-of-the-art classifiers (TinyHAR, DeepConvLSTM, GRU) with and without prepended MRWA-FM. Additionally, Section 9.5.2 analyzes the number of true classifications, Section 9.5.3 evaluates inference times and the number of multiply-accumulate operations, and Section 9.5.4 compares MRWA-FM to other Wavelet-integrated classifiers.

To this end, this work pre-trains MRWA-FM using LODO cross-validation with seven pre-training datasets and multiple random seeds (cf. Table 9.3). Fine-tuning and the training of the baseline classifiers is performed on an eighth holdout dataset using varying random seeds.

Given the configurability of MRWA-FM (cf. Table 9.1), this section also explores variations in the key dimension $kd$ of the MHA layer, the number of Wavelet decomposition levels $J$, and the aggregation function agg. Additionally, this work investigates the effect of adapting the weights of pre-trained layers during fine-tuning. In the None configuration, all MRWA-FM layers are frozen and only the downstream classifiers are trained. In the InvProj configuration, the InvProj layer is trainable, while all other pre-trained layers remain frozen. In the MHA configuration, both the InvProj layer and the MHA layer are fine-tuned.

### 9.5.1 Comparison with State-of-the-Art Classifiers

Figures 9.7 and 9.8 present the average macro F1-scores and accuracies of the plain classifiers (grey bars) and downstream classifiers with prepended MRWA-FM (orange and green bars). The orange bars represent the best MRWA-FM configurations per dataset and classifier (bar labels: $kd$, $J$, agg, trainable MRWA-FM layers), while the green bars present the scores of the best configurations per dataset. Thus, the orange bars are always higher than or as high as the green bars. To keep the configurations consistent between Figures 9.7 and 9.8, this work selects the best performing configurations according to their average macro F1-scores.

In general, the macro F1-scores of all classifiers closely align with their accuracies, indicating consistent predictions across all activities. Only for USC-HAD, the accuracies are up to 0.04 points higher than the macro F1-scores. This discrepancy suggests a slight bias of the models toward the majority class in this dataset. Although the following paragraphs primarily focus on macro F1-scores, their general findings are also applicable to the accuracy metrics due to their strong similarity.

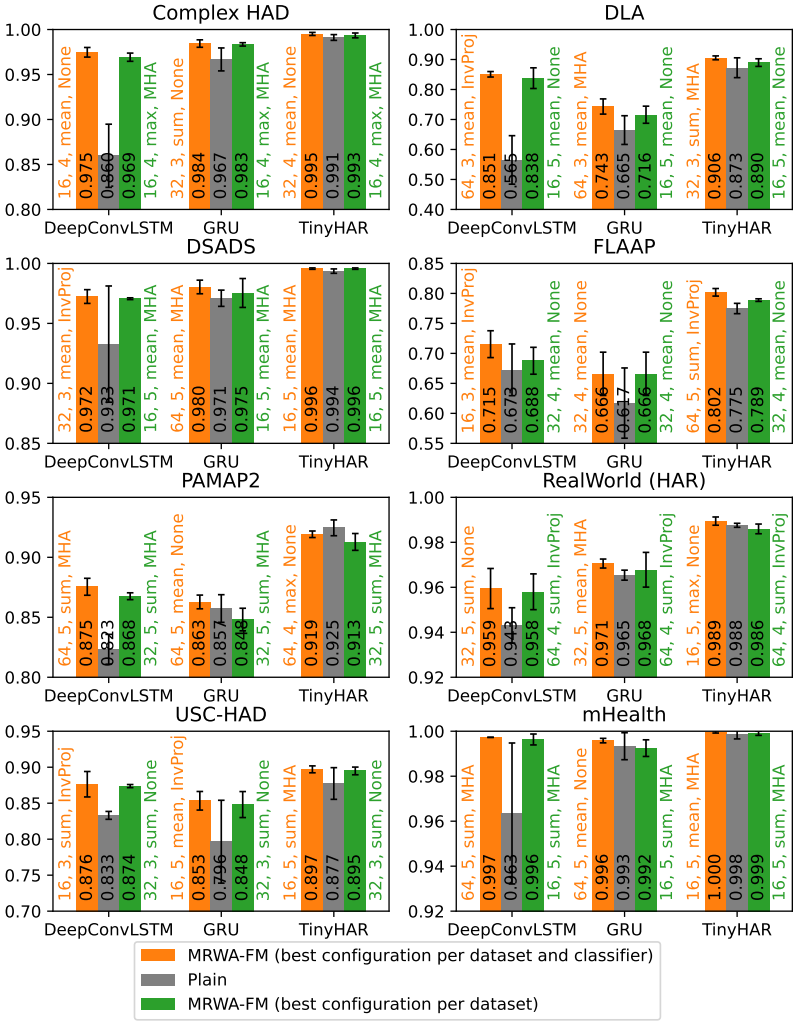



**Figure 9.7:** Average macro F1-scores of state-of-the-art classifiers with and without prepended MRWA-FM over three runs, including standard deviations. Grey bars represent plain classifiers without MRWA-FM. Green bars represent the best MRWA-FM configurations per dataset, while orange bars represent the best configurations per dataset and classifier, selected based on macro F1-scores. Bar labels specify the corresponding MRWA-FM configurations ($kd$, $J$, agg, trainable layers).

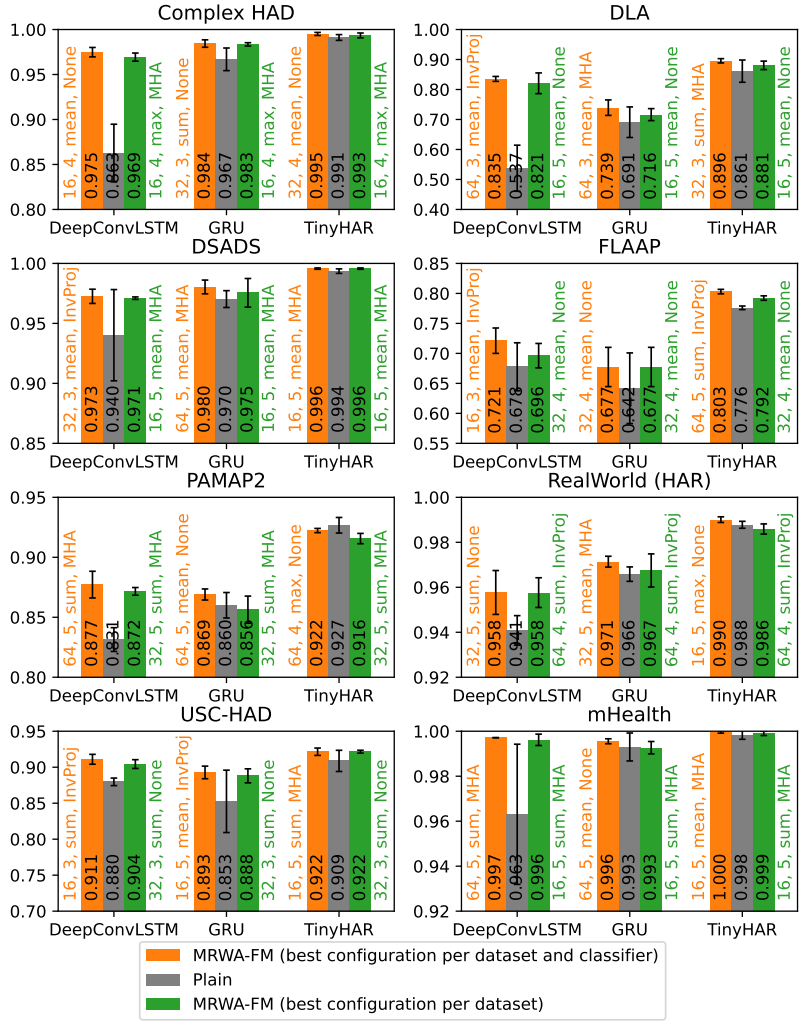



**Figure 9.8:** Average accuracies of state-of-the-art classifiers with and without prepended MRWA-FM over three runs, including standard deviations. Grey bars represent plain classifiers without MRWA-FM. Green bars represent the best MRWA-FM configurations per dataset, while orange bars represent the best configurations per dataset and classifier, selected based on macro F1-scores. Bar labels specify the corresponding MRWA-FM configurations ($kd$, $J$, agg, trainable layers).

MRWA-FM significantly enhances the classification performance of the downstream classifiers over their plain counterparts across all datasets. With an improvement of 0.286 points in the macro F1-score, the performance gains of the orange configurations are most pronounced for DeepConvLSTM on DLA. GRU and TinyHAR also achieve the largest gains on DLA with F1 improvements of 0.078 and 0.033 points, respectively. Remarkably, the orange configurations improve DeepConvLSTM and GRU on Complex HAD, DSADS, USC-HAD, and mHealth close to the performance of plain TinyHAR, which is the most advanced of the three classifiers. The only dataset where a plain classifier outperforms MRWA-FM is PAMAP2. Here, TinyHAR slightly surpasses MRWA-FM's F1-score by a mere 0.006 points. On average, prepending MRWA-FM with orange configurations to the classifiers improves their F1-scores by 0.039 points. These results demonstrate the expressiveness of the learned multi-resolution features.

The smallest performance gains are noticeable on datasets where the plain classifiers already achieve high scores. For instance, MRWA-FM improves the macro F1-score of DeepConvLSTM on RealWorld (HAR) by 0.016, GRU achieves an improvement of 0.003 points on mHealth, and TinyHAR gains 0.002 points on DSADS. While these numbers seem small, the performance gains are still significant as improving classifiers that are close to the optimal performance of 1.0 is extremely challenging.

In 21 of 24 cases, MRWA-FM with orange configurations also reduces the standard deviations of F1-scores, with an average decrease of 0.015. On Complex HAD, for example, the standard deviations drop by 0.030, 0.009, and 0.002 points for DeepConvLSTM, GRU, and TinyHAR, respectively. These reductions demonstrate MRWA-FM's robustness and stability, as the model exhibits lower sensitivity to variations in weight initialization and data splits compared to the plain classifiers. Consequently, MRWA-FM not only achieves higher classification rates but also delivers more consistent results, making it well-suited for real-world applications.

The general findings observed for the orange configurations also apply to the green configurations. However, the average performance gains of the green configurations over the plain classifiers are slightly smaller (0.032 points). These results are expected as, unlike the orange configurations, the green configurations are not explicitly optimized for a specific classifier. In 5 of 24 cases, the plain classifiers even surpass the green configurations in macro F1-scores (GRU on PAMAP2, RealWorld (HAR), and mHealth; TinyHAR on PAMAP2 and RealWorld (HAR)), though these degradations are typically minor (less than 0.01 points).

Overall, MRWA-FM enhances state-of-the-art classifiers in almost all dataset-classifier combinations. The consistently high classification rates highlight the advantage of incorporating advanced feature extraction mechanisms. While the optimal MRWA-FM configuration depends on the specific dataset and classifier, the trends observed in Figures 9.7 and 9.8 provide practical guidance for future developments.

In terms of hyperparameter choice, the orange and green configurations perform best with $J = 5$ (13 of 24 orange configurations, 4 of 8 green configurations), suggesting that deeper Wavelet decompositions and finer frequency analyses contribute to improved classification performance. During the Wavelet enrichment step (E3), mean aggregation is preferred (14 orange configurations, 3 green configurations), followed by sum aggregation (8 orange configurations, 4 green configurations). The dominance of mean and sum over max aggregation is expected, as those two functions consider the full set of attention coefficients rather than relying on a single maximum value.

Fine-tuning only the InvProj layer yields limited improvements (6 of 24 orange cases, 1 of 8 green cases). Adjusting both MHA and InvProj or freezing all layers is generally more effective. Regarding the key dimension $kd$ of the MHA layer, no clear trend is evident; therefore, its selection should be primarily based on the resource constraints of the target platform.

### 9.5.2 Activity-Level Performance

Figure 9.9 compares the total number of true classifications of the orange and green MRWA-FM configurations per classifier and general activity (cf. Section 7.1.1) relative to the plain classifiers (baseline) using divergent bar charts. In these charts, bars above the baseline highlight activities where MRWA-FM outperforms the plain classifiers, while bars below the baseline indicate where MRWA-FM underperforms.

Both configurations improve DeepConvLSTM in the detection of 35 of 40 general activities. The most significant improvements are visible for more static activities such as handwriting, sit, laptop, and lay. The orange configuration matches the plain classifier in two general activities (crosstrainer, sleep), while the green configuration matches the plain classifier in a single general activity.

MRWA-FM improves GRU in the detection of 31 (orange) or 22 (green) general activities. Activities with the largest improvements include walk, sit, and downstairs. The orange configuration matches the plain classifier in four general activities, while the green configuration matches the plain classifier in five general activities.

MRWA-FM improves TinyHAR in the detection of 23 (orange) or 19 (green) general activities. Sit, elevator, and teeth brush show the largest improvements. Both configurations match their plain counterparts in four general activities.

Overall, the charts confirm the general findings from Figures 9.7 and 9.8: DeepConvLSTM benefits the most from MRWA-FM, while TinyHAR benefits the least but the improvements are still significant. Additionally, the charts suggest a trend where MRWA-FM specifically enhances the detection of more static activities that involve minimal body movement.

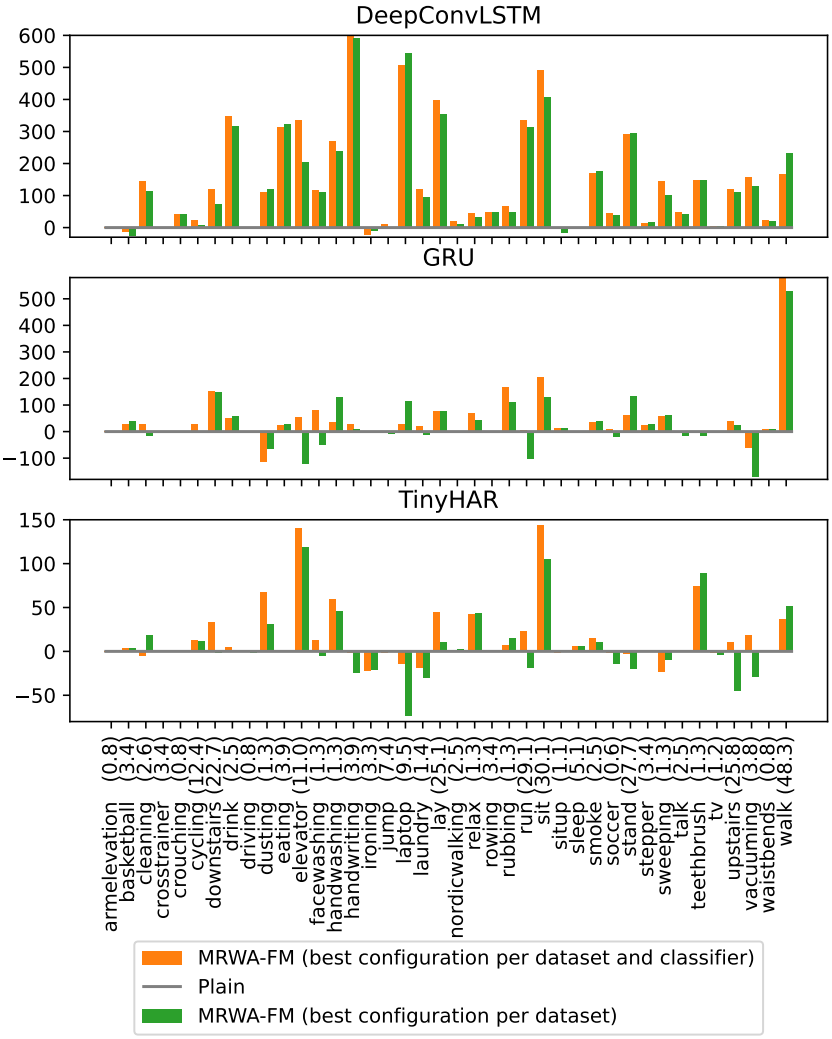



**Figure 9.9:** Comparison of the total number of true classifications for the orange and green MRWA-FM configurations relative to the plain classifiers per general activity $g$. The labels on the horizontal axis are $g$ (number of test samples per $g$ in 1000).

### 9.5.3 Efficiency Analysis

To evaluate the lightness and real-world applicability of MRWA-FM, this work deploys the models and the plain classifiers as TensorFlow Lite models with $32\,\mathrm{bit}$ weights on a Fossil Gen 5 Carlyle HR smartwatch and measures their inference times. This wearable features a four core ARM Cortex-A7 at $1.2\,\mathrm{GHz}$ and $1\,\mathrm{GB}$ of LPDDR3 RAM at $400\,\mathrm{MHz}$. Figure 9.10 compares the average inference times (in milliseconds) of the plain classifiers and the best MRWA-FM configurations (orange configurations in Figure 9.7) over 100 runs per dataset. To further analyze the computational complexity of the models, Figure 9.10 also presents the estimated MMACs (Millions of Multiply-Accumulate Operations) of the models after TensorFlow Lite conversion.

As expected, MRWA-FM increases the inference times and MMACs over the plain classifiers for all dataset-classifier combinations due to the additional MRWA operations. The inference times increase by up to $25.630\,\mathrm{ms}$ for DeepConvLSTM on DSADS, up to $44.396\,\mathrm{ms}$ for GRU on Complex HAD, and up to $68.396\,\mathrm{ms}$ for TinyHAR on RealWorld (HAR). However, the inference times never rise above $430\,\mathrm{ms}$ which is still well below the duration of the $2.56\,\mathrm{s}$-long input signal windows (cf. Section 7.1.2).

The MMACs increase by up to 7.350 for DeepConvLSTM on PAMAP2, up to 6.248 for GRU on DLA, and up to 18.655 for TinyHAR on mHealth and DSADS. These increases arise from the additional MRWA operations and the doubling of feature dimensions in the inputs to the downstream classifiers. The growth in input dimensions results from concatenating the MRWA outputs with the original inputs along the feature axis prior to classification (cf. Figure 9.1b). Consequently, the MMACs of MRWA-FM vary across downstream classifiers, even under identical configurations.

Despite the GRU models consistently having the lowest MMACs, their inference times are always the largest. The reason for that is the repeated application of recurrent layers on full-length inputs in the early stages of the GRU classifier.

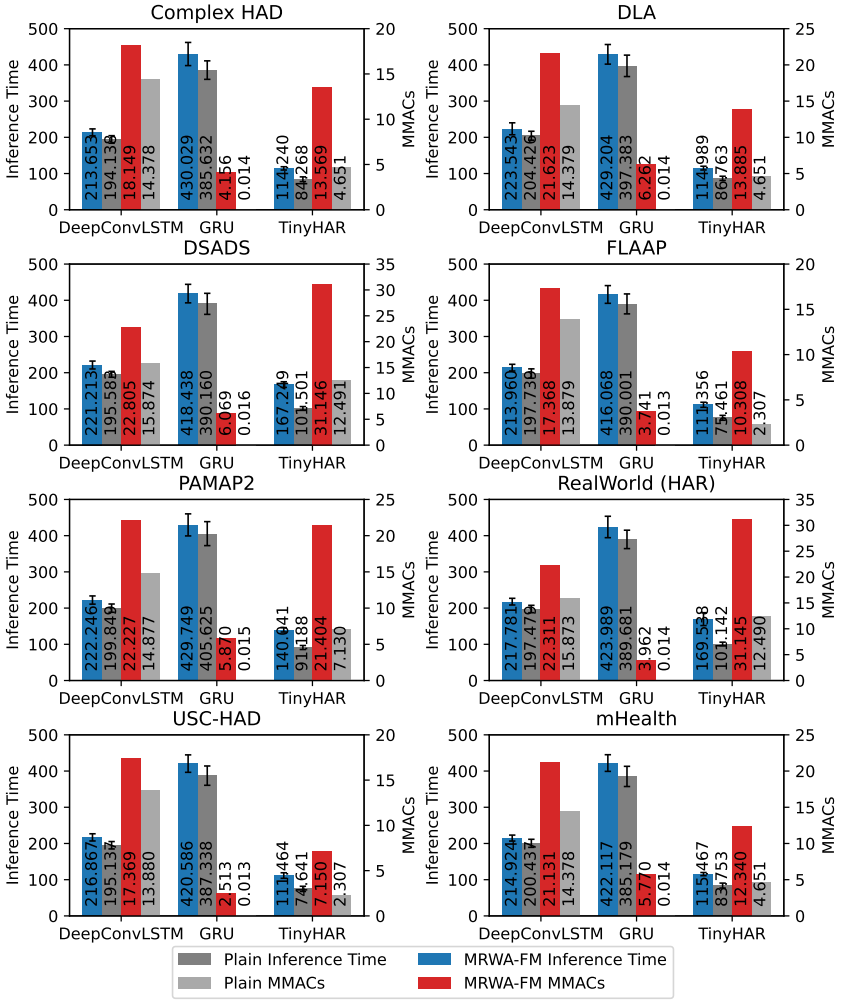



**Figure 9.10:** Comparison of the average inference times in ms over 100 runs and the MMACs of the plain classifiers with the best MRWA-FM's per dataset and classifier after TensorFlow Lite conversion, including standard deviations for inference times; the MMACs remain constant across runs (adapted from [4]).

Appendix A.2 contains further comparisons of inference times (Figure A.3) and MMACs (Figure A.4) for different MRWA-FM configurations. The Wavelet decomposition level $J$, the key dimension $kd$ of the MHA layer, and the aggregation function agg have only a minor impact on the inference times with no particular trends being visible. The MMACs consistently increase with $kd$ but reduce with increasing $J$ since each added Wavelet level shrinks the Wavelet kernel size $K(J)$ by a factor of 2 (cf. Table 9.1).

Overall, the efficiency analysis, the impressive classification performance, and the small parameter counts (cf. Table 9.2) confirm the practical applicability of MRWA-FM to real-world scenarios where processing resources are limited.

### 9.5.4 Comparison with other Wavelet-Integrated Models

Table 9.4 compares MRWA-FM with two WHAR classifiers that incorporate Wavelets directly into their architectures: MLCNNwav [32] and Wavelet-based Learnable Filters (WLF) [174]. MLCNNwav is an end-to-end classifier that utilizes Daubechies Wavelets of various orders (db1, db2, db3, db4, db5, or db6) to decompose the hidden representations of a CNN. In contrast, WLF convolves input signals with a selection of discrete Wavelets, applies a learnable weight vector to the resulting coefficients, and then forwards the learned representations to a downstream classifier. To ensure a fair comparison between MRWA-FM and WLF, this study appends the same three downstream classifiers to both models. Furthermore, it trains and evaluates MLCNNwav and WLF using the same data splits and training parameters established during the fine-tuning of MRWA-FM (cf. Sections 9.3 and 9.4.2).

Even with the green configurations, MRWA-FM consistently outperforms MLCNNwav across all datasets. Overall, MRWA-FM's best configurations exceed the macro F1-scores of the best MLCNNwav (db4) by 0.268 points ($40.0\,\%$) and outperform the weakest MLCNNwav (db5) by 0.338 points ($56.3\,\%$).

**Table 9.4:** Average macro F1-scores of MRWA, MLCNNwav [32], and WLF [174] per dataset, along with the overall averages across all datasets (adapted from [4]).

| | Model | Complex HAD | DLA | DSADS | FLAAP | PAMAP2 | RealWorld (HAR) | USC-HAD | mHealth | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| MLCNNwav | db1 | 0.745 | 0.343 | 0.630 | 0.439 | 0.493 | 0.820 | 0.691 | 0.846 | 0.626 |
| | db2 | 0.696 | 0.429 | 0.810 | 0.396 | 0.383 | 0.885 | 0.609 | 0.789 | 0.625 |
| | db3 | 0.802 | 0.297 | 0.884 | 0.377 | 0.315 | 0.739 | 0.592 | 0.948 | 0.619 |
| | db4 | 0.850 | 0.351 | 0.829 | 0.382 | 0.513 | 0.819 | 0.742 | 0.874 | 0.670 |
| | db5 | 0.788 | 0.306 | 0.808 | 0.403 | 0.200 | 0.741 | 0.665 | 0.885 | 0.600 |
| | db6 | 0.577 | 0.374 | 0.912 | 0.386 | 0.324 | 0.866 | 0.622 | 0.945 | 0.626 |
| WLF | DeepConvLSTM | 0.785 | 0.578 | 0.957 | 0.672 | 0.800 | 0.969 | 0.856 | 0.991 | 0.826 |
| | GRU | 0.984 | 0.654 | 0.979 | 0.653 | 0.857 | 0.962 | 0.797 | 0.996 | 0.860 |
| | TinyHAR | 0.992 | 0.874 | 0.992 | 0.751 | 0.911 | 0.984 | 0.895 | 0.994 | 0.924 |
| MRWA-FM | DeepConvLSTM | 0.969 | 0.838 | 0.971 | 0.688 | 0.868 | 0.958 | 0.874 | 0.996 | 0.895 |
| | DeepConvLSTM | 0.975 | 0.851 | 0.972 | 0.715 | 0.875 | 0.959 | 0.876 | 0.997 | 0.903 |
| | GRU | 0.983 | 0.716 | 0.975 | 0.666 | 0.848 | 0.968 | 0.848 | 0.992 | 0.875 |
| | GRU | 0.984 | 0.743 | 0.980 | 0.666 | 0.863 | 0.971 | 0.853 | 0.996 | 0.882 |
| | TinyHAR | 0.993 | 0.890 | 0.996 | 0.789 | 0.913 | 0.986 | 0.895 | 0.999 | 0.933 |
| | TinyHAR | 0.995 | 0.906 | 0.996 | 0.802 | 0.919 | 0.989 | 0.897 | 1.000 | 0.938 |

While WLF demonstrates significantly better performance than MLCNNwav, MRWA-FM still surpasses WLF in nearly all scenarios. Only in a few cases, such as DeepConvLSTM on RealWorld (HAR) and GRU on ComplexHAD and mHealth, WLF either matches or slightly surpasses the performance of MRWA-FM with orange configurations. For TinyHAR on USC-HAD and GRU on DSADS and PAMAP2, WLF marginally outperforms the green configurations. On average, MRWA-FM with its optimal configuration (orange) improves over WLF by 0.077 points (9.3 %), 0.022 points (2.5 %), and 0.014 points (1.5 %) with DeepConvLSTM, GRU, and TinyHAR, respectively.

These observations highlight the superior generalizability and consistency of MRWA-FM over other state-of-the-art Wavelet-based approaches across a variety of sensing setups and activity contexts.

## 9.6 Summary

This chapter presented MRWA-FM, a lightweight FM architecture that analyzes sensor signals at multiple frequency resolutions to extract detailed and discriminative signal features. The model uses location-invariant pre-training on seven consolidated datasets to learn Wavelet filter banks that emphasize relevant frequency bands across diverse sensor signals. It then performs multi-location fine-tuning on an independent eighth dataset to adapt the pre-trained weights for sensing setups with multiple sensor locations. This adaptation enables downstream classifiers to exploit inter-location correlations to improve their accuracy. Prepending MRWA-FM to state-of-the-art classifiers consistently boosts their performance while only marginally increasing model sizes and inference times on a smartwatch with limited processing resources. Thus, combining modern machine learning with advanced signal processing supports the development of efficient WHAR-FMs that are suitable for wearable deployment.

# Part IV

# Conclusion and Future Directions

# 10 Conclusion

The primary objective of this work was to advance the field of wearable human activity recognition through advanced signal representations, the development of universal machine learning models, and their rigorous evaluation in wearable scenarios. This chapter concludes the dissertation by summarizing its core contributions C1 to C4 (cf. Section 1.4), reflecting on their implications, and outlining promising directions for future research.

### C1 – Graph Transformations.

This work introduced two graph transformations tailored for WHAR. One transformation is designed for basic activity classification, while the other addresses sequential activity recognition. By design, both transformations are lightweight and construct sparse graph structures suitable for deployment on resource-constrained wearable devices. Given their versatility, graph representations present a promising path for advancing WHAR applications, especially in scenarios involving compound activities. Future research should therefore focus on developing graph transformations that capture dynamic interactions between activities, users, and even objects over extended signal periods, further enhancing the recognition of complex human behavior. Such improvements could significantly benefit applications in virtual and extended reality, where real-time recognition of activities and interactions is crucial for delivering immersive user experiences.

### C2 – Graph Neural Networks.

Building on C1, this work developed two lightweight GNN architectures for diverse activity classification tasks. The first architecture achieved performance

on par with more complex, state-of-the-art models in classifying basic activities. The second demonstrated the potential of GNNs in predicting relationships between interrelated activities. However, the scarcity of large-scale, real-world datasets presents a significant barrier to training models capable of recognizing complex, compound activities. To support emerging applications in domains where real-time understanding of user intent and context is essential (e.g., assistive technologies, smart environments, and extended reality), future research must prioritize the collection of comprehensive datasets. These datasets should capture both basic and compound activities, as well as interactions between users and their environment. Such detailed information will be key to enabling more advanced GNN architectures that can learn from rich, structured behavioral data and power the next generation of intelligent activity recognition systems.

### C3 – Harmonizing Heterogeneous Signals.

A major obstacle in WHAR is the diversity of publicly available training datasets in terms of user-specific behavior, sensor placements, signal properties, and activity labels. To address this challenge, this work proposed a harmonization framework that standardizes heterogeneous data sources by aligning sensor signals, metadata, and activity annotations. This framework enables more robust, large-scale cross-dataset training of machine learning models and lays the foundation for developing advanced applications that operate reliably across domains, devices, and user populations, while supporting novel use cases that require adaptability to diverse sensing contexts. To realize its full potential, future datasets should include raw sensor signals, detailed metadata, such as sensor specifications and environmental conditions, as well as standardized annotation schemas for activities and sensor placements.

### C4 – Foundation Models.

Finally, this work investigated the applicability of foundation model concepts to the WHAR domain and introduced the term WHAR-FM, referring to foundation models specifically tailored to wearable sensor data, human activity recognition tasks, and deployment on resource-constrained devices. This work introduced

two distinct architectures that analyze wearable signals from multiple perspectives and demonstrate superior capabilities in both signal forecasting and activity classification compared to state-of-the-art methods. These results highlight the strong potential of WHAR-FMs for improving generalization across datasets, sensor configurations, and application domains. To fully leverage this potential, future research could focus on exploring innovative pre-training strategies that emphasize the unique properties of wearable sensor signals. Additionally, developing scalable architectures could facilitate the automatic detection of nuanced human behavior and interactions, while integrating alternative data modalities could broaden the spectrum of applications beyond WHAR. Such advancements pave the way for more adaptive, personalized, and intelligent applications that enhance user experience in real-world environments.

### Closing Remarks.

In conclusion, this dissertation marks a significant step toward advanced wearable human activity recognition. By combining structured signal representations and sophisticated signal processing with scalable machine learning models, the presented method portfolio equips researchers and engineers with practical tools for advancing both WHAR research and real-world application of future WHAR use cases. Beyond its technical innovations, this work contributes to the growing role of wearable activity recognition in society as it lays the groundwork for developing reliable and versatile applications across various domains, including daily life, sports, health monitoring, and interactive systems.

# A Appendix

## A.1 Signal Forecasting with Foundation Models – Comparison Models

TSMixer and the Transformer model implement the same high-level architecture as Dual-View FM (cf. Figure 8.3) but replace the Dual-View Mixing blocks with Mixer Layers (cf. Figure 8.1) or Transformer Encoders [149], respectively. For consistency, both models leverage similar hyperparameters as Dual-View FM (cf. Section 8.4.2).

**TSMixer Architecture (Figure A.1).**

TSMixer reuses the hyperparameters $N_B$, $N_A$, $d$, and $n_{\text{mask}}$. In line with its publication [26], TSMixer leverages hidden size $h = 64$ and omits the location input, making it location-unaware. The Temporal Projection layer [26, 171] transforms the $w$-dimension of an input signal $s \in \mathbb{R}^{w \times f}$ to $w'$.

**Transformer Architecture (Figure A.2).**

The location-aware Transformer reuses the hyperparameters $N_B$, $N_A$, $N_H$, $d$, and $n_{\text{mask}}$. Its base and head consist of convolutional layers (CNN), positional encoding (PE, [149]), Transformer Encoders [149] with $n_h = 4$ attention heads, and fully-connected layers (FC). This configuration has a similar number of parameters as Dual-View FM.

The CNN layers in the base increase the feature dimension $f$ of an intermediate representation $(w, f)$ to $w$; the FC layers reduce the dimensions back to $(w, f)$. The encoders in the base use hidden size $h = w = 128$.

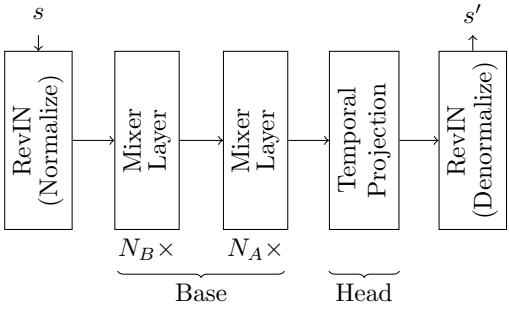



**Figure A.1:** High-level architecture of the location-unaware TSMixer [26].

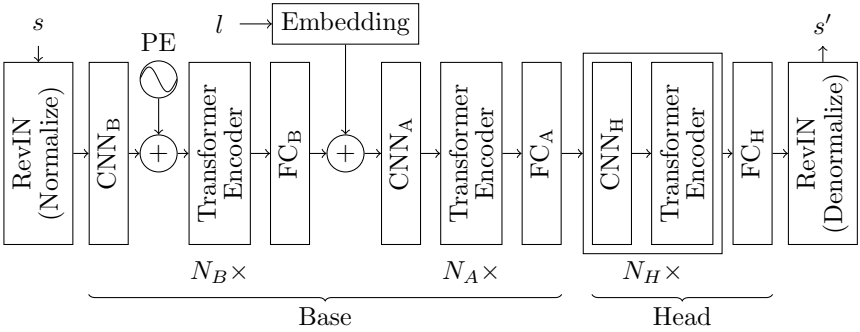
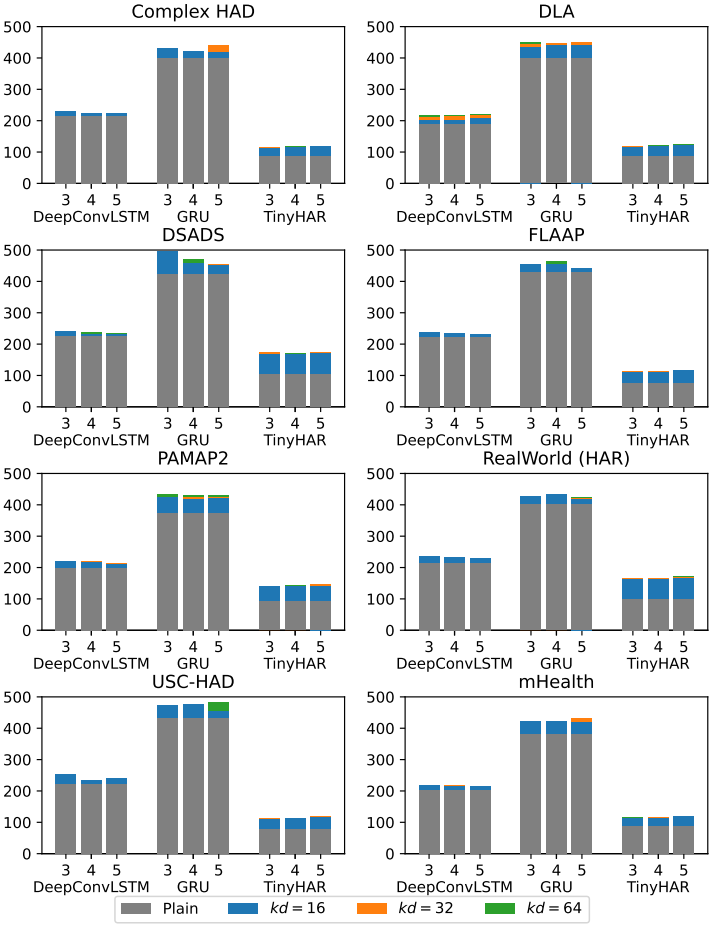



**Figure A.2:** High-level architecture of the location-aware Transformer model.

With the MSM objective, the $N_H = 1$ CNN layer and the Transformer encoder in the head use the same configuration as the CNN layers and encoders in the base.

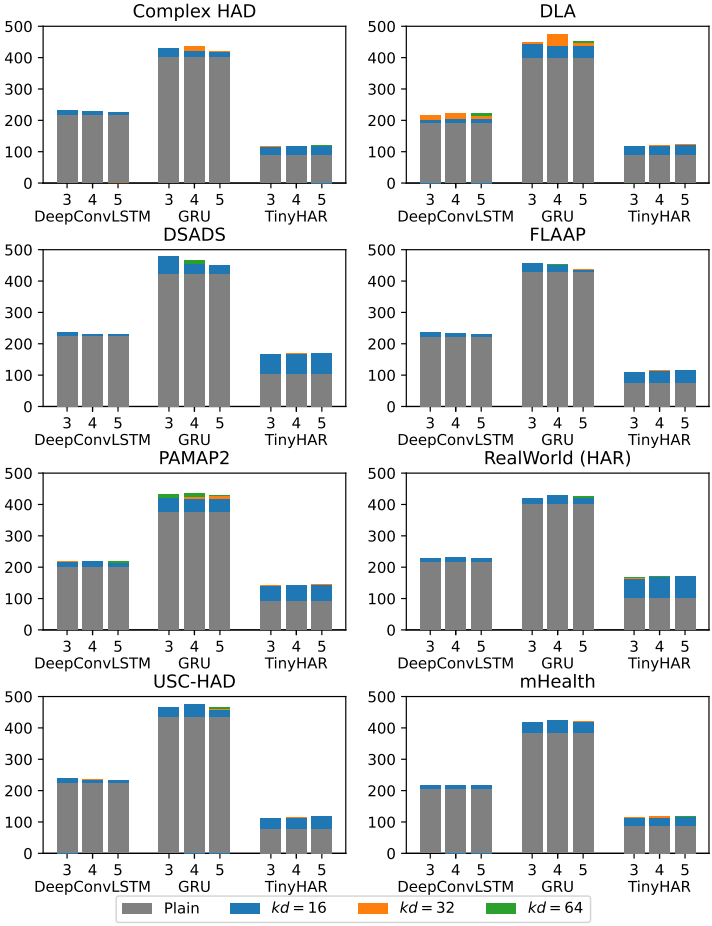
With the CSM objective, the $N_H = 3$ CNN layers and the Transformer encoders in the head reduce the $w$-dimension to the target length $w' = n_{\text{mask}}$. Each CNN layer uses a stride of 2, halving the width at each step. Simultaneously, the output width at each stage is used as both the number of CNN output channels and the hidden size of the corresponding Transformer encoder. Thus, starting from an input of shape $(w, f)$, the first CNN-Encoder pair produces an output of shape $(w/2, w/2)$, the second pair reduces it to $(w/4, w/4)$, and the third to $(w/8, w/8)$. For $w = 128$, this yields a final width of $w/8 = 16 = n_{\text{mask}}$. Finally, $\text{FC}_\text{H}$ reduces the feature dimension to the desired output shape $(w', f)$.

## A.2 Activity Classification with Foundation Models – Model Efficiency



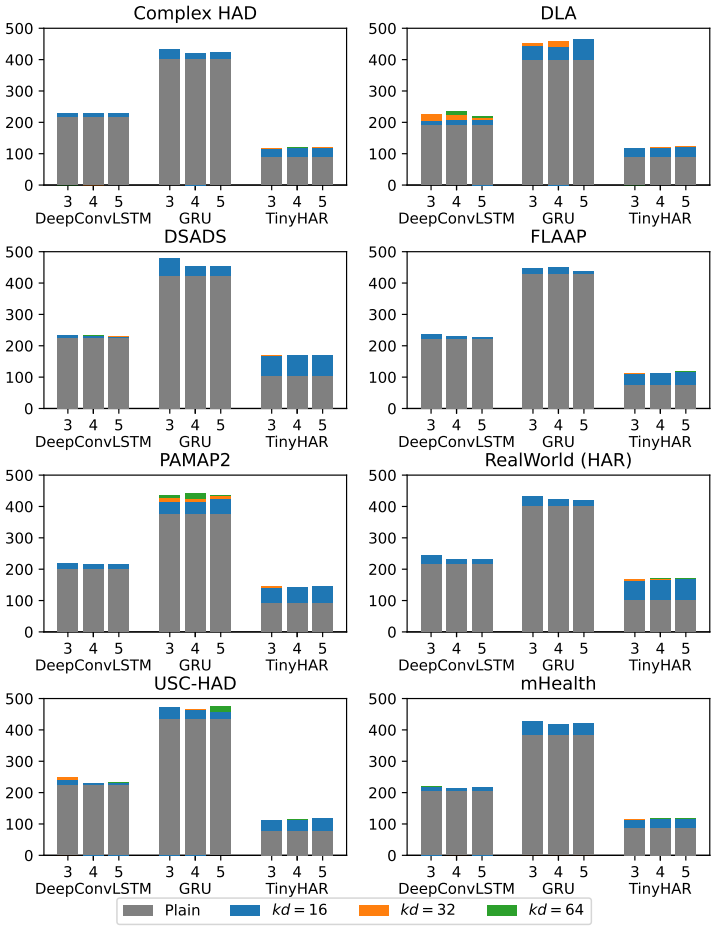

**(a)** Max Aggregation.

**Figure A.3:** Inference times of MRWA-FM in ms averaged over 100 runs on a Fossil Gen 5 Carlyle HR for different Wavelet levels $J \in \{3, 4, 5\}$ and key dimensions $kd$.




**(b)** Mean Aggregation.

**Figure A.3:** Continued.




**(c)** Sum Aggregation.

**Figure A.3:** Continued.

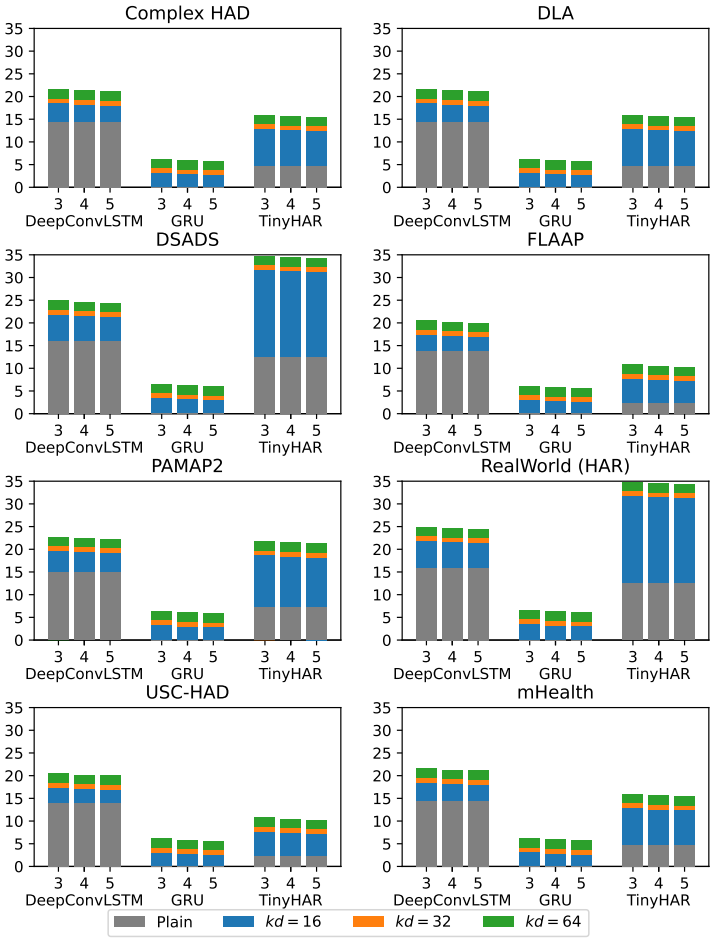



**Figure A.4:** MMACs of MRWA-FM after TensorFlow Lite conversion for different Wavelet levels $J \in \{3, 4, 5\}$ and key dimensions $kd$. Aggregation functions are ignored as they do not affect MMACs at this resolution.

# List of Figures


1.1 Tradeoff triangle of key challenges in WHAR. . . . . 4
1.2 Tradeoff triangle of key challenges in WHAR labeled with this work's objectives. . . . . 6
1.3 Tradeoff triangle of key challenges in WHAR labeled with this work's research questions. . . . . 7
1.4 Tradeoff triangle of key challenges in WHAR labeled with this work's contributions. . . . . 9
1.5 Thesis outline and interplay between chapters. . . . . 10
2.1 Sliding window that is shifted through a three-axis accelerometer signal. 14
2.2 Table tennis exercise "Wide Falkenberg" with three subsequent strokes. 15
2.3 Illustration of a single-level DWT. . . . . 17
2.4 Classification of a three-axis accelerometer signal recorded at the ankle during walking with a learned classification function. . . . 20
2.5 Forecast of a three-axis accelerometer signal recorded at the ankle during walking with a learned forecasting function. . . . . 23
2.6 Three-axis accelerations recorded at the ankle during running. The right plot contains a typical running pattern with distinct peaks corresponding to steps, while the left plot shows low-variance signals that are nearly constant. . . . . 28
3.1 Example graphs with three vertices. . . . . 32
3.2 Graph-level, vertex-level, and edge-level prediction tasks. . . . . 33
3.3 Receptive fields after applying one, two, and three GNN layers to a graph. . . . . 34
3.4 Example window, time step, and activity graphs. . . . . 39
3.5 Illustrations of graph structures. . . . . 40
4.1 Workflow of WHAR-FMs. . . . . 49
4.2 Overview of different masking approaches. . . . . 51

4.3 Task-specific fine-tuning by appending new layers to a pre-trained model. . . . . 54
5.1 Signal-graph transformation for lightweight WHAR with GNNs. . . 59
5.2 The lightweight GNN architecture for signal graph classification. . . 61
5.3 The average confusion matrix of the best performing GNN configuration on KU-HAR. . . . . 68
5.4 The average confusion matrix of the best performing GNN configuration on PAMAP2. . . . . 70
5.5 The average confusion matrix of the best performing GNN configuration on UCI-HAR. . . . . 71
5.6 The average confusion matrix of the best performing GNN configuration on Daphnet. . . . . 72
5.7 Comparison of network parameters, F1-scores, and accuracies between the best graph and GNN configurations and state-of-the-art classifiers on KU-HAR. . . . . 73
5.8 Comparison of network parameters, F1-scores, and accuracies between the best graph and GNN configurations and state-of-the-art classifiers on PAMAP2. . . . . 74
5.9 Comparison of network parameters, F1-scores, and accuracies between the best graph and GNN configurations and state-of-the-art classifiers on UCI-HAR. . . . . 75
5.10 Comparison of network parameters, F1-scores, and accuracies between the best graph and GNN configurations and state-of-the-art classifiers on Daphnet. . . . . 76
5.11 The average inference times and MMACs of the best GNN configurations on UCI-HAR, PAMAP2, KU-HAR, and Daphnet. . . . . 77
6.1 Activity-graph transformation for the prediction of sequential activities with GNNs. . . . . 80
6.2 Appending a newly observed micro-activity to the graph. . . . . 81
6.3 The GNN architecture for sequential activity prediction. . . . . 83
6.4 PCAs of the mean UMAP embeddings for two Cooking datasets. . . 88
6.5 Graph validation and link prediction results for different standard deviations and varying UMAP embedding sizes on the Cooking dataset. . . . . 92

6.6 Graph validation and link prediction results for different standard deviations and varying UMAP embedding sizes on the TT-Strokes dataset. . . . . 94
7.1 Three-axis accelerometer signals recorded at the ankle and wrist during ironing. . . . . 101
7.2 The standardized sensor placements on the human body and the number of six-axis sensors at these locations. . . . . 102
7.3 Leave-One-Dataset-Out (LODO) pre-training split for eight available datasets. . . . . 106
8.1 Simplified view of the Mixer Layer. . . . . 110
8.2 Dual-View FM's Local and Global Time and Feature Mixing. . . . . 112
8.3 High-level architecture of Dual-View FM. . . . . 113
8.4 Pre-training and fine-tuning data splits. . . . . 115
8.5 MAEs of Dual-View FMs on unseen holdout datasets plotted against the number of pre-training datasets sorted by size. . . . . 125
8.6 Comparison of the fine-tuned CSM models per dataset and metric. . 128
8.7 Average MAEs and RMSEs of the fine-tuned, location-unaware Dual-View FMs with CSM objective per general activity. . . . . 129
8.8 Example forecasts of unseen walking and running signals from the fine-tuned, location-unaware Dual-View FMs with CSM objective. 130
9.1 Architecture of MRWA-FM. . . . . 133
9.2 Decomposition of a normalized signal into Wavelet coefficients using a learned, four-level Wavelet transform, followed by concatenation of the coefficients. . . . . 134
9.3 Internal processing of the attention matrix within the Enrich layer, resulting in the weighted Wavelet coefficients. . . . . 137
9.4 Illustration of the interactions between MRWA's decomposition, attention, enrichment, and reconstruction layers. . . . . 138
9.5 The effects of different aggregation functions in the Enrich layer on MRWA-FM's outputs and the aggregated and normalized attention coefficients for different Wavelet levels and filter sizes. . . . 140
9.6 Learned Wavelet filter banks (left) and their frequency responses (right) for different Wavelet levels and Wavelet filter sizes. . . . . 145
9.7 Average macro F1-scores of state-of-the-art classifiers with and without prepended MRWA-FM. . . . . 147

9.8 Average accuracies of state-of-the-art classifiers with and without prepended MRWA-FM. . . . . 148
9.9 Comparison of the total number of true classifications for the orange and green MRWA-FM configurations relative to the plain classifiers per general activity. . . . . 152
9.10 Comparison of the average inference times and the MMACs of the plain classifiers with the best MRWA-FM's per dataset and classifier. . . . . 154
A.1 High-level architecture of the location-unaware TSMixer. . . . . 166
A.2 High-level architecture of the location-aware Transformer model. . . 166
A.3 Average inference times of MRWA-FM. . . . . 167
A.4 MMACs of MRWA-FM after TensorFlow Lite conversion. . . . . 170

# List of Tables


2.1 Activity hierarchy with related terms from literature. . . . . 15
2.2 The WHAR datasets used by this work. . . . . 26
3.1 Adjacency matrices of the graphs from Figure 3.1. . . . . 32
3.2 State-of-the-art GNNs for WHAR. . . . . 38
4.1 Pre-training objectives, network architectures, validated downstream tasks, and the existence of cross-dataset validation in self-supervised WHAR models. . . . . 50
5.1 Model configuration. . . . . 63
5.2 Graph configuration. . . . . 63
5.3 Vertex feature dimensions $X \in \mathbb{R}^{w \times f}$ and window durations per dataset. 64
5.4 Model sizes per dataset and channel count $c$ in $\mathrm{kB}$. . . . . 65
5.5 Comparison of space requirements in $\mathrm{kB}$ for storing sparse adjacency matrices in COO format with different neighborhoods versus dense matrices. . . . . 66
5.6 The average accuracies and macro F1-scores for different channels and neighborhoods on KU-HAR. . . . . 68
5.7 The average accuracies and macro F1-scores for different channels and neighborhoods on PAMAP2. . . . . 69
5.8 The average accuracies and macro F1-scores for different channels and neighborhoods on UCI-HAR. . . . . 71
5.9 The average accuracies and macro F1-scores for different channels and neighborhoods on Daphnet. . . . . 72
6.1 Graph and model configuration. . . . . 84
6.2 Average micro-activity similarity per dataset. . . . . 87
6.3 The prefix sequences of the table tennis exercise $M_T = 1230$ along with its possible and impossible links. . . . . 89
6.4 Class distributions of Cooking and TT-Strokes datasets. . . . . 90

7.1 Properties of the eight selected datasets. . . . . 103
7.2 Total number of windows per sensor location. . . . . 104
7.3 Differences between location-invariant and multi-location processing of WHAR signals. . . . . 105
8.1 Model configuration. . . . . 117
8.2 Pre-training and fine-tuning parameters of Dual-View FM. . . . . 118
8.3 Average pre-training performance of the location-unaware (LU) Dual-View FMs, their location-aware (LA) counterparts, TS-Mixer, Transformer, and TimeGPT-1 on the CSM pre-training task. . . . . 121
8.4 Average pre-training performance of the location-unaware (LU) Dual-View FMs, their location-aware (LA) counterparts, TS-Mixer, and Transformer on the MSM pre-training task. . . . . 123
8.5 Average fine-tuning performances across all holdout dataset. . . . . 127
9.1 Model configuration. . . . . 142
9.2 The number of MRWA-FM's weights per hyperparameter configuration and number of sensor locations. . . . . 143
9.3 Pre-training and fine-tuning parameters of MRWA-FM. . . . . 143
9.4 Average macro F1-scores of MRWA-FM, MLCNNwav, and WLF per dataset. . . . . 156

# List of Relevant Publications


[1] Christoph Wieland and Victor Pankratius. TinyGraphHAR: Enhancing Human Activity Recognition With Graph Neural Networks. In *2023 IEEE World AI IoT Congress (AIIoT)*, pages 47–54, June 2023.

[2] Christoph Wieland and Victor Pankratius. Detection and Validation of Macro-Activities in Human Inertial Signals Using Graph Link Prediction. *Sensors*, 24(4), 2024.

[3] Christoph Wieland and Victor Pankratius. Inertial Signal Forecasting With Foundation Model Techniques. *IEEE Sensors Journal*, 25(19):36843–36854, Oct 2025.

[4] Christoph Wieland and Victor Pankratius. Enhancing Sensor-Based Human Activity Recognition With Multiresolution Wavelet-Attention. *IEEE Sensors Journal*, 25(14):26920–26930, July 2025.

# Bibliography

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`, 2015.

[6] Alireza Abedin, Farbod Motlagh, Qinfeng Shi, Hamid Rezatofighi, and Damith Ranasinghe. Towards deep clustering of human activities from wearables. In *Proceedings of the 2020 ACM International Symposium on Wearable Computers*, ISWC '20, pages 1–6, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Md Atiqur Rahman Ahad, Anindya Das Antar, and Masud Ahmed. *Sensor-Based Human Activity Recognition: Challenges Ahead*, pages 175–189. Springer International Publishing, Cham, 2021.

[8] Abrar Ahmed, Harish Haresamudram, and Thomas Ploetz. Clustering of human activities from wearables by adopting nearest neighbors. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, ISWC '22, pages 1–5, New York, NY, USA, 2022. Association for Computing Machinery.

[9] Sayeda Alia, Paula Lago, Shingo Takeda, Kohei Adachi, Brahim Benaissa, Md Atiqur Rahman Ahad, and Sozo Inoue. *Summary of the Cooking Activity Recognition Challenge*, pages 1–13. Springer Singapore, Singapore, 01 2021.

[10] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN 2013: 21st European Symposium on Artificial Neural Networks, Computational Intelligence And Machine Learning*, pages 437–442, 2013.

[11] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[12] Marc Bachlin, Meir Plotnik, Daniel Roggen, Inbal Maidan, Jeffrey M. Hausdorff, Nir Giladi, and Gerhard Troster. Wearable assistant for parkinson's disease patients with the freezing of gait symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):436–446, 2010.

[13] Oresti Banos, Rafael Garcia, Juan A Holgado-Terriza, Miguel Damas, Hector Pomares, Ignacio Rojas, Alejandro Saez, and Claudia Villalonga. mhealthdroid: a novel framework for agile development of mobile health applications. In *Ambient Assisted Living and Daily Activities: 6th International Work-Conference, IWAAL 2014, Belfast, UK, December 2-5, 2014. Proceedings 6*, pages 91–98. Springer, 2014.

[14] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive Computing*, pages 1–17, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[15] Billur Barshan and Aras Yurtman. Classifying daily and sports activities invariantly to the positioning of wearable motion sensor units. *IEEE Internet of Things Journal*, 7(6):4801–4815, 2020.

[16] Jr. Biden, Joseph R. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence. https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence/, 2023.

[17] Marius Bock, Alexander Hölzemann, Michael Moeller, and Kristof Van Laerhoven. Improving deep learning for har with shallow lstms. In *Proceedings of the 2021 ACM International Symposium on Wearable Computers*, ISWC '21, pages 7–12, New York, NY, USA, 2021. Association for Computing Machinery.

[18] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa,

Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022.

[19] Bosch Sensortec GmbH. BMI270. `https://www.bosch-sensortec.com/media/boschsensortec/downloads/product_flyer/bst-bmi270-fl000.pdf`, 2019. Last access: January 28, 2025.

[20] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

[21] Business Wire. Absatz von smartphones weltweit in den jahren 2009 bis 2023 (in millionen stück) [graph]. `https://de.statista.com/statistik/daten/studie/173049/umfrage/weltweiter-absatz-von-smartphones-seit-2009/`, Jan 2024. Last access: October 14, 2024.

[22] Cătălina Cangea, Petar Veličković, Nikola Jovanović, Thomas Kipf, and Pietro Liò. Towards sparse hierarchical graph classifiers, 2018.

[23] Jie Cao, Youquan Wang, Haicheng Tao, and Xiang Guo. Sensor-based human activity recognition using graph lstm and multi-task classification model. *ACM Trans. Multimedia Comput. Commun. Appl.*, 18(3s), Oct 2022.

[24] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Comput. Surv.*, 54(4), May 2021.

[25] Ling Chen, Yingsong Luo, Liangying Peng, Rong Hu, Yi Zhang, and Shenghuan Miao. A multi-graph convolutional network based wearable human activity recognition method using multi-sensors. *Applied Intelligence*, 53(23):28169–28185, Dec 2023.

[26] Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecasting. *Transactions on Machine Learning Research*, 2023.

[27] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

[28] François Chollet. Transfer learning & fine-tuning. `https://keras.io/guides/transfer_learning/`, 2020. Last access: March 1, 2024.

[29] François Chollet et al. Keras. `https://keras.io`, 2015.

[30] Committee on Data for Science and Technology (CODATA). Fundamental physical constants: standard acceleration of gravity. `https://physics.nist.gov/cgi-bin/cuu/Value?gn`, 2018. Last access: October 11, 2023.

[31] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction To Algorithms*. MIT Electrical Engineering and Computer Science. MIT Press, 2nd edition, 2001.

[32] Abdelghani Dahou, Mohammed A. A. Al-Qaness, Mohamed Abd Elaziz, and Ahmed M. Helmi. Mlcnnwav: Multilevel convolutional neural network with wavelet transformations for sensor-based human activity recognition. *IEEE Internet of Things Journal*, 11(1):820–828, 2024.

[33] Abdelghani Dahou, Mohammed A.A. Al-qaness, Mohamed Abd Elaziz, and Ahmed Helmi. Human activity recognition in ioht applications using arithmetic optimization algorithm and deep learning. *Measurement*, 199:111445, 2022.

[34] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, 1988.

[35] Carl de Boor. *A Practical Guide to Splines*. Applied Mathematical Sciences. Springer New York, NY, revised edition, 1978.

[36] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 3844–3852, Red Hook, NY, USA, 2016. Curran Associates Inc.

[37] Shohreh Deldari, Dimitris Spathis, Mohammad Malekzadeh, Fahim Kawsar, Flora D. Salim, and Akhil Mathur. Crossl: Cross-modal self-supervised learning for time-series through latent masking. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, pages 152–160, New York, NY, USA, 2024. Association for Computing Machinery.

[38] Shohreh Deldari, Hao Xue, Aaqib Saeed, Daniel V. Smith, and Flora D. Salim. Cocoa: Cross modality contrastive learning for sensor data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(3), sep 2022.

[39] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[40] Reinhard Diestel. *Graph Theory*. Springer Berlin, Heidelberg, 5th edition, 2017.

[41] Iveta Dirgová Luptáková, Martin Kubovčík, and Jiří Pospíchal. Wearable sensor-based human activity recognition with transformer model. *Sensors*, 22(5), 2022.

[42] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[43] European Parliament and Council. Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down

harmonised rules on artificial intelligence and amending regulations (ec) no 300/2008, (eu) no 167/2013, (eu) no 168/2013, (eu) 2018/858, (eu) 2018/1139 and (eu) 2019/2144 and directives 2014/90/eu, (eu) 2016/797 and (eu) 2020/1828 (artificial intelligence act) (text with eea relevance). https://eur-lex.europa.eu/eli/reg/2024/1689/oj, 2024.

[44] Abu Zaher Md Faridee, Md Abdullah Al Hafiz Khan, Nilavra Pathak, and Nirmalya Roy. Augtoact: scaling complex human activity recognition with few labels. In *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '19, pages 162–171, New York, NY, USA, 2020. Association for Computing Machinery.

[45] Vitor Fortes Rey, Sungho Suh, and Paul Lukowicz. Learning from the best: Contrastive representations learning across sensor locations for wearable activity recognition. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, ISWC '22, pages 28–32, New York, NY, USA, 2022. Association for Computing Machinery.

[46] Hongyang Gao and Shuiwang Ji. Graph u-nets, 2019.

[47] Xin Gao, Tianheng Qiu, Xinyu Zhang, Hanlin Bai, Kang Liu, Xuan Huang, Hu Wei, Guoying Zhang, and Huaping Liu. Efficient multi-scale network with learnable discrete wavelet transform for blind motion deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2733–2742, June 2024.

[48] Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1, 2024. API accessible via: https://dashboard.nixtla.io/.

[49] Wen Ge, Guanyi Mou, Emmanuel O. Agu, and Kyumin Lee. Heterogeneous hyper-graph neural networks for context-aware human activity recognition. In *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 350–354, Mar 2023.

[50] Wen Ge, Guanyi Mou, Emmanuel O. Agu, and Kyumin Lee. Deep heterogeneous contrastive hyper-graph learning for in-the-wild context-aware human activity recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 7(4), Jan 2024.

[51] Dirk Gehrig. *Automatic Recognition of Concurrent and Coupled Human Motion Sequences*. PhD thesis, Karlsruher Institut für Technologie (KIT), 2015.

[52] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[54] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[55] Daniele Grattarola and Cesare Alippi. Graph neural networks in tensorflow and keras with spektral, 2020.

[56] Fuqiang Gu, Mu-Huan Chung, Mark Chignell, Shahrokh Valaee, Baoding Zhou, and Xue Liu. A survey on deep learning for human activity recognition. *ACM Comput. Surv.*, 54(8), October 2021.

[57] Alfred Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910.

[58] Jindong Han, Yuan He, Juan Liu, Qianqian Zhang, and Xiaojun Jing. Graphconvlstm: Spatiotemporal learning for activity recognition with

wearable sensors. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2019.

[59] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[60] Harish Haresamudram, David V. Anderson, and Thomas Plötz. On the role of features in human activity recognition. In *Proceedings of the 2019 ACM International Symposium on Wearable Computers*, ISWC '19, pages 78–88, New York, NY, USA, 2019. Association for Computing Machinery.

[61] Harish Haresamudram, Apoorva Beedu, Varun Agrawal, Patrick L. Grady, Irfan Essa, Judy Hoffman, and Thomas Plötz. Masked reconstruction based self-supervision for human activity recognition. In *Proceedings of the 2020 ACM International Symposium on Wearable Computers*, ISWC '20, pages 45–49, New York, NY, USA, 2020. Association for Computing Machinery.

[62] Harish Haresamudram, Irfan Essa, and Thomas Plötz. Contrastive predictive coding for human activity recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(2), jun 2021.

[63] Debapriya Hazra and Yung-Cheol Byun. Synsiggan: Generative adversarial networks for synthetic biomedical signal generation. *Biology*, 9(12), 2020.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770 – 778, June 2016.

[65] Yulong He, Hanming Huang, Yezheng Wu, and Guanglei Zhu. Research on abnormal behavior recognition of the elderly based on spatial-temporal feature fusion. In *Proceedings of the 3rd International Symposium on Artificial Intelligence for Medicine Sciences*, ISAIMS '22, pages 85–92, New York, NY, USA, 2022. Association for Computing Machinery.

[66] Zhen-Yu He and Lian-Wen Jin. Activity recognition from acceleration data using ar model representation and svm. In *2008 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2245–2250, July 2008.

[67] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[68] Alexander Hoelzemann and Kristof Van Laerhoven. Digging deeper: towards a better understanding of transfer learning for human activity recognition. In *Proceedings of the 2020 ACM International Symposium on Wearable Computers*, ISWC '20, pages 50–54, New York, NY, USA, 2020. Association for Computing Machinery.

[69] Zhiqing Hong, Zelong Li, Shuxin Zhong, Wenjun Lyu, Haotian Wang, Yi Ding, Tian He, and Desheng Zhang. Crosshar: Generalizing cross-dataset human activity recognition via hierarchical self-supervised pretraining. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(2), May 2024.

[70] Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):10173–10196, Aug 2023.

[71] Wenbo Huang, Lei Zhang, Wenbin Gao, Fuhong Min, and Jun He. Shallow convolutional neural networks for human activity recognition using wearable sensors. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2021.

[72] Yi Huang, Xiaoshan Yang, Junyu Gao, Jitao Sang, and Changsheng Xu. Knowledge-driven egocentric multimodal activity recognition. *ACM Trans. Multimedia Comput. Commun. Appl.*, 16(4), Dec 2020.

[73] Ron J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 3rd edition, 2021.

[74] IDC. Absatz von wearables weltweit in den jahren 2014 bis 2023 (in millionen stück) [graph]. https://de.statista.com/statistik/daten/studie/515723/umfrage/absatz-von-wearables-weltweit/, Mar 2024. Last access: October 14, 2024.

[75] Ozlem Durmaz Incel, Mustafa Kose, and Cem Ersoy. A review and taxonomy of activity recognition on mobile phones. *BioNanoScience*, 3(2):145–171, 2013.

[76] Md. Rabiul Islam, Shuji Sakamoto, Yoshihiro Yamada, Andrew W. Vargo, Motoi Iwata, Masakazu Iwamura, and Koichi Kise. Self-supervised learning for reading activity classification. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(3), sep 2021.

[77] Yash Jain, Chi Ian Tang, Chulhong Min, Fahim Kawsar, and Akhil Mathur. Collossl: Collaborative self-supervised learning for human activity recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(1), mar 2022.

[78] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

[79] Ismael Espinoza Jaramillo, Channabasava Chola, Jin-Gyun Jeong, Ji-Heon Oh, Hwanseok Jung, Jin-Hyuk Lee, Won Hee Lee, and Tae-Seong Kim. Human activity prediction based on forecasted imu activity signals by sequence-to-sequence deep neural networks. *Sensors*, 23(14), 2023.

[80] Bulat Khaertdinov, Esam Ghaleb, and Stylianos Asteriadis. Contrastive self-supervised learning for sensor-based human activity recognition. In *2021 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–8, Aug 2021.

[81] Aftab Khan, Sebastian Mellor, Rachel King, Balazs Janko, William Harwin, R. Simon Sherratt, Ian Craddock, and Thomas Plötz. Generalized and efficient skill assessment from imu data with applications in gymnastics

and medical training. *ACM Trans. Comput. Healthcare*, 2(1), December 2021.

[82] Taehwan Kim, Jeongho Park, Juwon Lee, and Jooyoung Park. Predicting human motion signals using modern deep learning techniques and smartphone sensors. *Sensors*, 21(24), 2021.

[83] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

[84] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision, 2021.

[85] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[86] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[87] Prabhat Kumar and S Suresh. Flaap: An open human activity recognition (har) dataset for learning and finding the associated activity patterns. *Procedia Computer Science*, 212:64–73, 2022.

[88] Paula Lago, Shingo Takeda, Kohei Adachi, Sayeda Shamma Alia, Moe Matsuki, Brahim Benai, Sozo Inoue, and Francois Charpillet. Cooking activity dataset with macro and micro activities, 2020.

[89] Oscar D. Lara and Miguel A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, March 2013.

[90] Gregory R. Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O'Leary. Pywavelets: A python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237, 2019.

[91] Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts, 2023.

[92] Maurizio Leotta, Andrea Fasciglione, and Alessandro Verri. Daily living activity recognition using wearable devices: A features-rich dataset and a novel approach. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part II*, pages 171–187. Springer, 2021.

[93] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[94] Qiufu Li, Linlin Shen, Sheng Guo, and Zhihui Lai. Wavelet integrated cnns for noise-robust image classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7243–7252, 2020.

[95] Shuangjian Li, Tao Zhu, Mingxing Nie, Huansheng Ning, Zhenyu Liu, and Liming Chen. P2lhap:wearable sensor-based human activity recognition, segmentation and forecast through patch-to-label seq2seq transformer, 2024.

[96] Xiaomin Li, Vangelis Metsis, Huangyingrui Wang, and Anne Hee Hiong Ngu. Tts-gan: A transformer-based time-series generative adversarial network. In Martin Michalowski, Syed Sibte Raza Abidi, and Samina Abidi, editors, *Artificial Intelligence in Medicine*, pages 133–143, Cham, 2022. Springer International Publishing.

[97] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.

[98] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[99] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 125–143, Cham, 2016. Springer International Publishing.

[100] Tianzheng Liao, Jinjin Zhao, Yushi Liu, Kamen Ivanov, Jing Xiong, and Yan Yan. Deep transfer learning with graph neural network for sensor-based human activity recognition. In *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2445–2452, Dec 2022.

[101] Hui Liu, Yale Hartmann, and Tanja Schultz. Motion units: Generalized sequence modeling of human activities for sensor-based activity recognition. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 1506–1510, 2021.

[102] Mengna Liu, Dong Xiang, Xu Cheng, Xiufeng Liu, Dalin Zhang, Shengyong Chen, and Christian S. Jensen. Disentangling imperfect: A wavelet-infused multilevel heterogeneous network for human activity recognition in flawed wearable sensor data, 2024.

[103] Shengzhong Liu, Shuochao Yao, Yifei Huang, Dongxin Liu, Huajie Shao, Yiran Zhao, Jinyang Li, Tianshi Wang, Ruijie Wang, Chaoqi Yang, and Tarek Abdelzaher. Handling missing sensors in topology-aware iot applications with gated graph neural network. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(3), Sep 2020.

[104] Aleksej Logacjov. Self-supervised learning for accelerometer-based human activity recognition: A survey. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(4), November 2024.

[105] Fei Luo, Salabat Khan, Yandao Huang, and Kaishun Wu. Binarized neural network for edge intelligence of sensor-based human activity recognition. *IEEE Transactions on Mobile Computing*, 22(3):1356–1368, 2023.

[106] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

[107] S. Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Elsevier Science, 2008.

[108] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[109] Kurt Mehlhorn and Peter Sanders. *Algorithms and data structures: The basic toolbox*. Springer Berlin, Heidelberg, 1st edition, 2008.

[110] Yaowen Mei, Ting Jiang, Xue Ding, Yi Zhong, Sai Zhang, and Yang Liu. Wiwave: Wifi-based human activity recognition using the wavelet integrated cnn. In *2021 IEEE/CIC International Conference on Communications in China*, pages 100–105, 2021.

[111] Alfred Mertins. *Signaltheorie: Grundlagen der Signalbeschreibung, Filterbänke, Wavelets, Zeit-Frequenz-Analyse, Parameter- und Signalschätzung*. Springer Fachmedien Wiesbaden, Wiesbaden, 5th edition, 2023.

[112] Shenghuan Miao, Ling Chen, and Rong Hu. Spatial-temporal masked autoencoder for multi-device wearable human activity recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 7(4), jan 2024.

[113] Shenghuan Miao, Ling Chen, Rong Hu, and Yingsong Luo. Towards a dynamic inter-sensor correlations learning framework for multi-sensor-based wearable human activity recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(3), Sep 2022.

[114] Gabriel Michau, Gaetan Frusque, and Olga Fink. Fully learnable deep wavelet transform for unsupervised monitoring of high-frequency time series. *Proceedings of the National Academy of Sciences*, 119(8), 2022.

[115] Abduallah Mohamed, Fernando Lejarza, Stephanie Cahail, Christian Claudel, and Edison Thomaz. Har-gcnn: Deep graph cnns for human activity recognition from highly unlabeled mobile sensor data. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 124–126, Mar 2022.

[116] Riktim Mondal, Debadyuti Mukherjee, Pawan Kumar Singh, Vikrant Bhateja, and Ram Sarkar. A new framework for smartphone sensor-based human activity recognition using graph neural network. *IEEE Sensors Journal*, 21(10):11461–11468, May 2021.

[117] José Manuel Negrete Ramírez and Yudith Cardinale. Activities of daily living detection on healthcare: A categorization. In *Proceedings of the 7th International Workshop on Sensor-Based Activity Recognition and Artificial Intelligence*, iWOAR '22, New York, NY, USA, 2023. Association for Computing Machinery.

[118] Newgy Industries, Inc. Robo-Pong 3050XL Owner's Manual. `https://cdn.shopify.com/s/files/1/2677/3302/files/3050XL_Owners_Manual_1.5.21.pdf`, 2018. Last access: March 22, 2021.

[119] Aixin Nian, Xianqiang Zhu, Xiang Xu, Xueqin Huang, Fei Wang, and Yu Zhao. Hgcnn: Deep graph convolutional network for sensor-based human activity recognition. In *2022 8th International Conference on Big Data and Information Analytics (BigDIA)*, pages 422–427, Aug 2022.

[120] Nixtla. About timegpt. `https://docs.nixtla.io/docs/getting-started-about_timegpt`. Last access: January 20, 2025.

[121] Juri Opitz and Sebastian Burst. Macro F1 and Macro F1, 2021.

[122] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1), 2016.

[123] Jinxing Pan, Xiaoshan Yang, Yi Huang, and Changsheng Xu. Few-shot egocentric multimodal activity recognition. In *Proceedings of the 3rd ACM International Conference on Multimedia in Asia*, MMAsia '21, New York, NY, USA, 2022. Association for Computing Machinery.

[124] Sangjun Park, Hyung Ok Lee, Yu Min Hwang, Seok-Kap Ko, and Byung-Tak Lee. Enhanced prediction model for human activity using an end-to-end approach. *IEEE Internet of Things Journal*, 10(7):6031–6041, April 2023.

[125] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[126] Olena Pavliuk, Myroslav Mishchuk, and Christine Strauss. Transfer learning approach for human activity recognition based on continuous wavelet transform. *Algorithms*, 16(2), 2023.

[127] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[128] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

[129] Xin Qin, Jindong Wang, Shuo Ma, Wang Lu, Yongchun Zhu, Xing Xie, and Yiqiang Chen. Generalizable low-resource activity recognition with diverse and discriminative representation learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23,

pages 1943–1953, New York, NY, USA, 2023. Association for Computing Machinery.

[130] Alec Radford. Improving language understanding by generative pre-training, 2018.

[131] Setareh Rahimi Taghanaki, Michael J Rainbow, and Ali Etemad. Self-supervised human activity recognition by learning to predict cross-dimensional motion. In *Proceedings of the 2021 ACM International Symposium on Wearable Computers*, ISWC '21, pages 23–27, New York, NY, USA, 2021. Association for Computing Machinery.

[132] Daniel Recoskie. *Learning sparse orthogonal wavelet filters*. PhD thesis, University of Waterloo, 2018.

[133] Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th International Symposium on Wearable Computers*, pages 108–109, 2012.

[134] Aaqib Saeed, Tanir Ozcelebi, and Johan Lukkien. Multi-task self-supervised learning for human activity detection. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(2), jun 2019.

[135] Abhishek Sarkar, Tanmay Sen, and Ashis Kumar Roy. Grafehty: Graph neural network using federated learning for human activity recognition. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1124–1129, Dec 2021.

[136] Roshwin Sengupta, Ilia Polian, and John P. Hayes. Wavelet transform assisted neural networks for human activity recognition. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1254–1258, May 2022.

[137] Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations*, 2021.

[138] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul JM Havinga. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors*, 16(4):426, 2016.

[139] Muhammad Shoaib, Stephan Bosch, Hans Scholten, Paul JM Havinga, and Ozlem Durmaz Incel. Towards detection of bad habits by fusing smartphone and smartwatch sensors. In *2015 IEEE international conference on pervasive computing and communication workshops (PerCom Workshops)*, pages 591–596. IEEE, 2015.

[140] Niloy Sikder, Md Atiqur Rahman Ahad, and Abdullah-Al Nahid. Human action recognition based on a sequential deep learning model. In *2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–7, 2021.

[141] Niloy Sikder and Abdullah-Al Nahid. Ku-har: An open dataset for heterogeneous human activity recognition. *Pattern Recognition Letters*, 146:46–54, 2021.

[142] Dimitris Spathis, Ignacio Perez-Pozuelo, Soren Brage, Nicholas J. Wareham, and Cecilia Mascolo. Self-supervised transfer learning of physiological representations from free-living wearable data. In *Proceedings of the Conference on Health, Inference, and Learning*, CHIL '21, pages 69–78, New York, NY, USA, 2021. Association for Computing Machinery.

[143] Robyn Speer and Catherine Havasi. *ConceptNet 5: A Large Semantic Network for Relational Knowledge*, pages 161–176. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[144] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[145] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, June 2016.

[146] Timo Sztyler and Heiner Stuckenschmidt. On-body localization of wearable devices: An investigation of position-aware activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. IEEE, 2016.

[147] Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, Soren Brage, Nick Wareham, and Cecilia Mascolo. Selfhar: Improving human activity recognition through self-training with unlabeled data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(1), mar 2021.

[148] Qi Teng, Kun Wang, Lei Zhang, and Jun He. The layer-wise training convolutional neural networks using local loss for sensor-based human activity recognition. *IEEE Sensors Journal*, 20(13):7265–7274, 2020.

[149] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[150] Petar Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, 2023.

[151] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[152] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0

Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[153] Thi Hong Vuong, Tung Doan, and Atsuhiro Takasu. Deep wavelet convolutional neural networks for multimodal human activity recognition using wearable inertial sensors. *Sensors*, 23(24), 2023.

[154] Chongyang Wang, Yuan Gao, Akhil Mathur, Amanda C. De C. Williams, Nicholas D. Lane, and Nadia Bianchi-Berthouze. Leveraging activity recognition to enable protective behavior detection in continuous data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(2), Jun 2021.

[155] Yan Wang, Xin Wang, Hongmei Yang, Yingrui Geng, Hongnian Yu, Ge Zheng, and Liang Liao. Mhagnn: A novel framework for wearable sensor-based human activity recognition combining multi-head attention and graph neural networks. *IEEE Transactions on Instrumentation and Measurement*, 72:1–14, 2023.

[156] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3029–3038, July 2017.

[157] Yucheng Wang, Min Wu, Xiaoli Li, Lihua Xie, and Zhenghua Chen. Multivariate time-series representation learning via hierarchical correlation pooling boosted graph neural network. *IEEE Transactions on Artificial Intelligence*, 5(1):321–333, Jan 2024.

[158] Christoph Wieland. Domain knowledge infusion in machine learning for digital signal processing applications : An in-depth case study on table tennis stroke recognition. Master's thesis, Karlsruher Institut für Technologie (KIT), 2021.

[159] Christoph Wieland and Victor Pankratius. Domain-Knowledge Enhanced Machine Learning for Table Tennis Stroke Characterization. In *2023 IEEE World AI IoT Congress (AIIoT)*, pages 590–597, June 2023.

[160] Moritz Wolter, Shaohui Lin, and Angela Yao. Neural network compression via learnable wavelet transforms. In *Artificial Neural Networks and Machine Learning – ICANN 2020*, pages 39–51, Cham, 2020. Springer International Publishing.

[161] Qingyu Wu, Jianfei Shen, Feiyi Fan, Yang Gu, Chenyang Xu, and Yiqiang Chen. Auto-augmentation contrastive learning for wearable-based human activity recognition. In *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1537–1544, Dec 2023.

[162] Kun Xia, Jianguang Huang, and Hanyu Wang. Lstm-cnn architecture for human activity recognition. *IEEE Access*, 8:56855–56866, 2020.

[163] Lei Xie, Xu Dong, Wei Wang, and Dawei Huang. Meta-activity recognition: A wearable approach for logic cognition-based activity sensing. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017.

[164] Huatao Xu, Pengfei Zhou, Rui Tan, and Mo Li. Practically adopting human activity recognition. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '23, New York, NY, USA, 2023. Association for Computing Machinery.

[165] Huatao Xu, Pengfei Zhou, Rui Tan, Mo Li, and Guobin Shen. Limu-bert: Unleashing the potential of unlabeled data for imu sensing applications. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, SenSys '21, pages 220–233, New York, NY, USA, 2021. Association for Computing Machinery.

[166] Xuhai Xu, Jun Gong, Carolina Brum, Lilian Liang, Bongsoo Suh, Shivam Kumar Gupta, Yash Agarwal, Laurence Lindsey, Runchang Kang, Behrooz Shahsavari, Tu Nguyen, Heriberto Nieto, Scott E Hudson, Charlie Maalouf, Jax Seyed Mousavi, and Gierad Laput. Enabling hand gesture customization on wrist-worn devices. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery.

[167] Jianjun Yan, Xueqiang Wang, Jiangtao Shi, and Shuai Hu. Skeleton-based fall detection with multiple inertial sensors using spatial-temporal graph convolutional networks. *Sensors*, 23(4), 2023.

[168] Jing Yang, Tianzheng Liao, Jingjing Zhao, Yan Yan, Yichun Huang, Zhijia Zhao, Jing Xiong, and Changhong Liu. Domain adaptation for sensor-based human activity recognition with a graph convolutional network. *Mathematics*, 12(4), 2024.

[169] Raneen Younis, Zahra Ahmadi, Abdul Hakmeh, and Marco Fisichella. Flames2graph: An interpretable federated multivariate time series classification framework. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pages 3140–3150, New York, NY, USA, 2023. Association for Computing Machinery.

[170] Han Yu and Akane Sano. Semi-supervised learning for wearable-based momentary stress detection in the wild. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 7(2), jun 2023.

[171] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023.

[172] Mi Zhang and Alexander A Sawchuk. Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, pages 1036–1043, 2012.

[173] Yuxin Zhang, Yiqiang Chen, Jindong Wang, and Zhiwen Pan. Unsupervised deep anomaly detection for multi-sensor time-series signals. *IEEE Transactions on Knowledge and Data Engineering*, 35(2):2118–2132, Feb 2023.

[174] Haibin Zhao, Yexu Zhou, Till Riedel, Michael Hefenbrock, and Michael Beigl. Improving human activity recognition models by learnable sparse wavelet layer. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, ISWC '22, pages 84–88, New York, NY, USA, 2022. Association for Computing Machinery.

[175] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns, 2020.

[176] Ziyun Zhao, Aohua Song, Qingyun Xiong, Siyu Zheng, and Junqi Guo. Mmst-gcn+: A multi-modal st-gcn+ for computer-aided education by introducing a sensor skeleton. In *2023 8th International Conference on Image, Vision and Computing (ICIVC)*, pages 913–922, July 2023.

[177] Yexu Zhou, Haibin Zhao, Yiran Huang, Till Riedel, Michael Hefenbrock, and Michael Beigl. Tinyhar: A lightweight deep learning model designed for human activity recognition. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, ISWC '22, pages 89–93, New York, NY, USA, 2022. Association for Computing Machinery.