Graph Representation Learning in Smart Environments

Sensor Embeddings with Graph Neural Networks

MARCO ANDRES TELLO GUERRERO

university of groningen / faculty of science and engineering / bernoulli institute

The research presented in this dissertation was performed in the Distributed Systems Group of the Bernoulli Institute for Mathematics, Computer Science, and Artificial Intelligence of the University of Groningen, The Netherlands.

The work is supported by the University of Groningen, and partially funded by the project DiTEC: Digital Twin for Evolutionary Changes in Water Networks (NWO 19454).

Published by: University of Groningen
Groningen, The Netherlands

Cover designed by: Andrés Tello

university of groningen

Graph Representation Learning in Smart Environments

Sensor Embeddings with Graph Neural Networks

PhD Thesis

to obtain the degree of PhD of the
University of Groningen
on the authority of the
Rector Magnificus Prof. J.M.A. Scherpen
and in accordance with
the decision by the College of Deans.

This thesis will be defended in public on

Tuesday 3 February 2026 at 12.45 hours

by

MARCO ANDRES TELLO GUERRERO

born on 6 April 1982
in Machala, Ecuador

Supervisor
Prof. A. Lazovik

Co-Supervisor
Dr. V. Degeler

Assessment committee
Prof. G. Azzopardi
Prof. M. Aiello
Prof. O. Durmaz-Incel

To my Family, my Parents, and my Brother.

Acknowledgments

This dissertation involved years of intense work, eureka moments celebrating ideas that led to achievements, periods of heavy stress while struggling with less successful ideas, and many other emotions and experiences. Looking back, I feel compelled to express my heartfelt gratitude to all the people who made this work possible.

I am deeply grateful to the person who trusted in me and believed I was the right person for this work, my co-supervisor Dr. Victoria Degeler. Her continuous support and guidance were instrumental in shaping both my research direction and my development as a researcher. I am especially thankful for her unwavering empathy during difficult times outside of my research; her understanding was vital when personal circumstances threatened to overwhelm my academic focus. When Victoria later moved to another institution, Prof. Alexander Lazovik assumed a more direct supervisory role, which was crucial to the successful completion of this dissertation. I am sincerely thankful to Alexander for the increasing trust and responsibility he placed in me as the project progressed, particularly during the final stages of this work. I would also like to thank the members of the assessment committee for their time, insightful comments, and valuable feedback.

I would like to thank my colleagues at the Distributed Systems Group for making this journey so enjoyable. Thanks to Huy, Revin, Mahmoud, and Michel for the entertaining conversations during lunch. Special thanks go to Huy for his collaboration on different projects and several other academic duties. This fruitful collaboration led to impactful research outcomes that form a core part of this dissertation. Thank

you, Huy, for the insightful discussions during the projects on which we worked together.

I would like to thank my paranymphs, Michel and Huy, for helping me with the organization of my PhD defense. Everything would have been very difficult without your help.

A very special thanks to Evelyn, Petter, and Charlotte, my lovely family. This work would not have been possible without their emotional support. They were always there during stressful moments, even when my mood was not at its best. Thank you for your understanding when my work demanded time that I could not spend with you. Thanks to my parents, Carmen and Efrén, and to my brother, Rafael, for their constant support and for believing in me even when I doubted myself. Thanks to my friends, both my international and my Ecuadorian crew, for their time and support during those moments when I needed to disconnect from everything, step away from the world, and simply be myself while enjoying their company.

Agradecimientos

Esta tesis doctoral implicó años de trabajo intenso, momentos de “eureka!” celebrando ideas que dieron buenos resultados, períodos de mucho estrés cuando las ideas fueron menos exitosas, y muchas otras emociones y experiencias. Al mirar atrás, siento la necesidad de expresar mi más sincero agradecimiento a todas las personas que hicieron posible este trabajo.

Estoy profundamente agradecido con la persona que confió en mí y creyó que yo era la persona indicada para llevar a cabo este trabajo, mi codirectora de tesis, la Dra. Victoria Degeler. Su apoyo continuo y su orientación fueron fundamentales para dar forma tanto a la dirección de mi investigación como a mi desarrollo como investigador. Estoy especialmente agradecido por su gran empatía durante momentos difíciles fuera del ámbito académico; su comprensión fue vital cuando circunstancias personales amenazaron con sobrepasar mi enfoque en la investigación. Cuando Victoria posteriormente se trasladó a otra institución, el Prof. Alexander Lazovik asumió un rol de supervisión más directo, lo cual fue crucial para la finalización exitosa de esta tesis. Agradezco sinceramente a Alexander por la creciente confianza y responsabilidad que depositó en mí a medida que el proyecto avanzaba, particularmente durante las etapas finales de este trabajo. Asimismo, agradezco a los miembros del comité evaluador por su tiempo, sus acertados comentarios y su valiosa retroalimentación.

Quiero agradecer a mis colegas del Grupo de Sistemas Distribuidos por hacer de este camino una experiencia tan agradable. Gracias a Huy, Revin, Mahmoud y Michel por las entretenidas conversaciones durante

el almuerzo. Un agradecimiento especial a Huy por su colaboración en distintos proyectos y en diversas responsabilidades académicas. Esta colaboración fructífera dio lugar a resultados de investigación de gran impacto que constituyen una parte central de esta tesis. Gracias, Huy, por las valiosas y profundas discusiones durante los proyectos en los que trabajamos juntos.

Quiero agradecer a mis paraninfos, Michel y Huy, por su ayuda en la organización de mi defensa doctoral. Todo habría sido mucho más difícil sin su apoyo.

Un agradecimiento muy especial a Evelyn, Petter y Charlotte, mi querida familia. Este trabajo no habría sido posible sin su apoyo emocional. Siempre estuvieron ahí en los momentos de mayor estrés, incluso cuando mi estado de ánimo no era el mejor. Gracias por su comprensión cuando mi trabajo demandó tiempo que no pude pasar con ustedes. Gracias a mis padres, Carmen y Efrén, y a mi hermano, Rafael, por su apoyo constante y por creer en mí incluso cuando yo mismo dudé. Gracias a mis amigos, tanto a mi grupo internacional como a mi grupo ecuatoriano, por su tiempo y su apoyo en esos momentos en los que necesité desconectarme de todo, alejarme del mundo y simplemente ser yo mismo mientras disfrutaba de su compañía.

Summary

The problem addressed in this work is to learn better representations of graph-structured data to improve inference in smart environments. A data representation is considered *better* if it is useful for increasing the performance of an inference model on a specific task. State-of-the-art deep learning models undoubtedly excel at learning useful representations to solve complex prediction tasks. However, they are designed to operate on fixed structures such as grids (e.g., images) or sequences (e.g., text). On the contrary, graphs are of arbitrary size and structure. When dealing with graphs, there is no explicit notion of locality. We cannot define boundaries, determined by borders, as with images. Hence, it is not straightforward to slide a kernel along a graph as we do with convolutional neural networks over a grid or a sequence.

This thesis is organized in two parts, each focusing on a different problem domain, a different nature of the graph topology, and a different type of prediction task. Part I focuses on the domain of Human Activity Recognition (HAR). HAR is performed using wearable and mobile sensors data. In this setting, the graph topology is not explicit, i.e., it is not part of the input data. Hence, the graph needs to be inferred or constructed based on additional assumptions on the underlying data. The HAR problem is framed as a graph classification task. Thus, the model is trained to learn a single representation for entire graphs. Part II focuses on the water distribution networks (WDNs) domain. In this domain, the graph topology is explicitly given by the layout of the WDN. The problem addressed in this part is to reconstruct the pressure signal

for all nodes in the network from a few sparsely located sensors. In this case, the pressure reconstruction is framed as a node-level regression task. The model has to learn representations for each node in order to estimate the missing pressure values. The core chapters of this thesis are organized as follows.

In Chapter 2, we present the theoretical background for the topics and methods discussed in Parts I and II of this thesis.

In Chapter 3, we describe how conventional approaches to HAR can produce misleading results, making them impractical or even useless in real-world settings. This part of the research was crucial for applying an unbiased evaluation method to HAR and conducting a fair comparison between the HAR models studied in Chapter 4.

In Chapter 4, we propose an architecture for learning better representations from wearable sensor data to improve the performance of HAR models. We propose two graph constructions to capture the global and local dependencies in sensor data. Then, a contrastive learning approach maximizes mutual information between those graphs. We empirically validate our approach, showing that the learned representations significantly increase the accuracy of the classification model.

In Chapter 5, we combine physics-based modeling and GNNs to address the pressure estimation problem in WDNs. We propose a training strategy that relies on random sensor placement to increase the robustness of the model to unexpected sensor location changes. In addition, we propose a realistic evaluation protocol that considers real temporal patterns and noise injection to mimic the uncertainties inherent to real-world scenarios. As a result, GATRes, a new state-of-the-art model for pressure estimation, is available.

In Chapter 6, we discuss how current deep learning methods for graphs, tailored to the water domain, can be the enablers of Graph Foundation Models for WDNs. We discuss how to go from conception to actual implementation of this type of model. We discuss the challenges and

propose the strategies to its realization, from data generation, to model training, to model adaptation to solve WDNs-specific tasks.

Finally, in Chapter 7, we present the concluding remarks and the future directions of this research.

Samenvatting

Het probleem waar dit werk zich op richt, is het leren van betere representaties van data met een graafstructuur om de inference in slimme omgevingen te verbeteren. Een representatie van de data wordt als beter beschouwd als deze bijdraagt aan het verhogen van de prestaties van een inference-model voor een specifieke taak. De nieuwste deep learning-modellen blinken ongetwijfeld uit in het leren van goede representaties voor het doen van complexe voorspellingen. Ze zijn echter ontworpen om met vaste structuren te werken, zoals rasters (bijvoorbeeld afbeeldingen) of reeksen (zoals tekst). Grafen, daarentegen, hebben een willekeurige grootte en structuur. Ook is er geen expliciet besef van lokaliteit, en is het niet mogelijk om randen te bepalen op basis van lijnen zoals bij afbeeldingen. Om die reden is het niet eenvoudig om een kernel over een graaf te verschuiven zoals gebeurt met convolutionele neurale netwerken bij rasters en reeksen.

Dit proefschrift is opgesplitst in twee delen, waarbij elk deel zich richt op een ander toepassingsgebied, een andere topologie van de graaf, en een ander soort voorspellingstaak. Deel I houdt zich bezig met Human Activity Recognition (HAR). HAR wordt uitgevoerd op basis van draagbare en mobiele sensoren. In deze context wordt de topologie van de graaf niet expliciet gemaakt, wat betekent dat deze geen onderdeel is van de invoerdata. De graaf moet dus worden afgeleid of gecreëerd op basis van aanvullende aannames over de onderliggende data. Het HAR-probleem wordt geformuleerd als een classificatietaak voor grafen, waarbij het model wordt getraind om één enkele representatie te leren

voor een gehele graaf. Deel II richt zich op het domein van waterdistributienetwerken (WDN's). In dit domein wordt de topologie van de graaf expliciet bepaald door het ontwerp van het WDN. Het probleem dat in dit deel wordt aangepakt, is het reconstrueren van de waterdruk voor alle knopen in het netwerk op basis van een klein aantal sensoren die verspreid zijn over het gehele netwerk. In dit geval wordt het reconstrueren van de waterdruk geformuleerd als een regressietaak op het niveau van de knopen. Het model moet de representaties voor elke knoop leren om zo een schatting te kunnen maken van de ontbrekende drukwaarden. De hoofdstukken van dit proefschrift zijn als volgt onderverdeeld.

In hoofdstuk 2 presenteren we de theoretische achtergrondinformatie voor de onderwerpen en methoden die in deel I en II van dit proefschrift worden gebruikt.

In hoofdstuk 3 beschrijven we hoe traditionele benaderingen voor HAR misleidende resultaten kunnen opleveren, wat onpraktisch of zelfs onbruikbaar is voor echte toepassingen. Dit deel van het onderzoek was cruciaal voor het toepassen van een onbevooroordeelde evaluatiemethode voor HAR en voor een eerlijk vergelijk van de HAR-modellen die in hoofdstuk 4 worden bestudeerd.

In hoofdstuk 4 introduceren we een architectuur voor het leren van betere representaties uit de data van draagbare sensoren om zo de prestaties van HAR-modellen te verbeteren. We dragen methoden aan voor het maken van twee grafen die de globale en lokale verbanden uit de sensordata kunnen vastleggen. Vervolgens wordt er gebruik gemaakt van contrastive learning om de mutual information tussen deze grafen te maximaliseren. We valideren onze aanpak door middel van empirisch onderzoek en tonen aan dat de geleerde representaties de nauwkeurigheid van het classificatiemodel aanzienlijk verhogen.

In hoofdstuk 5 combineren we modelleringen die op de fysica gebaseerd zijn met GNN's om het probleem van het schatten van de druk in WDN's aan te pakken. We stellen een trainingsstrategie voor die gebruik maakt van willekeurige plaatsing van sensoren om het model robuuster te

maken tegen onverwachte veranderingen in de locaties van de sensoren. Daarnaast beschrijven we een realistisch evaluatieprotocol dat gebruikmaakt van echte tijdgerelateerde patronen en ruis toevoeging om de onzekerheden na te bootsen die inherent zijn aan reële scenario's. Het resultaat is GATRes, een nieuw state-of-the-art model voor het schatten van de druk.

In hoofdstuk 6 bespreken we hoe de huidige deep learning methoden voor grafen, afgestemd op het waterdomein, de weg vrij kunnen maken voor Graph Foundation Models voor WDN's. We leggen de stappen uit die nodig zijn om van een ontwerp tot een daadwerkelijke implementatie van dit soort modellen te komen. Ook behandelen we de uitdagingen en stellen we strategieën voor om deze modellen te realiseren, van het genereren van de data, tot het trainen van het model, tot aan het aanpassen van het model om WDN-specifieke taken op te lossen.

Tot slot presenteren we in hoofdstuk 7 de conclusies en mogelijke vervolgstappen voor dit onderzoek.

Resumen

El problema abordado en este trabajo consiste en aprender mejores representaciones de datos estructurados como grafos con el fin de mejorar los procesos de inferencia en entornos inteligentes. Una representación de los datos se considera *mejor* si resulta útil para incrementar el desempeño de un modelo de inferencia en una tarea específica. Los modelos de aprendizaje profundo de última generación destacan, indudablemente, por su capacidad para aprender representaciones útiles y resolver tareas complejas de predicción. Sin embargo, estos modelos están diseñados para operar sobre estructuras fijas, como cuadrículas (por ejemplo, imágenes) o secuencias (por ejemplo, texto). Por el contrario, los grafos tienen tamaño y estructura arbitrarios. Al trabajar con grafos, no existe una noción explícita de localidad. No es posible definir límites determinados por bordes, como ocurre en las imágenes. Por lo tanto, no resulta sencillo deslizar un kernel a lo largo de un grafo de la misma manera en que se hace con las redes neuronales convolucionales sobre una cuadrícula o una secuencia.

Esta tesis está organizada en dos partes, cada una enfocada en un problema de un dominio diferente, una estructura del grafo de distinta naturaleza y una diferente tarea de predicción. La Parte I se centra en el dominio del Reconocimiento de Actividades Humanas (HAR, por sus siglas en inglés). El HAR se realiza utilizando datos provenientes de sensores portables y dispositivos móviles inteligentes. En este contexto, la topología del grafo no es explícita, es decir, no forma parte de los datos de entrada. Por lo tanto, el grafo debe inferirse o construirse a partir de

supuestos adicionales sobre los datos subyacentes. El problema de HAR se plantea como una tarea de clasificación de grafos. En consecuencia, el modelo se entrena para aprender una única representación para el grafo completo. La Parte II se enfoca en el dominio de las redes de distribución de agua (WDNs, por sus siglas en inglés). En este dominio, la topología del grafo está explícitamente definida por el diseño de la red de distribución. El problema abordado en esta parte consiste en reconstruir la señal de presión para todos los nodos de la red a partir de un número reducido de sensores ubicados de manera dispersa. En este caso, la reconstrucción de la presión se plantea como una tarea de regresión a nivel de nodo. El modelo debe aprender representaciones para cada nodo con el fin de estimar los valores de presión faltantes. Los capítulos centrales de esta tesis están organizados de la siguiente manera.

En el Capítulo 2, presentamos los fundamentos teóricos de los temas y métodos discutidos en las Partes I y II de esta tesis.

En el Capítulo 3, describimos cómo los enfoques convencionales para el HAR pueden producir resultados ficticios, volviéndose poco prácticos o incluso inútiles en entornos reales. Esta parte de la investigación fue crucial para aplicar un método de evaluación imparcial al HAR y llevar a cabo una comparación justa entre los modelos de HAR estudiados en el Capítulo 4.

En el Capítulo 4, proponemos una arquitectura para aprender mejores representaciones a partir de datos de sensores portables con el fin de mejorar el desempeño de los modelos de HAR. Proponemos dos construcciones de grafos para capturar las dependencias globales y locales en los datos de sensores. Posteriormente, un enfoque de aprendizaje contrastivo maximiza la información mutua entre dichos grafos. Validamos empíricamente nuestro enfoque, demostrando que las representaciones aprendidas incrementan significativamente la precisión del modelo de clasificación.

En el Capítulo 5, combinamos el modelado basado en principios físicos y las GNNs para abordar el problema de estimación de presión

en redes de distribución de agua. Proponemos una estrategia de entrenamiento que se basa en la colocación aleatoria de sensores para incrementar la robustez del modelo frente a cambios inesperados en la ubicación de los sensores. Además, proponemos un protocolo de evaluación realista que considera patrones temporales reales e inyección de ruido para imitar las incertidumbres inherentes a escenarios del mundo real. Como resultado, se presenta GATRes, un nuevo modelo de última generación para la estimación de presión en WDNs.

En el Capítulo 6, analizamos cómo los métodos actuales de aprendizaje profundo para grafos, adaptados al dominio del agua, pueden dar paso a los Modelos Fundacionales de Grafos (GFMs, por sus siglas en inglés) para redes de distribución de agua. Discutimos cómo pasar de la concepción a la implementación real de este tipo de modelos. Abordamos los desafíos y proponemos estrategias para su realización, desde la generación de datos, pasando por el entrenamiento del modelo, hasta la adaptación del modelo para resolver tareas específicas de las WDNs.

Finalmente, en el Capítulo 7, presentamos las conclusiones y las futuras líneas de investigación que se desprenden de este estudio.

Contents

Chapter 1

Introduction

In the early years of deep learning, simple cognitive tasks for humans, such as distinguishing between cats and dogs images or recognizing handwritten digits, were among the biggest challenges for Artificial Intelligence (AI) research. Nowadays, these and several more complex tasks can be easily solved by computers with high level of accuracy. These improvements in the field of AI were achieved, among other factors, due to a better *representation* of the input data. The way data are represented plays an important role in the level of difficulty in solving a particular task, either by humans or computer-based agents [87]. For example, just a simple mathematical operation such as $11 + 9$ becomes less intuitive for people, unfamiliar with the binary notation, if it is represented in the binary numerical system as $1011 + 1001$.

Representation learning allows AI algorithms to extract and organize the discriminative information from the raw input data. Those learned representations disentangle the explanatory factors of the observed input, making it easier for AI algorithms to solve a particular task, e.g., classify images [15]. The advent of deep learning implied major advances in representation learning. Deep learning models are composed of multiple layers interleaved with non-linear transformations. Each layer extracts representations at different levels of abstraction. Taking the cats and dogs image classification example, the first layer of a deep learning model extracts edges from the raw image pixel data. Those representations allow the next layer to extract shapes or motifs despite small variations in the edges' orientation. The next layer uses those motifs to identify object parts, e.g., nose or ears in our example. The subsequent layers combine those parts to identify the objects in the image [126]. At each layer, different low-level concept representations are learned, which are composed to formulate higher-level concepts "deeper" in the layers' hierarchy. The composition of learned representations resembles the way we, humans,

use learned concepts, identify associations, and determine relationships between them to understand the world around us. The structure that naturally allows us to represent associations and relationships between concepts or things is a *graph*.

The term graph may create a different mental image on every reader. Everyone may have a notion of a graph. Nonetheless, perhaps like me before I started this research, readers may not realize that graphs are everywhere. They manifest in nature in several ways. For example, molecules are graphs of atoms interconnected by the chemical bonds between them. Social networks are graphs of people connected, e.g., by a friend relationship. Transportation networks are graphs of bus/train stations connected by routes. Financial networks are graphs of monetary transactions between people or organizations. The internet is the largest graph of connected computer networks. In our brain, a connectome is a graph of connected neurons through synapses. With the right abstraction, most phenomena—if not all—can be modeled as a graph.

In simple terms, graphs are a mathematical construct comprised of a set of entities, or nodes, connected pairwise by edges, which denote some type of relationship between them. The simplicity and suitability of graphs for modeling complex relationships between entities of interest have attracted the attention of Academia and Industry in the search for solutions to practical real-world problems.

1.1 Smart environments and deep learning on graphs.

Smart environments are systems capable of acquiring and applying knowledge about their inhabitants to meet certain comfort and efficiency related goals [49]. At the origins of smart environments research, its definition was largely limited to smart homes and its applications related to smart assisted living. Today, the notion of smart environments has expanded to other domains and applications due to the increasing availability of sensing devices, the increasing power of modern computing hardware, and the digital transformation within business, and industry. The human-centric approach, of the original conception, is now

extended to business or industrial settings, e.g., smart energy grids [125], or smart water distribution [113]. The common characteristics between these *smart* systems is that they acquire data by sensing the physical environment and use them to understand the context of the current situation [56]. While smart environments leverage interconnected devices to enhance automation and efficiency, deep learning plays an important role in making these systems more intelligent and adaptive.

Deep Learning has contributed to significant advances in a wide variety of domains, including computer vision [18, 119, 171], speech recognition [9, 172], drug discovery [145, 194], genomics [2, 157], and more recently, an increasing number of breakthroughs in natural language processing with the advent of Large Language Models [30, 79, 207]. Although state-of-the-art deep learning models undoubtedly excel at learning useful representations to solve complex prediction tasks, most of them are designed to operate on data in the Euclidean space as the de facto standard, and sampled on fixed structures such as grids (e.g., images) or sequences (e.g., text, audio). However, as presented before, there are several phenomena whose data reside in non-Euclidean spaces, e.g., *graphs* [29].

The fixed structure for which deep learning models were designed is a constraint for operating on graph-structured data. Graphs are of arbitrary size and structure. When dealing with graphs, there is no explicit notion of locality. In graphs, we cannot define boundaries determined by the top, bottom, left, and right borders as in images. Hence, it is not straightforward to slide a kernel along a graph as we do, with convolutional neural networks (CNNs), over a grid or a sequence. The researchers then found the need to generalize deep learning architectures and adapt them to operate on non-Euclidean spaces [28]. *Graph Neural Networks* (GNNs) are a prominent example of those adaptations, specialized for graph-structure data. This thesis explores the use of GNN architectures to learn better representations of data and increase the performance of the models in smart environments.

1.2 Problem statement and research questions

The problem addressed in this work is to *learn better representations of the data acquired from sensors embedded in a physical environment, whose underlying structure is modeled as a graph*. In the context of this research, a representation is *better* if it is useful to increase the performance of a prediction model on a specific task. Better data representations can be learned if the symmetry and invariance in the underlying data are preserved and introduced as inductive biases in deep learning models [28]. First, by modeling them as a graph, inductive biases are introduced in the form of relationships between the nodes in the graph. Using GNN architectures enforces that such inductive biases are captured because the learning process considers the features of nodes and edges, and the graph topology to learn the representations. In this thesis, smart environments are considered from the broader perspective, and not restricted to smart home applications. Hence, the problem of graph representation learning is addressed within two separate domains: Human Activity Recognition (HAR) and Water Distribution Networks (WDNs). Each domain has its own challenges and requires different analyses from different perspectives.

The field of HAR has been studied for several decades [38, 51, 160], becoming a well-founded discipline with solid methodologies and established procedures to implement HAR pipelines [31, 122]. Those methods describe each stage in the HAR pipeline, the techniques and algorithms that can be used at each stage, and how to evaluate the overall performance of a HAR system. The input data are transformed or rearranged to be passed to the next component in the computational pipeline. Although the entire process is thoroughly described in the literature, practitioners and researchers should carefully analyze how the operations applied to the data at each stage affect the subsequent stages. HAR computational pipelines combine techniques from signal processing, pattern recognition, and machine learning [31]. These techniques require certain assumptions about the data to be made. Therefore, we argue that the implementation of HAR pipelines without analyzing those assumptions and how the data are affected at each stage can invalidate the results. For example, what

considerations have to be taken into account when splitting the data into training, validation, and test sets if a window mechanism is used to segment the sensor data? The way data segmentation and dataset splits are applied may introduce "data leakage", i.e., data in the test set that are also included in the train set and therefore seen by the model during training. This motivates the first research question.

RQ-1. *How common are methodological flaws in the application of (Deep) Machine Learning models for Human Activity Recognition, and how do they affect the validity of claimed or reported results?*

A common approach to HAR is the use of the data produced by smartphones and wearable sensors attached to subjects while performing daily activities within a smart environment. These data come in the form of time series of sensor readings sampled at some regular interval. In this setting, common approaches exploit the temporal dependencies in the data using Long Short-Term Memory Networks or other type of Recurrent Neural Network models [42, 160]. In addition, the capability of Deep Learning models to automatically extract features from raw input data has popularized the use of CNNs in this domain [222, 244]. We posit that collected signals from sensors have additional dependencies that go beyond the temporal dimension, and that such dependencies can be modeled as a graph. Nonetheless, the lack of an underlying graph topology poses an additional challenge. The graph needs to be inferred or constructed on the basis of additional assumptions of the underlying data. Since data is not given naturally as a graph, the inferred graph may introduce spurious correlations. Hence, another challenge in this context is how to mitigate the effect of those false correlations in the data. The learned representations should not include, or at least minimize, those non-existent relationships that were introduced in the input graph. This motivates our second research question.

RQ-2. *Can a graph representation of sensor data be used to improve the performance of prediction models on problems where there is no existing underlying graph topology? How can these models be trained to learn better data representations for Human Activity Recognition?*

Turning now to a domain where the data is naturally in the form of a graph, we leverage graph representation learning in Water Distribution Networks. Although we have a pre-defined graph topology given by the WDN layout, the first challenge in this domain is how to deal with partially observable data due to the reduced number of sensors installed in the network. The problem we address within this domain is state estimation. A proper estimation of the state of a WDN allows practitioners in the field to monitor, control, and optimize network operation. The models need to learn useful representations that allow them to estimate the pressure values at all nodes in the network, from just a few known pressures measured by sparsely located sensors. The learned representations have to be useful for making estimations in different WDNs. Hence, the models need to be able to generalize to unseen WDNs topologies. Sensors can be added or removed due to planned maintenance, sensor malfunctions, or other unexpected circumstances. Hence, the limited number of sensors requires the models remain reliable despite dynamic sensor availability, and they adapt to changes in sensor layouts over time. In addition, the models need to be robust to the uncertainties inherent to real-world scenarios, e.g., noisy observations, errors in data transmission, and changes in user behavior related to water consumption. These requirements for state estimation models in WDNs motivates the third research question.

RQ-3. *Can graph-based neural network models be employed for state estimation in Water Distribution Networks, and what capabilities should they possess in this case? How can we enforce such capabilities from design time, and how can we incorporate them in the training strategies of graph-based state estimation models?*

A clear example of the model generalization capabilities has been observed in commercial products like ChatGPT [30], Gemini [79], Llama [67], or DeepSeek [135]. These so-called Foundation Models were trained on vast amounts of data on generic language tasks, but they can be used to solve mathematical problems, to summarize text, to write software, among other specific tasks. While there are several state-of-the-art

language and multi-modal Foundation Models, less attention has been paid to Foundation Models for graph-structured data, let alone its applicability to the WDNs domain. The first attempts of Graph Foundation Models (GFMs) are domain specific, e.g., ULTRA [76] focuses exclusively on Knowledge Graphs, and MolGPS [195] is a GFM for molecular graphs. This motivates the fourth research question.

RQ-4. *How can Graph Foundation Models be designed and implemented to be tailored to the WDNs domain? How the scarcity of data in this domain can be solved in order to train such data-hungry models? How can these models learn generalizable and transferable representations useful for solving specific downstream tasks?*

These are the questions addressed in this dissertation. The first question, **RQ-1**, is answered in Chapter 3, where we describe a methodological issue that leads to accuracy overestimation of deep learning models. The question **RQ-2** is answered in Chapter 4, which proposes a graph contrastive learning approach to maximize mutual information between two different graph construction methods. The answer to question **RQ-3** is given in Chapter 5 where a graph-based pressure estimation model is presented, describing how the proposed model meets the requirements of generalizability, adaptability and robustness. The final research question, **RQ-4**, is answered in Chapter 6, which describes the first attempt of Graph Foundation Models tailored to the WDNs domain, and shows the feasibility of this type of models to learn generalizable and transferable representations for WDNs.

1.3 Thesis Scope and Organization

The main goal of this thesis is to design deep learning architectures that enable inference models in smart environments to learn useful representations from sensor data modeled as a graph. The thesis is organized in two parts, each focusing on a different problem domain, a different nature of the graph topology, and a different type of prediction task.

Part I focuses on the domain of Human Activity Recognition. HAR is

performed based on wearable and mobile sensors. In this setting, the graph topology is not explicit, i.e., it is not part of the input data. Hence, the graph needs to be inferred or constructed on the basis of additional assumptions on the underlying data. The HAR problem is framed as a graph classification task. Thus, the model is trained to learn a single representation for entire graphs.

Part II focuses on the water distribution networks. In contrast to the HAR domain, in this domain the graph topology is explicitly given by the layout of the WDN. The problem addressed in this part is to reconstruct the water pressure, for all nodes in the network, from a few sparsely located sensors. In this case, the pressure reconstruction is framed as a node-level regression task. The model has to learn the representations for each node in order to estimate the missing pressure values. The core chapters of this thesis are organized as follows.

In **Chapter 2**, we present the theoretical background for the topics and methods discussed in Parts I and II of this thesis. First, we present the background in the field of graph representation learning, including the definitions and principles of Graph Neural Networks (GNNs), the central deep learning architecture used in this research. Then, we describe HAR and WDNs in the context of Smart Environments, and present the general definitions within those domains.

In **Chapter 3**, we show that methodological problems in the implementation of HAR models can produce misleading results due to performance overestimation. Several experiments with different types of datasets and different types of classification models allow us to expose the problem and to show it persists independently of the method or dataset used. The results show that such highly overestimated models would become impractical or even useless in real settings. This part of the research was crucial to designing an unbiased evaluation method for HAR and carrying a fair comparison between HAR models studied in Chapter 4.

In **Chapter 4**, we propose an architecture for learning better representations from wearable sensor data to improve the performance of

HAR models. We propose two graph constructions to capture the global and local dependencies from intra- and inter-sensor data channels. Then, a contrastive learning approach maximizes mutual information between these graphs. The learned representations enabled a significant improvement of the HAR model performance.

In **Chapter 5**, we combine physics-based modeling and GNNs to address the pressure estimation problem in WDNs. We propose a training strategy that relies on random sensor placement to increase the robustness of the model to unexpected sensor location changes. In addition, we propose a realistic evaluation protocol that considers real temporal patterns and noise injection to mimic the uncertainties inherent to real-world scenarios. As a result, GATRes, a new state-of-the-art model for pressure estimation is available. Our model surpasses the performance of previous studies on several WDNs benchmarks, showing a reduction in absolute error of $\approx 40\%$ on average.

In **Chapter 6**, we discuss how current deep learning methods for graphs, tailored to the water domain, can be the enablers of Graph Foundation Models for WDNs. We discuss how to go from conception to actual implementation of this type of models. We discuss the challenges and propose the strategies to its realization, from data generation, to model training, to model adaptation to solve WDNs specific tasks. The DiTEC-WDN dataset is one of the main contributions towards the realization of GFM for WDNs domain, providing openly accessible data to train data-hungry deep learning models.

Finally, in **Chapter 7**, we present the concluding remarks and the future directions of this research.

1.4 Publications

This dissertation is based on several scientific publications, which has been published in or, at the time of writing, are under consideration for publication in journals or conferences. Those publications were done in collaboration with other colleagues, in particular Victoria Degeler, Huy

Truong, Alexander Lazovik, Mostafa Hadadian, Erkan Karabulut, Robert Riesebos, and Hester van het Loo.

The following is a list of the papers on which this thesis is based, and the chapters they are mostly relevant for.

PART I: Deep Learning on Graphs for Human Activity Recognition

- **Chapter 3**: Accuracy overestimation in Human Activity Recognition.

  This chapter is based on the paper:

  A. Tello, V. Degeler, and A. Lazovik. *Too good to be true: accuracy overestimation in (re) current practices for human activity recognition*. In IEEE Int. Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), pages 511–517. IEEE, 2024 [202].

  AT and VD proposed and refined the idea. AT developed the methods and experiments, executed the project and did the main writing. VD reviewed and edited the manuscript, and supervised the project. AL reviewed the manuscript, and supervised the project.

- **Chapter 4**: Graph contrastive learning for Human Activity Recognition.

  This chapter is based on the paper:

  A. Tello and V. Degeler. *Contrasting global and local representations for human activity recognition using graph neural networks*. In Proceedings of the 40th ACM/ SIGAPP Symposium on Applied Computing (SAC '25), 2025 [201].

  AT proposed and refined the idea, developed the methods and experiments, did the main writing. VD helped refining the idea, reviewed the manuscript, and supervised the project.

PART II: Deep Learning on Graphs for Water Distribution Networks

- **Chapter 5**: Graph Neural Networks for Pressure Estimation in Water Distribution Systems

This chapter is based on the papers:

- H. Truong*, **A. Tello***, A. Lazovik, and V. Degeler. *Graph neural networks for pressure estimation in water distribution systems*. Water Resources Research, 60(7): e2023WR036741, 2024 [209]. *Both authors are considered first authors.

  AT contributed with the main writing of the (sub)sections: Introduction, Problem Statement, Partially-Observable Data and Realistic Model Evaluation, Experiment Settings, Generalization, Baseline Comparison on Benchmark WDNs, Ablation Study, and Conclusions. AT ran the experiments to compare the distribution of data used in the paper, ran the experiments to compare different masking approaches and select the one included in the implementation of the proposed architecture, and ran the experiments to compare GATRes against the baselines on benchmark datasets. AT developed the GATRes Block architecture. HT contributed with the main writing of the (sub)sections: Criteria for Model Assessment, Related Work, Water Network as Graph, Data Set Creation, Model Architecture, Model Training, Baseline Comparison on Oosterbeek WDN, The Effect of Masking Ratios. HT proposed and implemented the dataset creation algorithm, ran the experiments to compare GATRes against the baselines on Oosterbeek WDN, and ran the experiments to check the effect of masking ratios. HT and AT proposed the idea, contributed to the writing of Discussion section. AL and VD helped refining the idea, reviewed and edited the manuscript, and supervised the project.

- V. Degeler, M. Hadadian, E. Karabulut, A. Lazovik, H. van het Loo, **A. Tello**, and H. Truong. *Ditec: Digital twin for evolutionary changes in water distribution networks*. In International Symposium on Leveraging Applications of Formal Methods, pages 62–82. Springer, 2024 [57].

  VD proposed the idea, did the main writing, and supervised the project. AT contributed to the writing of State Estimation in Section 4.2, and refining the proposed architecture. MH, EK, and HT contributed to the writing and refining the proposed

architecture. AL reviewed and edited the manuscript, and supervised the project, HL contributed to the writing.

- **Chapter 6**: Towards Graph Foundation Models for Water Distribution Networks.

  This chapter is based on the papers:

  - **A. Tello**, H. van het Loo, A. Lazovik and V. Degeler. *Towards Graph Foundation Models for Water Distribution Networks*. In the 21st International Computing & Control in the Water Industry Conference (CCWI2025) [204].

    AT proposed and refined the idea, developed the methods and experiments, did the main writing. HL contributed with writing. AL and VD helped refining the idea, reviewed and edited the manuscript, and supervised the project.

  - **A. Tello***, H. Truong*, A. Lazovik, and V. Degeler. *Large-scale multipurpose benchmark datasets for assessing data-driven deep learning approaches for water distribution networks*. Engineering Proceedings, 69(1):50, 2024 [203]. *Both authors are considered first authors.

    AT and HT proposed the idea, did the main writing, developed the methods and experiments. AL and VD helped refining the idea, reviewed the manuscript, and supervised the project.

  - H. Truong*, **A. Tello***, A. Lazovik, and V. Degeler. *DiTEC-WDN: A Large-Scale Dataset of Hydraulic Scenarios across Multiple Water Distribution Networks*. Scientific Data, 12(1):1733, 2025. [210]. *Both authors are considered first authors.

    AT developed and implemented the Automatic Demand Generation algorithm. HT developed and implemented the data generation and the Particle Swarm Optimization algorithms. AT and HT proposed the idea, did the main writing, and ran the simulations for data generation. AL and VD helped refining the idea, reviewed the manuscript, and supervised the project.

The following publications are relevant to the topics presented in this thesis, but are not directly related to any chapter.

- **A. Tello** and V. Degeler. *Digital twins: an enabler for digital transformation*. In The Digital Transformation handbook. 2021 [200].

  AT proposed and refined the idea, and did the main writing. VD helped refining the idea, contributed to the main writing, reviewed and edited the manuscript.

- R. Riesebos, V. Degeler, and **A. Tello**. *Smartphone-based real-time indoor positioning using ble beacons*. In 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), pages 1281–1288. IEEE, 2022 [176].

  RR refined the idea, executed the project, and did the main writing. VD proposed and helped refining the idea, reviewed the manuscript, and supervised the project. AT helped refining the idea, reviewed the manuscript, and co-supervised the project.

Chapter 2
Background

In this chapter, we present the theoretical background of the topics that are the fundamental part of this research. First, we introduce Graph Representation Learning in Section 2.1, including the preliminaries on graphs, the different graph-related predictive tasks, and a description of GNNs, which are the core deep learning approach used in this research. Next, we describe Smart Environments in the context of this research in Section 2.2. We present the application domains of the graph representation learning models proposed in this thesis, i.e., HAR in Section 2.2.1 and WDNs in Section 2.2.2.

The formalizations presented in this chapter are generally applicable to the domain addressed in each Part of this dissertation, i.e., either HAR or WDNs. In the following chapters, those definitions may be adapted according to the specific requirements of the problem addressed in each chapter. Some definitions presented in this chapter may be repeated in subsequent chapters to make each chapter self-contained.

2.1 Graph Representation Learning

Graph Representation Learning is a subfield of deep learning focused on extracting useful representations of data whose underlying structure can be more naturally modeled as a graph. The learned representations have to encode the properties inherent to the graph structure and the features of nodes and edges in the graph. These representations, called *embeddings*, are mappings of nodes, or of the entire graph, to a point in a low-dimensional vector space $\mathbb{R}^d$. The goal is that the learned embeddings reflect the structure of the original graph such that connected nodes in the graph are also closer in the embedding space. Then, the embeddings can be used for specific downstream machine learning predictive tasks.

The main difference between graph representation learning and previous approaches to encoding graph-structured data lies in how the mapping is performed. In previous approaches, before the deep learning era, embeddings were computed using hand-engineered statistics extracted from the graph. Such processes were labor-intensive and relied heavily on domain expert knowledge. In contrast, in graph representation learning the mapping is an automatic process that minimizes a loss function $\mathcal{L}$ using stochastic optimization techniques such as gradient descent [117, 179].

2.1.1 Preliminaries on Graphs

Before describing specific graph representation learning approaches, we formally introduce the concept of a *graph*. The definition and notation presented in this section, along with minor adaptations when necessary, will be used throughout the rest of this thesis.

A graph is defined as an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, or vertices, and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the set of edges. The edges represent the relationships between each pair of nodes in the graph. An edge between node $i \in \mathcal{V}$ and node $j \in \mathcal{V}$ is represented as $e_{i,j} \in \mathcal{E}$. In this formalization, we assume that graphs are simple, undirected, and connected. A simple graph is a graph without loops and without multiple edges between a pair of nodes. In undirected graphs, the connections between nodes i and j go in both directions, i.e., $e_{i,j} \in \mathcal{E} \leftrightarrow e_{j,i} \in \mathcal{E}$. A graph is connected if there exists at least one path between any pair of nodes $i, j \in \mathcal{V}, \forall i, j (i \neq j)$

A common approach to mathematically representing a graph is to use its adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$. The indices of rows and columns of $\mathbf{A}$ represent the nodes in the graph. Then, the entry $\mathbf{a_{i,j}} = \mathbf{1}$ if $e_{i,j} \in \mathcal{E}$, or $\mathbf{a_{i,j}} = \mathbf{0}$ otherwise. This binary adjacency matrix $\mathbf{A} = a_{i,j} \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ only represents the connectivity of the graph. To express the strength of the connections between nodes in the graph, the entries of $\mathbf{A}$ can take arbitrary values, i.e., $\mathbf{A} = a_{i,j} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$.

In the context of graph representation learning, we consider attributed graphs, i.e., graphs where nodes, edges, or even the graph itself have associated information or *features*. The features of the nodes

in the graph are represented as a real-valued matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_{node}}$, where d_{node} is the dimension of the feature vector of the node $x_i, \forall i \in \mathcal{V}$. Likewise, the edge features are represented as a matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_{edge}}$.

The attributed graph notation is used in the problems addressed in Part I and Part II of this thesis. In the HAR domain presented in Chapter 4, the node feature matrix is given by the sensor data readings, and the graph has a label that represents the human activity. Similarly, in Chapter 5, each node is associated with a pressure value. The edges, which represent the pipes in the WDN, have length and diameter as features certain cases.

2.1.2 Graph Learning Tasks

Graph representation learning enables models to create embeddings for nodes, edges, and graphs, which can be used for specific downstream predictive tasks. Depending on which element of the graph the final task concerns, predictive tasks can be categorized as node-level, edge-level, or graph-level tasks [95, 258].

Node-level. This type of task focuses on predicting the properties of the nodes. It may involve predicting a real-valued property of each node (node regression) or assigning each node to a specific category (node classification). Node classification being the most common task in the context of graph machine learning. In both cases, the predictions are based on the graph information, i.e., node features and topology. The problem of node classification is regarded as inductive if the nodes to be classified belong to a previously unobserved graph during training. In contrast, in the transductive setting, the model is trained on a subset of the nodes and tested on a different subset of the same graph. Application examples of this type include anomaly detection [197] or document classification [255].

In this thesis, we address the node-level regression task in Chapter 5. The pressure estimation problem presented in that chapter is framed in both transductive and inductive settings.

Edge-level. At this level, the most common task is predicting the presence of an edge, usually referred to as link prediction. *link prediction*. Thus, given the set of nodes $\mathcal{V}$ and a subset of edges $\mathcal{E}_{train} \subset \mathcal{E}$, link prediction uses the available information to predict the missing edges $\mathcal{E} \backslash \mathcal{E}_{train}$. Applications of this kind include drug-target interactions [39], knowledge graph completion [54], or recommendation systems [243]. Edge-level tasks are not addressed in the context of this thesis. Its description is included for completeness with respect to graph machine learning tasks.

Graph-level. The goal of this type of task is to make predictions about the entire graph instead of individual nodes or edges. Similar to the node level, at the graph level we can perform classification or regression tasks. Thus, given a set of attributed graphs, the goal is to assign them to a specific category or to predict a property of each graph based on its features and structure. Application examples include the prediction of antibiotic activity in molecules represented as graphs [229] and skeleton-based action recognition [186].

In this thesis, we address this type of task in Chapter 4. We frame HAR as a graph classification problem in which human activity graphs are inferred from wearable sensor data.

2.1.3 Graph Neural Networks

Graph Neural Networks are a specialized type of artificial neural network (ANN) model designed for structured data presented in the form of a graph. The idea originated from the need to enable ANN models to operate not only on feature vectors but also on complex data structures (e.g., graphs) whose elements contain associated information [73, 192].

Gori *et al.* formally introduced the concept and coined the term Graph Neural Network for the type of model that learns a function τ that maps a graph $\mathcal{G}$ and a node n to a low-dimensional vector space $\mathbb{R}^d$ [88]. In the initial formulation, the learned representation for node n was conditioned only on its initial features, x_i, and the features of its neighbors, as defined by the graph topology. Later, Scarselli *et al.* [184] extended and generalized the formulation, including the feature vectors

of the edges in the neighborhood of node n, $x_{(i,j)} \forall j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the set of nodes connected to node i.

The pioneering GNN formulation led to a variety of models and architectures, e.g., Chebyshev graph convolution [55], Graph Convolutional Network (GCN) [118], Mixture Model Networks (MoNet) [151], GraphSAGE [96], Graph Attention Networks (GAT) [215], and Graph Isomorphism Network (GIN) [236], among others. The way GNN models operate is known as the message-passing framework [81], since GNNs learn node representations by exchanging messages (node states) between local neighborhoods in the graph. A GNN model is composed of K stacked GNN layers, and the representation or embedding for node i is computed by an iterative process in K iterations. Thus, the most general formulation of the learning process in state-of-the-art GNNs is as follows:

$$h_i^{(k)} = \Phi\left(h_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}_i} \Psi\left(h_j^{(k-1)}\right)\right) \tag{2.1}$$

where $k = \{1, 2, ..., K\}$ indicates the current iteration. Ψ is a learnable differentiable function that computes the messages sent to node i from the previous state of its neighboring nodes $h_j^{(k-1)} : \forall j \in \mathcal{N}_i$. $\bigoplus$ is a permutation-invariant function that aggregates the messages from the neighbors of i. Φ is a learnable differentiable function that updates the state of node i, $h_i^{(k)}$, based on its previous representation $h_i^{(k-1)}$ and the aggregated messages from its neighbors. Note that in the first iteration, the node states are the original features of the nodes, i.e., $h_i^{(0)} = x_i, \forall i \in \mathcal{V}$. The outputs produced by the final GNN layer K are the representations or embeddings of each node, i.e., $h_i = h_i^{(K)}, \forall i \in \mathcal{V}$.

The learned representations of the nodes can be used to solve different predictive tasks at the node level. However, for graph-level predictive tasks, we need to produce a single representation of the graph. This is achieved using an additional permutation-invariant aggregation function applied to the final embeddings of all nodes. Thus, the graph representation can be defined as follows:

$$h_{\mathcal{G}} = \bigoplus_{i \in \mathcal{V}} h_i^{(K)} \tag{2.2}$$

It is important to note that the node-level and graph-level aggregation

functions, $\bigoplus$, must be permutation-invariant. Nodes in a graph are presented in an arbitrary order. Thus, the results of the aggregation should not be affected by the ordering of the input features. The common aggregation functions used in practice are sum, mean, or max [236]. The aggregation function plays a role similar to the pooling operation in CNNs, which aggregates local information. While the pooling in CNNs is invariant to rotations and translations, the aggregation functions in GNNs are invariant to graph isomorphisms.

Another important consideration is the implementation of Φ and Ψ. The most common approach, and the one used in this thesis, is a learnable affine transformation with activation functions. Thus, $\Psi(\cdot) = W_1(\cdot) + b$, and $\Phi(\cdot,\cdot) = \sigma(W_1(\cdot) + W_2(\cdot) + b)$, where W_1, W_2, and b are learnable parameters, and σ is an activation function such as the rectified linear unit (ReLU) [28].

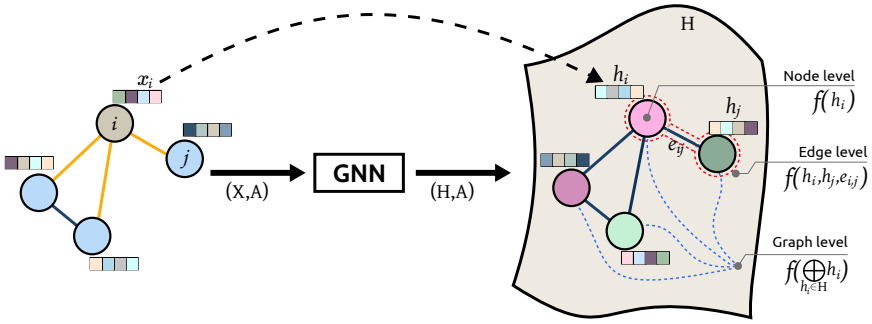


Figure 2.1: Graph neural networks for learning data representations

The learned representations produced by the iterative process described above can be used by various predictive models to solve domain-specific downstream tasks. These tasks can be performed at any level as previously described in Section 2.1.2. Figure 2.1 shows an overview of the graph representation learning process using a GNN model. The original input features of the nodes and edges are passed through the GNN and mapped into a latent space $\mathbf{H}$. The embeddings or learned representations are used as input to a function $f(\cdot)$ to solve a node-, edge-, or graph-level task.

2.2 Smart environments

Smart environments serve as the application domain of the graph-based representation learning models proposed in this work. We consider a smart environment to be a physical space equipped with sensing devices that measure various physical properties. As described earlier, the concept of a smart environment has shifted from a purely human-centered approach towards a broader business or industrial setting, depending on the goals to be achieved. In this work, we address graph representation learning from both perspectives. The human-centered approach is explored in Part I, which focuses on Human Activity Recognition, whereas the industrial setting is examined in Part II, which concerns Water Distribution Networks.

2.2.1 Human Activity Recognition

Human activity recognition is a fundamental component of a smart environment, enabling adaptive and context-aware responses. Hence, HAR has been extensively used as part of smart homes [19, 48], smart hospitals [34, 182], smart cities [109], smart industry [257], and other similar settings. The data used for HAR come from different modalities, e.g., camera sensors [47, 185], ambient sensors [27, 150], and inertial sensors embedded in wearable devices or smartphones [89, 251].

In the context of graph representation learning, GNNs have been used mainly for HAR tasks based on skeleton graphs extracted from video frames [40, 45, 165]. Less attention has been paid to GNN-based models for learning representations of inertial sensor data from smartphones and wearable devices. In Part I of this thesis, we leverage GNNs to learn sensor embeddings from Inertial Measurement Unit (IMU) Sensors. This type of sensor is commonly used in HAR because such sensors are seamlessly embedded in everyday wearable and mobile devices such as smartphones, smart watches, smart bands, and even clothing [38]. These sensors include accelerometers, gyroscopes, and magnetometers.

Research on HAR based on inertial sensors has established methodologies for designing and evaluating HAR systems. The commonly used HAR pipeline includes data collection and pre-processing, data segmentation,

feature extraction, model training and classification, and performance evaluation [31]. Figure 2.2 show an overview of the HAR pipeline.

The collected IMU sensor data are time series given by the sensor readings sampled at some regular time interval Δt. Some sensors provide multiple measurements, considered the channels of the sensor signal. For example, triaxial accelerometers provide values for the x, y, and z directions. Thus, the time series for IMU sensors is defined as follows.

$$\mathbf{S}_i \in \mathbb{R}^{T \times Ch}, \text{ for } i = 1, 2, ..., K \tag{2.3}$$

where T is the total number of samples, Ch is the number of channels in sensor $\mathbf{S}_i$, and k is the total number of sensors.

Then, the raw data collected from sensors is pre-processed to reduce noise or remove artifacts, typically by applying low-pass and high-pass band filters. Other common operations in this phase are data normalization, data imputation, resampling, and synchronization [31]. The raw input data signals are transformed into the pre-processed signals $\mathbf{S}'_i$. The next stage is data segmentation, which partitions the data into smaller segments. These segments are called windows, and in the context of HAR based on IMU sensor data, they represent the signal samples between time steps t_i and t_j. At this stage, the pre-processed data is transformed into a set of windows denoted as follows.

$$\mathrm{W} = \{w_1, w_2, ..., w_K\}, \; w_k = \{t_i, t_{i+\Delta t}, t_{i+2\Delta t}, ..., t_j\} \tag{2.4}$$

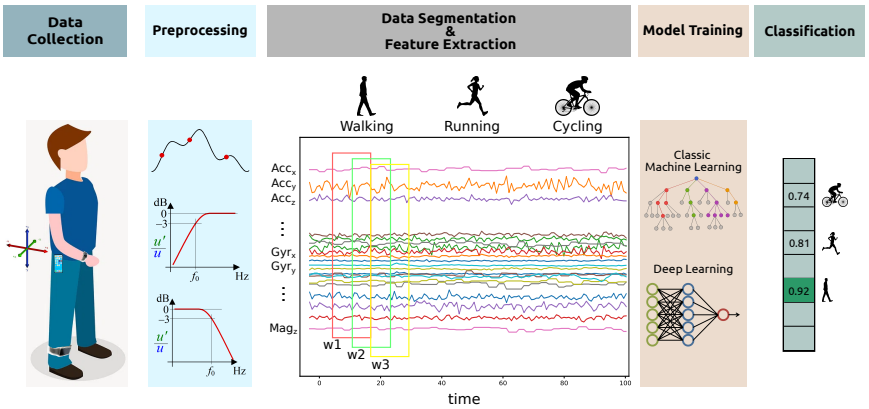


Figure 2.2: Overview of the Human Activity Recognition pipeline.

where K is the total number of windows, t_i is the start time of window w_k, t_j is the end time, for $t_i + m \cdot \Delta t \leqslant t_j$

At this stage, the data windows serve as the new samples used to extract features for activity recognition. HAR is framed as a supervised classification problem. Then, given a set of N labeled sample pairs $\{(x_i, y_i)\}_{i=1}^{N}$, where $x_i = (x_1, x_2, ..., x_d) \forall x_i \in X$ is the feature vector of sample i with dimension d, and $y_i \in Y = \{y_1, y_2, ..., y_c\}$, is the activity label, corresponding to sample i, the goal of HAR is to learn a mapping from X to Y. Thus, the HAR model is trained to produce a function that maps the input feature vector space to a value in the activity label space $\mathcal{F} : X \rightarrow Y$. This mapping function f is learned by a model with parameters θ and produces a score vector $P_i = (p_1, p_2, ..., p_c)$. The class probabilities are defined as follows.

$$p_i(y|x_i, \theta) = \mathcal{F}(x_i, \theta), \forall y \in Y \tag{2.5}$$

Then, the predicted activity label $\hat{y}$ for the input features x_i is the label corresponding to the maximum score in P_i

$$\hat{y}_i = \text{argmax}\ p_i(y|x_i, \theta) \tag{2.6}$$

In the context of this thesis, the function $\mathcal{F}$ in Equation 2.5 is implemented using end-to-end GNN-based models, which extract the features used for activity recognition, together with a final scoring function implemented as an affine transformation of the form $\theta x_i + b$. Chapter 4 describes how the pre-processed sensor data signals are transformed into windows and how these windows are transformed into graphs for further feature extraction and activity classification. It is known that the performance of HAR models depends heavily on the quality of the extracted features [107,249] and the limitations in domain knowledge [61]. Under this premise, representation learning plays a critical role within the HAR pipeline.

2.2.2 Water Distribution Networks

Water Distribution Networks are considered critical infrastructures, as they provide clean and safe water to humans, which is one of the Sustainable Development Goals proposed by the United Nations. Water providers

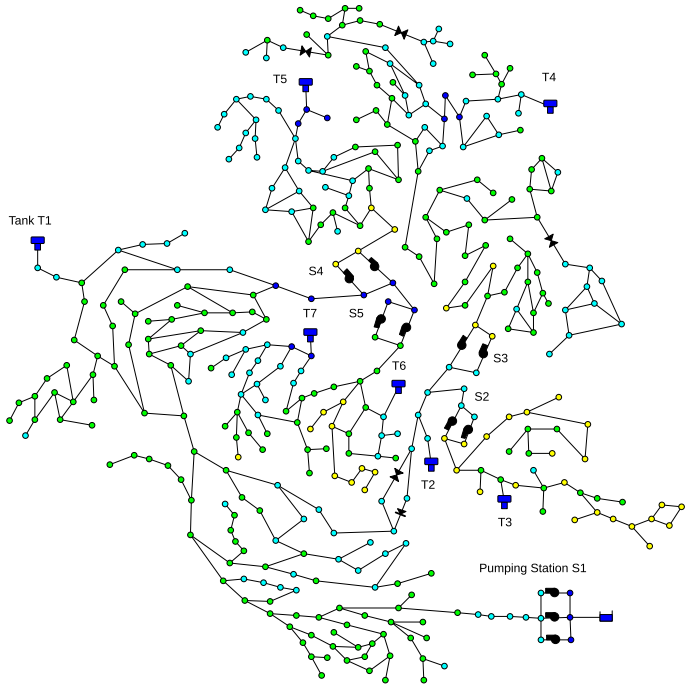


Figure 2.3: C-Town Water Distribution Network [161].

face critical challenges during the design, planning, and management phases of a WDN to achieve this goal. Adaptability and robustness to an ever changing environment constitute the most critical challenge. Climate, uncertain consumer behavior, aging infrastructure, natural disasters, failures, cyber- and physical-attacks can all lead to drastic changes in the conditions under which WDNs must continue operating effectively. Monitoring WDN operations plays an important role in ensuring the water supply. The state of the network must be known at any given time to prevent unwanted situations, e.g., pipe leaks.

Hydraulic modeling has been the most straightforward approach for practitioners to simulate WDN dynamics and aid design, planning, and management. While pure physics-based hydraulic modeling is still being commonly used in the water domain, water engineering research and practice are experiencing a shift towards hybrid data-driven approaches. Such approaches combine the power of physics and mathematical simulation tools with data-driven deep learning models to solve water engi-

neering problems.

WDNs are systems in which water is transported from sources to consumers at a prescribed pressure within a network of connected hydraulic components (e.g., pumps, valves, junctions) through pipes [70]. A WDN can be described as a graph in which junctions, tanks, and reservoirs represent the nodes, and pipes, pumps, and control valves represent the edges. Figure 2.3 shows the graph topology and the hydraulic components of the C-Town WDN [161].

In this thesis, we focus on graph representation learning of WDN data to aid monitoring tasks and improve day-to-day network operations. The WDN data are naturally represented as a graph. The graphs are attributed, i.e., the nodes and edges have associated properties, as described in Section 2.1.1. Then, GNN-based models use those properties to compute node embeddings during the representation learning process. Part II of this thesis addresses specific problems in this domain, and their definitions and formulations are described in detail in their corresponding chapters.

Part I

Deep Learning on Graphs for Human Activity Recognition

Chapter 3
Accuracy overestimation in Human Activity Recognition

Today, there are standard and well-established procedures within the Human Activity Recognition (HAR) pipeline. However, some of these conventional approaches lead to accuracy overestimation. In particular, sliding windows for data segmentation followed by standard random k-fold cross-validation produce biased results. An analysis of previous literature and present-day studies shows, surprisingly, that these are common approaches in state-of-the-art studies on HAR. It is important to raise awareness in the scientific community about this problem, whose negative effects are being overlooked. Otherwise, publications of biased results make papers that report lower accuracies, using correct unbiased evaluation methods, harder to publish. Several experiments with different types of datasets and different types of classification models allow us to exhibit the problem and show that it persists independently of the method or dataset.

3.1 Introduction

Human Activity Recognition is an ongoing research topic in the fields of ubiquitous and pervasive computing, healthcare, and ambient-assisted living, among others. Several methods have been proposed for HAR, from traditional machine learning algorithms [31, 124, 212] to current Deep Learning approaches [42, 97, 160, 222]. With either approach, supervised learning is commonly used to learn models that classify activities based on annotated sensor data collected from an instrumented testing environment, e.g., a smart home, wearable IMU sensors. Generally, HAR implementations include data collection, pre-processing, data segmentation, feature extraction, and classification.

Some previous studies on HAR noticed that conventional methods used within the HAR pipeline can lead to accuracy overestimation. Hammerla and Plötz [98] proved that standard k-fold cross-validation (CV) is biased due to statistical dependence between data samples, especially when using sliding windows for data segmentation. The problem was also mentioned in [3, 58, 59, 80, 111], although these papers focused on different goals, and did not go into analyzing the problem in detail.

Surprisingly, it is a widely used ongoing practice within the HAR domain. Table 3.1 shows a list of recent studies where *sliding-window segmentation* and *random k-fold CV* are used in the HAR pipeline. While not exhaustive, this list includes remarkable works from previous years, and a growing number of present-day studies. These works have been published in top journals or presented at top conferences, which have created a high impact in the HAR community.

In this work, we evaluated the effect of sliding-window data segmentation and random splitting on model accuracy. We used datasets with different data modality: CASAS [48] binary motion sensors, and MHEALTH [11] and PAMAP2 [175] on-body inertial sensors. Likewise, we applied classification models of different nature, including Random Forest (RF), Graph Neural Networks (GNNs) were applied. The results show that, independently of the type of data and the chosen model, the reported accuracy is highly overestimated following the aforementioned approach.

Table 3.1: HAR studies following a sliding-window data segmentation and random training/test split validation approach.

Authors	Year, (Citations)	Journal/Conference	(Impact Factor)
Khalifa *et al.* [116]	2017, (159)	IEEE Transactions on Mobile Computing	(6.1)
Micucci *et al.* [148]	2017, (481)	Applied Sciences	(2.8)
San-Segundo *et al.* [181]	2018, (101)	Engineering Applications of Artificial Intelligence,	(7.8)
Wang *et al.* [225]	2018, (29)	Smart Health	(5.1)
Mutegeki and Han [153]	2020, (268)	ICAIIC	-
Ni *et al.* [159]	2020, (20)	Sensors	(3.8)
Gupta [90]	2021, (74)	International Journal of Information Management Data Insights	-
Li *et al.* [129]	2021, (9)	UBICOMP 2021	-
Mekruksavanich and Jitpattanakul [144]	2021, (196)	Sensors	(3.8)
Bouchabou *et al.* [26]	2021, (24)	Communications in Computer and Information Science	-
Gómez Ramos *et al.* [86]	2021, (19)	Sensors	(3.8)
Zimbelman and Keefe [262]	2021, (14)	PLOS ONE	-
Yan *et al.* [238]	2022, (14)	IEEE International Conference on Bioinformatics and Biomedicine	-
Wang *et al.* [223]	2022, (28)	IEEE Sensors Journal	(4.3)
Huang *et al.* [106]	2022, (40)	IEEE Transactions on Mobile Computing	(6.1)
Luo *et al.* [136]	2023, (21)	IEEE Transactions on Mobile Computing	(6.1)
Wu *et al.* [230]	2023, (10)	Knowledge-Based Systems	(8.1)
Garcia-Gonzalez *et al.* [78]	2023, (15)	Knowledge-Based Systems	(8.1)

The main contribution of this chapter is twofold: (1) It provides an extensive survey of recent papers in the HAR domain that employ the flawed methodology, showing the importance of warning the community. (2) it provides a set of experiments using different types of HAR datasets and different types of supervised machine learning approaches, evidencing that the problem persists in all cases and configurations.

The remainder of this chapter is as follows. Section 3.2 describes the methodological issues in conventional approaches for HAR. Section 3.3 presents an extensive review of recent related work on HAR that incurs this problem. Section 3.4 describes our experiments, presents the results and discussion focused on showing the effects of the problem. Section 3.5 presents the lessons learned. Finally, Section 3.6 presents the conclusions.

3.2 Model performance overestimation

Data segmentation following some windowing technique is the conventional approach for HAR [122, 170], and sliding windows are the most widely adopted approach [10]. The windows can be overlapping or non-overlapping, where the length is defined in ***t*** seconds or ***s*** number of sensor readings. The input stream of sensor readings is split into windows of equal size. Then, a set of features is calculated from each window, and these features are used as input for the classification models. Fig. 3.1 shows the window-based data segmentation.

One way to evaluate the performance of the classifier is using a hold out part of the entire dataset, i.e., the test set. The conventional approach is to randomly split the dataset into training/test subsets at some predefined ratio (e.g., 80:20). The training set can be split further to obtain a validation subset, which is used for model selection and/or hyper-parameter optimization. The most commonly used technique to assess the performance of the classifiers is k-fold CV [58]. First, the data is split into k disjoint subsets of equal size. Then, the model is trained on k-1 subsets and evaluated on the k^{th}. This process is repeated k times with a different subset each time. The final performance of the model is the mean of all runs. The CV method assumes data samples to be Independent and Identically Distributed (i.i.d.) [6]; thus, the

way of choosing samples does not affect the classifier's performance. This is where the problem in (re)current practices for HAR lies. *When sliding windows are used for data segmentation and feature extraction, the statistical independence assumption does not hold anymore. Therefore, a random training/test split does affect the performance of the model because contiguous windows are assigned one to training and the previous and(or) next to the test set*.

The problem can be observed with greater clarity in Fig. 3.1a. Windows $w1$, $w2$, and $w3$ share the samples $s3 - s6$. Sample $s2$ is shared between $w1$ and $w2$, and $s7$ between $w2$ and $w3$. Hence, the features obtained from those windows are drawn from almost the same underlying data samples, breaking the i.i.d. assumption. Thus, if $w2$ is randomly chosen for testing and $w1$ and $w3$ are kept for training, the classifier is tested on data that has already been seen. Consequently, the performance of the classification model is overestimated. In the case of non-overlapping windows (Fig. 3.1b), the dependence between consecutive windows is less evident. However, for long-running activities (e.g., walking, standing, reading), the similarity between samples drawn in a short interval will create a strong correlation between consecutive windows [58, 98]. Hence, it is likely that consecutive windows have similar samples corresponding to the same activity. From Fig. 3.1b, if $w2$ is randomly assigned to the test set, it will be almost identical to $w1$ and $w3$ assigned to the training set. This creates an illusion of perfect accuracy because of overfitting, caused by the strong correlation between consecutive windows and random splitting of training and test sets. The models just memorize the training data instead of learning the patterns that uniquely characterize each activity. They produce the correct label because the same data has been seen during training.

Performance overestimation can be avoided by ensuring the independence between training and test sets. One option is applying Leave-One-Subject-Out CV (LOSO-CV), a variant of k-fold CV [58, 98]. In LOSO-CV, instead of randomly choosing the samples to include in each fold, the data samples belonging to one subject are used for testing, while the data from the remaining subjects are used for training. This is repeated for each subject in the experiment. This approach is more rigid than traditional k-fold CV, but it ensures the independence between training

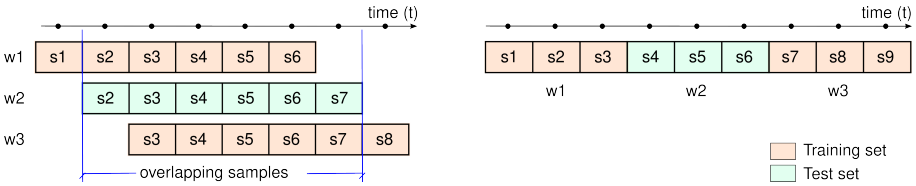


Figure 3.1: Overlapping and non-overlapping sliding-window data segmentation

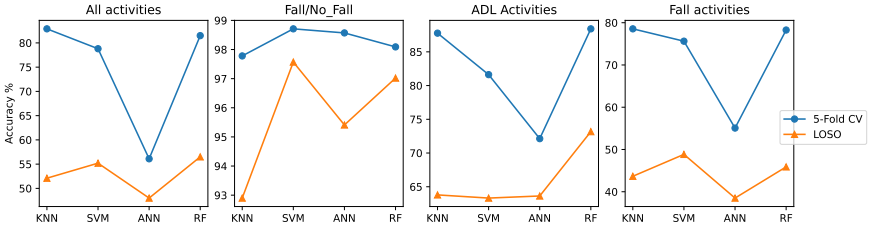
and test sets [80]. However, a LOSO-CV is not always feasible if the number of subjects in the experiments is either too small or too large [98]. Few users lead to an unrealistic view of the model performance. On the contrary, too many subjects increase the computational complexity, making the use of a model even impractical [98]. Group k-fold CV[1], a generalization of LOSO, is a valid and straightforward approach to ensure an unbiased evaluation strategy. In this case, the data samples are grouped by a third-party parameter, which can be defined on a per use-case basis, e.g., collection date, subject-id, etc. Grouped partitions ensure that data samples corresponding to the same group are not represented in both testing and training sets.

3.3 (Re)current practices in HAR

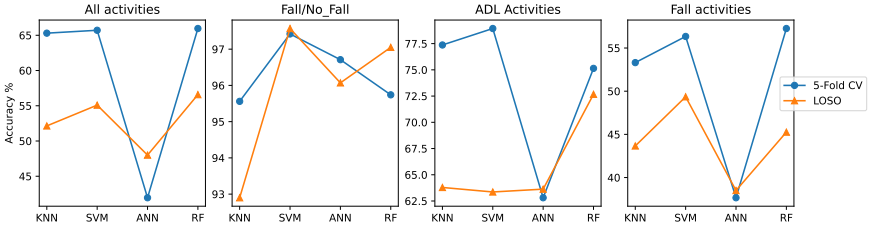
The methodological issues that lead to accuracy overestimation are an ongoing practice within the HAR community, as presented in Table 3.1. Some of those studies applied traditional machine learning algorithms for HAR [116,148,181,225,262]. The most common algorithms are Decision Trees (DT), K-Nearest Neighbours (kNN), Naive Bayes (NB), Support Vector Machine (SVM), Random Forest (RF), Hidden Markov Models (HMM), and Multilayer Perceptrons (MLP). They use different datasets, e.g., UniMiB-SHAR [148], HHAR [193], containing 3-axial accelerometer and/or gyroscope data collected using wearables or smartphones while people are performing different activities. All these studies used sliding-window data segmentation with different degrees of overlap. Then,

[1] https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation-iterators-for-grouped-data

they evaluated their models following the standard k-fold CV, and some of these studies also applied LOSO evaluation [148, 181, 225]. The results with LOSO show a significant drop in performance. However, such a drop is attributed solely to the variability in the way that different subjects perform the same activities, while the bias because of statistical dependence between consecutive windows and random training/test splits is overlooked. The bias introduced is independent of the input features (Fig. 3.2), feature extraction, and the normalization technique (Fig. 3.3).



(a) Raw signal feature vector



(b) Magnitude feature vector

Figure 3.2: 5-fold CV vs LOSO: reported accuracy comparison from Micucci *et al.* [148]

Other studies rely on Deep Learning approaches for HAR [90, 129, 144, 153, 159]. The most common approaches are CNNs, LSTMs, or the combination of both. The datasets used in these studies are UCI-HAR [5], WISDM [228], which also contains 3-axial acceleration and gyroscope data. They use sliding windows and evaluate their models using random training/validation/test splits, k-fold CV, and LOSO. The results also show a significant drop in performance using LOSO with respect to random partitioning and standard k-fold methods. It is important to note, in [144], that the performance is overestimated using both overlapping

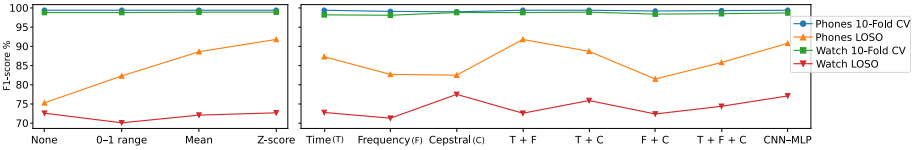


Figure 3.3: Reported F1-Score of an RF classifier from San-Segundo *et al.* [181].

and non-overlapping sliding windows (Fig. 3.4).

The works of Bouchabou *et al.* [26] and Gómez Ramos *et al.* [86] use a different type of data, binary motion and contact sensors, from the CASAS benchmark dataset. Specifically, the Aruba and Milan datasets are used in those studies. Both approaches applied overlapping sliding windows of different sizes and random data partitioning with different ratios. Although both works use different approaches, their results are comparable, F-score above 95%. More seriously, Bouchabou *et al.* [26] claim a "better generalization" obtained by means of a random shuffle before splitting, overlooking the negative effect of random splits after sliding-window data segmentation.

Yan *et al.* [238] propose a HAR model based on GNNs. They used data from the MHEALTH, PAMAP2, and their own dataset TNDA-HAR. The raw input data is transformed into a graph representation based on the Pearson correlation coefficient between the sensors channels' signals. Each channel represents a graph vertex, and a correlation of a pair of vertices above 0.2 implies an edge. A GNN model is trained to

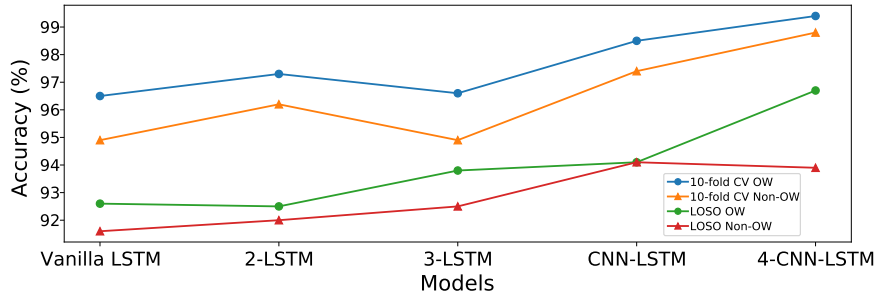


Figure 3.4: Reported results from Mekruksavanich and Jitpattanakul [144].

encode the graph, followed by two fully connected layers as the final classifier. The authors report accuracies of 98.18% for PAMAP2, and 99.07% for MHEALTH. But those accuracies are overestimated due to the use of random training/test sets splits, which is seen in the source code provided by the authors.

Wang *et al.* [223] compared the effect on performance of different data augmentation methods using UCI-HAR, USC-HAD [140], MotionSense [253] and MobiAct [36] datasets. They segmented the data using sliding windows with 12.5% to 50% overlap. They applied random training/test splits and additionally a subject-independent 5-fold CV for the MotionSense dataset. The F1-scores reported for all the experiments using randomly partitioned data were: UCI-HAR 98.28, MotionSense 99.35, USC-HAD 92.28, and MobiAct 98.32. In accordance with previous findings, in the experiments with a subject-independent CV for the MotionSense dataset, the F1-score dropped from 99.35 to 92.10.

Huang *et al.* [106] proposed Channel-Equalization-HAR, a variation of the normal CNNs. They used UCI-HAR, OPPORTUNITY [177], UniMiB-SHAR, WISDM, PAMAP2, and USC-HAD datasets. The data were segmented using sliding windows of different sizes with 30%, 50%, 78% overlap. Then, they applied a random train/test split using a 7:1:2 ratio. The reported F1-score for UCI-HAR was 97.12%, similar to the ones reported in [144, 223]. Likewise, the reported accuracy on the WISDM dataset was 99.04%, even higher than the one reported in [90] which followed the biased approach. The same can be observed for the USC-HAD dataset with a reported F1-score of 98.93%, higher than reported in [223] which also has the problem. With the PAMAP2 dataset, the reported F1-score is 92.18%, which is similar to our own experiments using random training/test set splits, shown later in Section 3.4.

Luo *et al.* [136] proposed a Binarized Neural Network for HAR, which moves the computation to the edge. They used the Radar HAR dataset [137], UCI-HAR and UniMib-SHAR datasets, applying sliding-window data segmentation followed by a random training/test split, in an 80:20 ratio for the Radar HAR dataset, and a 70:30 ratio for the UCI-HAR and UniMib-SHAR datasets. The F1-score reported for the Radar HAR dataset is 98.6%. The reported F1-score for the UCI-HAR dataset is 98.1%, similar to the ones reported in [106, 144, 223] which followed

the same biased approach. The reported F1-score for the UniMib-SHAR dataset is 93.3%, even higher than the one reported in [148] which is also overestimated.

Wu *et al.* [230] proposed a spatio-temporal LSTM model using data from a pedal wearable device attached to the shoe's tongue area. The approach combines a GNN model for the spatial features and an LSTM for the temporal patterns to recognize five different activities. A sliding-window of 200 samples was used for data segmentation. The segmented data was randomly split into training/test sets. The reported results show a perfect F1-score of 1.0 for the Sitting and Down the Stairs activities, 0.96 and 0.97 for Standing and Walking, respectively, and 0.83 for Up the Stairs. Once again, the followed approach shows overestimated results.

Garcia-Gonzalez *et al.* [78] used their own dataset containing accelerometer, gyroscope, magnetometer, and GPS data from smartphones. They evaluated different traditional ML algorithms: SVM, DT, MLP, NB, k-NN, RF, and Extreme Gradient Boosting (XGB). Data segmentation and feature extraction were performed using sliding windows from 20 to 90 seconds with 1-second step size, i.e, at least 95% overlap. Then, a stratified k-fold CV was used to evaluate the different models. While keeping a similar distribution of the samples per class, this method does not prevent consecutive windows from being assigned to two different folds. Hence, the reported accuracy, 92%, is overestimated.

3.4 Unbiased model evaluation

This section shows that the same approach can lead to a considerable difference in accuracy depending on the data segmentation and training/test set partition strategies.

3.4.1 Datasets

We evaluate whether the problem described in Section 3.2 affects in the same way datasets with different data modalities. We used the **MILAN** dataset [50] from the CASAS benchmark dataset collection[2], which contains binary data from motion and contact sensors, and the

[2]CASAS benchmark dataset collection: http://casas.wsu.edu/datasets/

PAMAP2 [175] and **MHEALTH** [11] datasets, which contain accelerometer, gyroscope, and magnetometer data from wearable on-body sensors.

MILAN The sensors mounted in this smart home testbed environment included 28 motion, 3 door contact, and 2 temperature sensors. The activities' "start" and "end" are annotated. Samples that fall outside these markers have been assigned the *"other"* class. This dataset is highly imbalanced, with the *"other"* being the dominant class.

PAMAP2 This dataset contains data collected from 9 subjects doing 12 different physical activities, using IMUs attached to the wrist, chest, and ankle while performing everyday, household, and sport activities. Each sensor includes two accelerometers, one gyroscope, and one magnetometer, producing 3-axial data at a sampling rate of 100Hz.

MHEALTH This dataset contains data of 10 volunteers performing 12 physical activities. The sensors were placed at the subjects' chest, right wrist, and left ankle. The data comprise 3-axis accelerometer, gyroscope and magnetometer signals collected at a sampling rate of 50Hz. The sensor placed at the chest also provides 2-lead ECG measurements, but those data points were not used in the experiments.

3.4.2 Data segmentation and feature extraction

We perform data segmentation using a fixed-size sliding-window approach, proposed in [122] and expanded in [170].

MILAN We used a window of k sensor events because binary/motion sensors do not fire a sample at a constant rate. The window size was 30, with a step size of 1, based on empirical evaluation as presented in [122, 226]. The windows were created based on the collection date to facilitate the independence between training and test sets during model evaluation. The last sensor event in the window defines the label and the preceding events in the window define its context. Following this approach, we can have a prediction every time a new sensor event arrives, achieving a near-real-time recognition. Then, feature vectors

are calculated for each window following the approach presented in [122, 226]. We transformed the *hour-of-day* and *day-of-week* to sine and cosine pairs to capture the equidistant relationship between time-based cyclical values.

PAMAP2 This dataset was segmented using sliding windows of 5.12 seconds with 1-second shift, following the approach of its original publication [175]. Since the data were sampled at 100Hz, the windows span 512 sample readings with a step size of 100 samples. This data were used to train a classification model based on GNNs. Therefore, the training data were transformed into a graph representation where the vertices correspond to the different channels of the sensors' signals. The edges are defined by means of the Pearson Correlation Coefficient between the channels, where a correlation threshold above 0.2 implies an edge between two channels.

MHEALTH Following the protocol in [4, 5], we segmented the data using a sliding-window of 2.56 seconds with 50% overlap. Since this dataset was sampled at 50Hz, the windows have 128 samples with 64-sample overlap. For feature creation, we first transformed the window data into a graph representation in the same way as described for the PAMAP2 dataset. Then, the activity graphs were used to train a GNN-based classification model.

3.4.3 Classification models and evaluation strategies

To measure the effects of the biased approach using classification models of different nature, we trained an RF and a GNN-based classifier. The RF classifier uses binary sensor data, and the GNN-based model uses on-body IMU sensors.

MILAN After feature extraction, the model was evaluated using the 5-fold CV approach. First, we partitioned the data randomly using the *StratifiedKFold* class, from the scikit-learn Python library [164], with the *shuffle* parameter set to *True*. Then, in a second experiment, we used the *StratifiedGroupKFold* CV scheme [164]. This scheme splits the data into

folds with non-overlapping groups, preserving the percentage of samples per class. In our case, the groups were determined by the *collection_date* of the raw data samples.

PAMAP2 and MHEALTH Most of the presented studies followed a conventional Deep Learning approach for HAR, e.g., CNN, LSTM, or a combination thereof. To check if the overestimation bias affects other Deep Learning models, we trained a 3-layer GNN implemented using the Graph Convolution presented in [152], followed by two fully connected layers and a softmax layer for final classification. We split our segmented data into a 6:2:2 ratio used for training, validation, and final model evaluation, respectively. We first partitioned the data randomly, and later on, we used the StratifiedGroupKFold approach, determined by *subject_id*. We did the same for both the PAMAP2 and MHEALTH datasets. In the case of PAMAP2, subjects 101 and 107 were used for validation, subjects 103 and 105 for testing, and the remaining subjects for training. For the MHEALTH dataset, subjects 6 and 10 were used as validation set, subjects 2 and 9 for testing and the rest for training. After hyper-parameter optimization and finding the best parameter combination, the model was updated with the entire training data, including the training and validation subsets. The model was evaluated using the hold-out test set.

3.4.4 Results and discussion

Table 3.2 shows that *window-based data segmentation* and *random data splits* lead to misleading results in all datatsets and for different type of classification models.

Table 3.2: Classification performance comparison on MILAN, PAMAP2 and MHEALTH datasets.

Partitioning	MILAN (RF)		PAMAP2 (GNN)		MHEALTH (GNN)	
	b. acc	f1-score	b. acc	f1-score	b. acc	f1-score
Random (biased)	93.53	86.74	89.36	90.19	98.00	97.99
Grouped (unbiased)	55.59	58.49	81.59	81.73	87.92	87.36

MILAN The performance with random splits is higher in all sixteen activities (Fig. 3.5), in concordance to Bouchabou *et al.* [26] and by Gómez Ramos *et al.* [86]. These results show that in both experiments the minority classes (*morning_meds*, *evening_meds*) are misclassified. The reason is because those activities occur in the same room, triggering the same set of sensors. However, it is important to point out that the biased model classified these activities with an accuracy over 88%. On the contrary, the accuracy on the same classes with the unbiased approach is under 10%. Similarly, the same effect occurs with the majority class, *"other"*. While the biased model produced an accuracy of 74% on the *"other"* class, the corrected model just obtains a 32% accuracy on this pseudo activity.

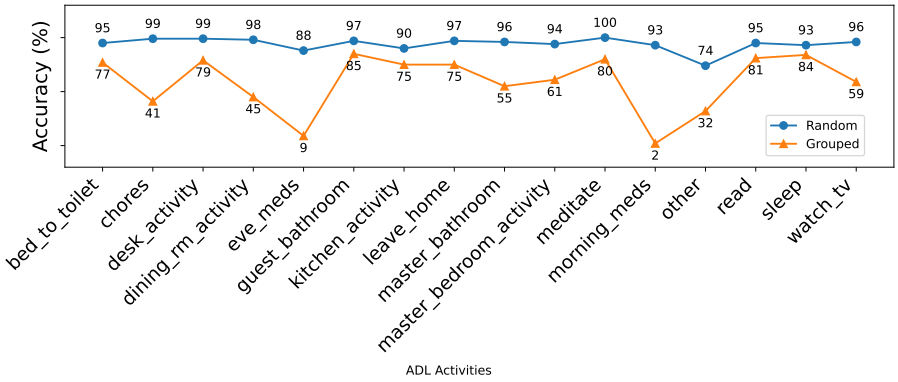
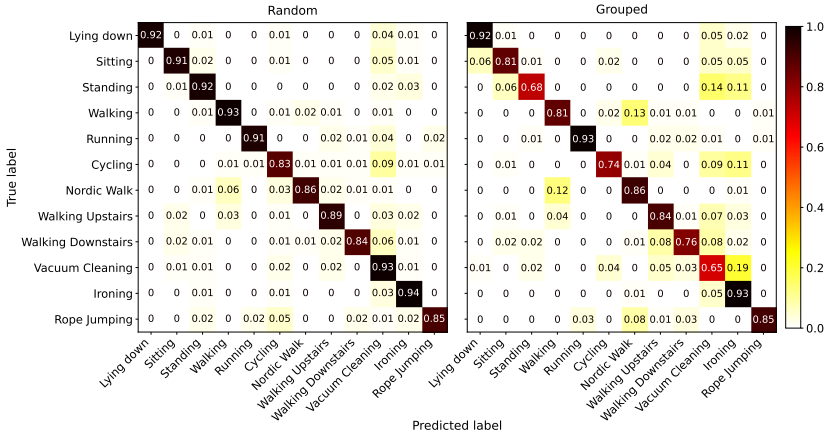


Figure 3.5: Accuracy obtained for each activity on the Milan dataset.

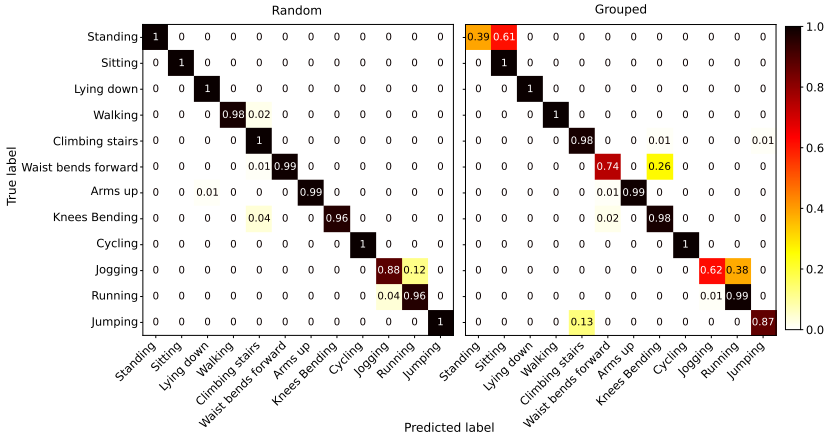
PAMAP2 and MHEALTH The results obtained with the GNN-based classification model on PAMAP2 and MHEALTH datasets also show a performance overestimation when the model is trained and evaluated on randomly partitioned data. For the PAMAP2 dataset, the accuracy and F1-score dropped $\approx 9\%$, when the data was partitioned following a group-based approach. In MHEALTH, the performance decreased $\approx 10\%$ (see Table 3.2). The confusion matrices for the PAMAP2 and MHEALTH datasets are shown in Fig. 3.6a and 3.6b.

They show that GNN-based classification models produce consistent results on both datasets. The corrected models misclassify the *standing* activity in both the PAMAP2 (68%) and MHEALTH (39%) datasets. In

contrast, with the biased approach the accuracy on the *standing* activity increases to 92% for PAMAP2 and a perfect 100% for MHEALTH. These results confirm that the "perfect" accuracy reported by Yan *et al.* [238] is overestimated due to the windowing mechanism and random training/test set splits.



(a) PAMAP2



(b) MHEALTH

Figure 3.6: Confusion matrices of the accuracy on PAMAP2 and MHEALTH datasets.

3.5 Lessons Learned

Model performance is affected by the data partition strategy that follows the window-based data segmentation. The results show that the performance of the models on randomly partitioned data is overestimated, regardless of the dataset and classification model used. The data imbalance negatively affects the performance of the classifier, but its effect is more acute when the independence between training and test sets is guaranteed. Conversely, it may be unnoticed following a biased approach. If data segmentation is performed using a windowing mechanism, the independence between the training and test sets, or between folds in CV, must be carefully considered. Hence, data partitioning must not be performed at random.

The group-based approach used in this work is a good alternative for guaranteeing an unbiased model evaluation. Splitting the data by a third-party parameter, chosen on a use-case basis, gives the flexibility to create unbiased scenarios for evaluation even for single-subject datasets.

3.6 Conclusions

Our study reviewed HAR works with approaches that lead to model performance overestimation to raise awareness of the HAR community about this ongoing problem. Due to publications with biased overestimated results, fair approaches with correct unbiased methods may be disregarded because of erroneously perceived low accuracy. We described and explained the downside of using sliding-window data segmentation and feature extraction, followed by a random k-fold cross-validation for model evaluation. We identified previous studies, including present day publications, where the reported performance is overestimated. The findings suggest that this is a recurrent practice with negative effects that are often disregarded by HAR practitioners.

Importantly, we are not implying that sliding windows and k-fold cross-validation should not be used in HAR. Those are well-established methods whose usage has been empirically justified. However, the data samples assigned to each fold should not be chosen at random. Our experiments used different classification model types and datasets with

different distributions, nature, and characteristics, proving that the bias introduced by the discussed issue is independent of the data modality and the classification model. The performance drop with an unbiased evaluation is significant to the point where those highly overestimated models would become impractical or even useless in real-world settings.

Chapter 4

Graph contrastive learning for Human Activity Recognition

Human Activity Recognition has achieved notable improvements with the emergence of deep learning models for automated feature extraction. Those models allow the extraction of complex translation-invariant features and to exploit the temporal dependencies from sensors' time series data. This work posits additional dependencies between sensors beyond the time dimension, such as physical proximity, which are equally important for the characterization of human activities. We leverage such spatial dependencies by modeling them as a graph. Using Graph Neural Networks (GNNs), we learn global and local representations of the intra- and inter-sensor dependencies. We empirically show that by maximizing the mutual information between the local and global representations, the performance of the recognition models can be significantly improved. Our results show a clear improvement over previous works based on CNNs, LSTMs, Attention-based, and other more complex GNN-based architectures.

4.1 Introduction

Human Activity Recognition (HAR) is a field that has attracted the attention of academia and industry for several years. In the early years of HAR, the problem of recognizing human activities was addressed using classical machine learning approaches, e.g., Decision Trees, Support Vector Machines, Naïve Bayes, and k-Nearest Neighbors [5,11,175]. Later on, deep learning approaches based on Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) have been extensively applied in the field of HAR, showing significant results [42, 97, 160, 222, 244]. Those approaches mainly exploit the relationships across the data in the

time dimension. On the one hand, CNNs are used for feature extraction to find patterns which are invariant to translations across different segments [97]. On the other hand, LSTMs, a type of RNN, only exploit temporal dependencies [160].

The current study, in addition to the temporal patterns, leverages the spatial dependencies in the data and introduces them as relational inductive biases represented as a graph structure. The assumption is that signals collected from smartphones and wearables, while people perform different activities, have additional dependencies that span beyond the temporal dimension. These spatial dependencies between intra- and inter-sensor data channels can be represented as a graph. GNNs have been successfully applied to learn complex relationships in graph-structured data [14, 62, 69, 94]. Then, such relationships can be learned by a GNN-based model and subsequently used to characterize human activities.

Although previous work introduced the idea of applying GNN-based models in the HAR domain [105,227,238], these works do not reflect the real performance of the models due to a methodological issue within the HAR pipeline that causes biased results. Such biases are introduced by the evaluation strategy that follows a sliding-windows data segmentation approach [98, 202]. Therefore, the real value that GNN-based models can bring to the field was not clearly established from these works.

In this chapter, we formulate HAR as a multi-class graph classification problem. We apply two different graph construction methods from raw sensor signals, which leverage *global* and *local* dependencies in the time series data. While global dependencies are captured by correlating the sensors' channels from the entire sequence, local dependencies capture the correlations within smaller time frames. Thus, global and local graphs represent different views of the same human activities. Then, a contrastive learning approach to maximize mutual information between the two views shows a significant performance gain for HAR models.

We evaluate our proposed approach on four benchmark datasets, UCIHAR [5], MHEALTH [11], PAMAP2 [175], and REALDISP [12]. These datasets comprise accelerometer, gyroscope, and magnetometer data from Inertial Measurement Unit (IMU) sensors. Our GNN-based models produce accuracies over 90% on UCIHAR, MHEALTH, and REALDISP

datasets, and ≈ 87% accuracy on PAMAP2. These results surpass previous approaches based on CNNs, LSTMs, Attention-based, and other more complex GNN-based architectures. It shows that GNNs have potential in the HAR domain, allowing the discovery of different relationships in the data beyond the temporal dimension.

The main contributions of this chapter are: (i) A new set of state-of-the-art GNN-based models for HAR with significantly reduced complexity compared to existing approaches, (ii) Global and local graph representations of the inter- and intra-sensor dependencies, and (iii) A contrastive learning approach that leverages the global and local dependencies to enhance the performance of HAR models, showing a clear improvement over other state-of-the-art methods.

The remainder of this chapter is as follows. Section 4.2 shows the related work in this field and discusses the main differences of our work with the existing GNN-based approaches for HAR. Section 4.3 describes and formalizes our approach. Section 4.4 describes the experiments performed on Human Activity Recognition using GNNs. Section 4.6 presents and discusses the results. Finally, Section 4.7 presents the conclusions.

4.2 Related Work

Most of the existing work on HAR using GNNs is based on skeleton graphs extracted from video sequences [35, 130, 237]. In recent years, the community started to explore the capabilities of GNNs for HAR using IMU sensors from smartphones or other types of wearable devices [105, 149, 227, 238].

Huang *et al.* [105] propose a shallow CNN that performs cross-channel communication for HAR. The cross-channel communication is achieved by message passing within a GNN. They propose a 3-layer CNN followed by a fully connected layer for final classification. However, they encode the output of each CNN layer via a graph attention network over a fully connected graph where each channel of the CNN is considered a node. Then, the encoded signals are sent back to the next CNN layer. They evaluated their proposed approach by comparing a 3-layer CNN, 6-layer CNN, and a 3-layer CNN + GAT. The results show

that cross-channel communication via GNN message passing outperforms a deeper CNN architecture. Since our architecture is based only on a 3-layer GNN, the number of parameters of our model is reduced by half compared to this work.

Mohamed *et al.* [149] propose HAR-GCNN, a deep graph CNN to classify human activities. Their approach is based on the assumption that people perform certain activities in a chronological order. Hence, they exploit the correlation between chronologically adjacent activities to predict unlabeled or future activities. They create a fully connected graph by taking a number of consecutive samples in a time window t, where each sample represents a node in the graph. Then, they randomly mask a number of activity labels and add noise to the features of some randomly selected nodes. The model is trained to predict the missing labels. Taking n consecutive activities and randomly masking some of the labels is not realistic because the nodes representing future activities may be used to predict the past ones. In a realistic implementation, only the most recently executed activities can be used to predict the next one (or at most n in the near future). Hence, masking can not be made at random. In addition, the authors assume that activities are usually executed by people in a certain particular order. Such assumption can lead to models that learn the sequence in which activities were performed rather than real patterns that characterize each activity [23]. This effect is more pronounced in datasets with strict data collection protocols [23], but can be less noticeable in a more natural setting with a loose order of activities. From the results reported in this chapter, it is clear that the model is learning the order on which the activities were performed. The results on PAMAP2 dataset, which has a strict data collection protocol, are almost perfect. On the contrary, the performance drops $\approx$10% points on Extra-Sensory dataset, in which subjects were not instructed on how to perform the activities. Our work does not make any assumption on the order on how the activities were performed. Thus, the graph construction does not depend on the sequence of data samples but on the correlations between sensors' data signals.

Yan *et al.* [238] propose a model based on 4 blocks composed of 4 Chebyshev [55] layers, followed by a normalization layer and a Leaky ReLU activation function. At the end, two fully connected layers are used

as a classifier. Their model architecture is composed of 18 layers with 5.29M trainable parameters in total. They also used the MHEALTH and PAMAP2 datasets. The raw input data signals are transformed into a graph representation based on the Pearson correlation coefficient matrix. A correlation above 0.2, as a threshold, implies an edge between those two signals. The authors reported accuracy above 98% for PAMAP2 and MHEALTH datasets. The work of Wang *et al.* [227] is built upon [238]. It also computes the adjacency matrix for the graph representations based on the Pearson correlation coefficient. What makes this work interesting is the combination of message passing GNNs with multi-head attention to learn channel-wise correlations.

Unfortunately, some results presented in the aforementioned works [227, 238] are misleading due to accuracy overestimation caused by a biased evaluation protocol. In those studies, the data segmentation is performed using the well-established sliding-windows approach. The problem arises when the training/validation/test sets are determined at random. The sliding-window approach introduces a high correlation between two consecutive windows, and therefore, the way in which training and test samples are chosen affects the performance of the models, as discussed in detail in [98, 202]. We show that, with a proper evaluation protocol, the performance reported in [238] drops 14 percentage points on average. This highlights the need for a thorough exploration of GNN-based models for HAR.

The main difference between our work and these approaches is that we add a local graph representation computed on per-window Pearson correlation coefficients. Then we contrast the global and local representations to maximize the mutual information between them. Since our model is shallow, the complexity is reduced to 0.43M parameters, achieving a significantly higher performance.

4.3 Graph Classification for Human Activity Recognition

In this section, we describe and conceptualize our approach. First, we describe how we model human activities as graphs. Then, we describe

how we formulate HAR as a graph classification problem.

4.3.1 Human Activities as Graphs

Given the raw data signals from the IMU sensors in the form of a matrix $X \in \mathbb{R}^{n \times m}$, where n is the number of observations and m is the number of channels in the raw signals, an activity graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, l)$. The set of nodes or vertices is given by $\mathcal{V}$, where each vertex represents a channel of the signal. The edge set is given by $\mathcal{E} = \{(i,j) \in \mathcal{V} \times \mathcal{V} : w_{ij} \neq 0 \}$, where w_{ij} is the weight of the edge connecting node i to node j, i.e., the strength of the connection between two signal channels. The label $l \in \mathbb{L}$ denotes the activity associated with a graph $\mathcal{G}$, where $\mathbb{L}$ is the set of all activity labels in the dataset.

The nodes $\mathcal{V}$, given by the different channels of the signals, are known beforehand. The edge set $\mathcal{E}$ and its associated weights w_{ij} have to be estimated from data observations. Initially, we assume a fully connected graph and the number of edges $|\mathcal{E}| = m(m-1)$. Then, the edge set is refined by choosing a subset of edges from $\mathcal{E}$ based on data statistics, e.g., covariance/kernel matrices.

In our work, we used the correlation matrix to build the graph topology and estimate the edge weights from the raw data signals. Similar to [238] and [227], we rely on the Pearson correlation coefficients matrix. This method has been widely used to discover the topology and estimate functional connectivity between the regions of the brain, and it is recognized to be one of the most used methods in that field [103, 169, 232]. The Pearson correlation coefficient is defined as the covariance of two random variables, in our case $v_i, v_j \in \mathcal{V}$, normalized by the product of their standard deviations.

$$\rho_{(v_i,v_j)} = \frac{cov(v_i, v_j)}{\sigma v_i \, \sigma v_j} \tag{4.1}$$

Using equation (4.1) we obtain a correlation matrix $X_{corr} \in \mathbb{R}^{m \times m}$, where a zero value implies no (linear) correlation between two random variables, i.e., there is an edge connecting two nodes *if and only if* the correlation coefficient is different from zero. The correlation coefficients in X_{corr} give us the strength of the correlation; hence, they can be used as the weights of the edges connecting a pair of nodes in our

activity graphs. However, instead of using the complete correlation matrix for computing the edges and their weights, we used a sparse version by setting a threshold τ. The threshold allows filtering out noisy connections in our activity graphs. According to [169], when using Pearson's correlation to define the graph connectivity, the threshold can act as an L1-regularization term included in other approaches, e.g., Graphical Lasso. Taking into account that the correlation coefficients can be negative, the edge weight can be defined as follows:

$$w_{ij} = \begin{cases} |\rho_{(v_i,v_j)}|, & \text{if } |\rho_{(v_i,v_j)}| > \tau. \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

The correlation coefficient values are considered weak for values $|\rho_{(v_i,v_j)}| < 0.2$, moderate for values of $0.2 \leqslant |\rho_{(v_i,v_j)}| \leqslant 0.3$, and strong for values of $|\rho_{(v_i,v_j)}| > 0.3$ [100, 199]. Therefore, the threshold τ was set to 0.2 to filter out weak connections and include moderate and strong categories in terms of correlation strength.

Applying the approach described above, we define two different types of activity graphs: $\mathcal{G}_{global}$ and $\mathcal{G}_{local}$. The first type of graph, $\mathcal{G}_{global}$, is based on a single correlation matrix calculated from the entire training set. Then, this matrix is used to define the edges and weights for all activity graphs. The difference between the activity graphs constructed using this approach is in the feature values of the nodes. The features for each node are given by the sensor readings within a temporal window t. For the second type of graphs, $\mathcal{G}_{local}$, a different graph is created for every time window t. Hence, a correlation matrix, $X_{corr}^{(t)}$, is calculated from the signal of the channels for each window, and this matrix is used to define the edges and the weights of each graph. The motivation to create a different graph for each time window is that every activity is expected to have an underlying topology that describes the structural relationships between the signals. Therefore, since every time window corresponds to a specific activity, the model can learn the hidden patterns in each activity class. This procedure is repeated for all the windows in the training, validation, and test sets.

4.3.2 Classification of Human Activity Graphs

We formulated HAR as a supervised multi-class graph classification problem. In the activity graph classification setting, given a set of graphs $\{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_N\} \subseteq \mathbb{G}$ with their corresponding activity labels $\{l_1, l_2, ..., l_N\} \subseteq \mathbb{L}$, a GNN-based classifier should learn a vector representation of a graph $h_{\mathcal{G}}$ that allows it to predict its associated label, $l_{\mathcal{G}} = g(h_{\mathcal{G}})$ [236].

In general terms, GNNs learn vector representations of a node, h_{v_i}, by iteratively aggregating the vector representations of its neighbors, $h_{v_j} : v_j \in \mathcal{N}(v_i)$, and then combining the aggregated vectors with its own representation from the previous iteration $t-1$. Using the notation presented in [236], this iterative learning process can be denoted as:

$$a_{v_i}^{(t)} = \textit{AGGR}^{(t)}(\{h_{v_j}^{(t-1)} : v_j \in \mathcal{N}(v_i)\}) \tag{4.3}$$

$$h_{v_i}^{(t)} = \textit{COMBINE}^{(t)}(h_{v_i}^{(t-1)}, a_{v_i}^{(t)}) \tag{4.4}$$

Both *AGGR* (equation 4.3) and *COMBINE* (equation 4.4), may be arbitrary differentiable, permutation-invariant functions [152]. In a node classification problem, the final representation of a node ($h_{v_i}^{(T)}$) would suffice. For graph classification, we need a global function to aggregate the final representation of all nodes $v_i \in \mathcal{V}$. Once again, any permutation invariant function can be used, e.g., *sum* or *max* or other more advanced readout implementations like DiffPool [241] or SortPool [252]. Then, the vector representation of an entire graph can be defined as:

$$h_{\mathcal{G}} = \textit{READOUT}(\{h_{v_i}^{(T)} : v_i \in \mathcal{G}\}) \tag{4.5}$$

The final vector representation of our activity graphs, $h_{\mathcal{G}}$ can be passed to a final classifier, e.g., a Multi Layer Perceptron (MLP), followed by a softmax function to predict the label associated to a graph, $l_{\mathcal{G}} = g(h_{\mathcal{G}})$.

4.3.3 Contrasting global and local activity graphs representations

We propose a contrastive learning algorithm based on the two different views of our activity graphs, $\mathcal{G}_{global}$ and $\mathcal{G}_{local}$, described in section 4.3.1. The idea is to maximize the mutual information between pairwise representations of the same activity. Thus, two separate GNN-based encoders compute the global and local vector representations (embeddings) of an activity graph by means of equations 4.3, 4.4, and 4.5. Then, the tuple $(h_{\mathcal{G}_{global}}, h_{\mathcal{G}^{+}_{local}})$, represents the positive pairs. Using the same local encoder to compute the embeddings of a corrupted version of $\mathcal{G}_{local}$, the negative pairs for contrasting are $(h_{\mathcal{G}_{global}}, h_{\mathcal{G}^{-}_{local}})$. The corruption of $\mathcal{G}_{local}$ is performed by a random permutation of the nodes, a random permutation of the edges, or both. Finally, both, the global and local embeddings of the activity graphs are concatenated and passed through two fully connected layers for classification. Figure 4.1 shows and overview of the proposed contrastive learning approach for HAR.

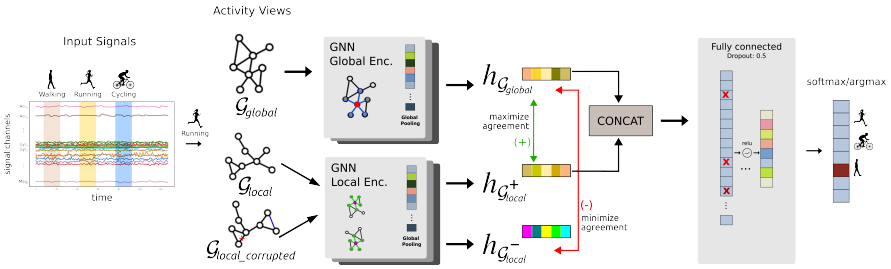


Figure 4.1: Contrastive global and local activity graph representations for HAR.

4.4 Experiments

First, using the $\mathcal{G}_{global}$ and $\mathcal{G}_{local}$ graph representations independently, we evaluated the ability of GNNs to classify the human activities. Then, combining both representations, we followed a contrastive learning approach to train a model that maximizes the mutual information between activity graphs that represent the same activities, or minimizes it otherwise.

4.4.1 Datasets

The datasets used in this study are UCIHAR [5], MHEALTH [11], PAMAP2 [175], and REALDISP [12]. They comprise accelerometer, gyroscope, and magnetometer data collected from smartphones and wearable devices while people perform different activities.

UCIHAR. The data were collected from a smartphone placed on the waist of 30 volunteers performing the activities: *Walking, Walking Up, Walking Down, Sitting, Standing* and *Lying*. These data were collected at a constant sampling rate of 50Hz. Subjects 2, 6, 12, 19, and 26 were used for validation; data from subjects 5, 8, 10, 14, 20, and 21 were used for testing; and the rest for training.

MHEALTH. This dataset contains data from 10 volunteers performing the activities: *Standing, Sitting, Lying down, Walking, Climbing stairs, Waist bends forward, Arms up, Knees Bending, Cycling, Jogging, Running*, and *Jumping*. The data were collected with wearables placed on subjects' chest, right wrist, and left ankle. The sampling rate was 50Hz. Two 2-lead ECG measurements on the chest were not used in the experiments. Data from subjects 6 and 10 were used for validation, subjects 2 and 9 for testing, and the rest for training.

PAMAP2. The data were collected from 9 subjects using IMUs attached to the wrist, chest, and ankle while performing everyday, household, and sport activities. The sampling rate was 100Hz. This dataset includes eighteen different activities. However, in our experiments, we only used the main twelve activities: *Lying down, Sitting, Standing, Walking, Running, Cycling, Nordic Walk, Walking Upstairs, Walking Downstairs, Vacuum Cleaning, Ironing*, and *Rope Jumping*. Data from subjects 101 and 107 were used for validation, subjects 103 and 105 for testing, and the remaining for training.

REALDISP. This dataset contains data collected from 17 subjects while performing 33 physical activities. In this study, we only used the activities *Walking, Jogging, Running, Jumping, Jump rope, Waist bends forward, Arms up, Knees to breast, Knees Bending*, and *Cycling* to be consistent with the type of activities in the previous datasets. The recordings were sampled at 50 Hz. Data from subjects 4, 6, 10, and 11 were used for validation; subjects 1, 7, 8, 9, 12, and 14 for testing; and the remaining

for training.

4.4.2 Data Pre-processing

We followed the conventional approach of splitting the data into training, validation, and test sets as described in the previous section. The splits were performed following a stratified-by-subject approach, where subjects for each subset were chosen at random. Using this scheme, the data are split into folds with non-overlapping subjects, where percentage of samples for each class is preserved. Splitting the data by subject ensures the independence between training and validation data [98, 202], which was not observed in previous works, such as ResGCNN [238]. Then, z-score normalization was applied to our training, validation, and test sets.

Next, the data were segmented following the sliding-window approach. Since the data was split by subject, the independence between sets is guaranteed, and the sliding windows approach does not introduce bias at the evaluation time. The sliding windows for UCIHAR and PAMAP2 datasets were created following the data segmentation protocol described in their original publications verbatim. In the first case, the data was segmented into windows of 2.56 seconds (i.e., 128 samples) with 50% overlap. For the PAMAP2 dataset, 5.12 seconds (i.e., 512 samples) with 1-second shift (i.e., 100 samples). In the case of the MHEALTH and REALDISP datasets, we followed the same protocol as for UCIHAR because the data were collected at the same sampling rate for both datasets.

4.4.3 Models Configuration

All our experiments are based on a 3-GraphConv-Layer GNN [152], followed by a ReLU activation and dropout after each of the first two convolutions. Then, two fully connected layers serve as the final classifier. We trained the model for 500 epochs and used early stopping if the validation loss did not improve for 100 consecutive epochs. The models were optimized using the hyperparameters with the Tree-structured Parzen Estimator algorithm [16], implemented using the hyperopt python library [17]. The learning rate and weight decay were sampled from

a *log uniform* distribution from the given lower and upper boundaries, and quantized to the specified increment value, i.e., the sampled value was rounded to the nearest multiple of increment inc. The values in the hyperparameters search space were chosen to cover the values found in seminar GNNs-related papers [55, 118, 215]. The values in the search space for hyperparameters optimization are shown in Table 4.1.

Table 4.1: Hyperparameters optimization search space.

*learning rate	*weight decay	*layer dropout	*classifier dropout	hidden channels	batch norm	AGGR	READOUT
1e-4, 1e-2 inc: 1e-5	1e-5, 1e-3 inc: 1e-6	0.0, 0.6 inc: 0.1	0.0, 0.6 inc: 0.1	32, 64, 128	True False	add mean max	add mean max

* min and max values of a range with increment inc

4.4.4 Loss functions

The baseline models, based on the $\mathcal{G}_{global}$ and $\mathcal{G}_{local}$ representations, were trained using the Adam optimizer [117] to minimize the categorical cross-entropy loss, which is used for multi-class classification problems like the one presented in this chapter.

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(p_{i,c}) \tag{4.6}$$

where N is the number of samples in the dataset or the batch size, C is the number of classes in the classification problem, $y_{i,c}$ is the ground-truth label for sample i, and $p_{i,c}$ is the predicted probability for sample i for class c.

In the case of the contrastive learning approach, the models were trained using a composite loss function that combines the *classification loss* and the *contrastive loss*. Categorical Cross Entropy Loss (eq. 4.6) was used for the supervised classification, and InfoNCE loss (eq. 4.7) was used for the contrastive part.

$$\mathcal{L}_{\text{infoNCE}} = -\log \frac{\exp(\text{sim}(z_i, c_i))}{\exp(\text{sim}(z_i, c_i)) + \sum_{j=1}^{N}\exp(\text{sim}(z_i, c_j))} \tag{4.7}$$

where N is the batch size, z_i is the representation of a global positive sample $h_{\mathcal{G}_{global}}$, c_i is the representation of the corresponding contrastive local positive sample $h_{\mathcal{G}_{local^+}}$, c_j is a corrupted version of c_i, representing the contrastive local negative sample $h_{\mathcal{G}_{local^-}}$, and $\text{sim}(z_i, c)$ is a similarity function (e.g., cosine similarity) between global (z_i) and local representations (c). Hence, the loss function for contrastive learning is given by:

$$\mathcal{L} = \mathcal{L}_{\text{infoNCE}} + \mathcal{L}_{\text{CE}} \tag{4.8}$$

4.5 Performance evaluation

After hyperparameter optimization, the models were updated with the entire training and validation sets. The learning rate of the best trained model was reduced by a factor of 0.1 to avoid the model learning significantly different weights. The model update was stopped when the validation loss did not improve for 10 epochs. The final model was evaluated using the accuracy and macro f1-score obtained on the holdout test set.

4.6 Results and Discussion

This section describes the results and discusses the findings obtained from the experiments conducted in this work. Table 4.2 shows the classification accuracy and macro F1-score of the models on all datasets.

4.6.1 Different graph constructions.

The first set of experiments was performed using the global and local correlation matrices described in Section 4.3.1. The global representation captures the correlation between sensor data channels across the entire training data. On the contrary, the local model captures the correlations within a time window frame. Comparing just these two models (see Table 4.2), the results are consistent across all datasets in favor of the global model. However, the difference between the global and local models for UCIHAR and MHEALTH dataset is small, in constrast to the

Table 4.2: Classification accuracy and macro f1-score of the models on all datasets. Best in bold, second best underlined.

	Params (millions) ↓	UCIHAR		MHEALTH		PAMAP2		REALDISP	
		acc↑	f1↑	acc↑	f1↑	acc↑	f1↑	acc↑	f1↑
CNN [222]	0.09	86.06	84.38	86.00	82.21	74.55	73.63	90.08	89.05
2-LSTM [144]	**0.08**	84.74	83.38	80.36	77.95	75.92	73.53	88.04	83.63
4-CNN-LSTM [144]	1.49	88.14	86.71	81.3	80.02	60.57	55.95	71.77	73.31
DeepConvLSTM [160]	2.92	89.42	87.82	83.74	79.75	80.53	77.92	88.26	89.34
Self-Attention [139]	0.43	85.00	85.00	79.00	78.00	83.00	81.00	72.00	68.00
ResGCNN [238]	5.29	83.13	83.33	86.00	84.03	82.18	81.97	77.75	74.35
Global (ours)	0.22	89.10	89.20	87.41	87.46	**86.86**	**86.36**	**93.65**	**91.52**
Local (ours)	0.09	88.42	88.80	86.56	86.61	81.29	83.12	84.03	82.59
contrastive$_{all}$ (ours)	0.43	90.81	91.07	89.66	89.82	80.44	82.28	90.37	89.91
contrastive$_{edges}$ (ours)	0.43	89.93	90.28	90.32	90.44	83.71	84.64	92.12	91.02
contrastive$_{nodes}$ (ours)	0.43	**91.43**	**91.82**	**90.60**	**91.00**	82.91	84.3	88.26	86.13

difference found for PAMAP2 and REALDISP datasets. The PAMAP2 dataset was the only one containing missing data. The results reported for the Local model on the PAMAP2 dataset come from the data that were cleaned using interpolation to fill missing values. Hence, the data windows containing mostly interpolated data points in the local model do not allow the model to properly define the topology and the strength of the connections, as opposed to the global model. Looking at global or local correlations independently, the global view allows GNNs to learn a better representation of the sensor data, even in the presence of noise like in the PAMAP2 dataset.

These results indicate that Pearson's correlation is a viable option to extract the hidden topology between signals from wearable sensors. Figure 4.2 shows the confusion matrices of the accuracy obtained with the Global model on all datasets. The model struggles to distinguish between standing and sitting activities in UCIHAR, MHEALTH, and PAMAP2. The model also confuses running and jogging activities in MHEALTH and REALDISP. This shows that there are common patterns, shared across all datasets, associated with a particular activity, and that GNN-based models are able to learn those patterns. Interestingly, the model shows consistency across datasets even for those activities that it fails to classify.

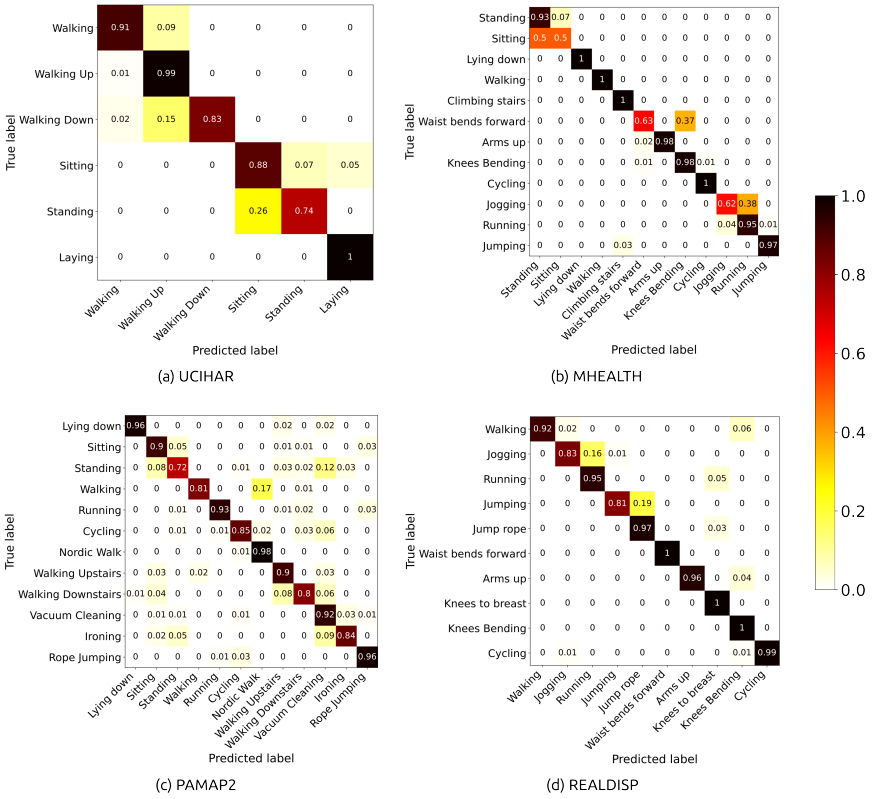
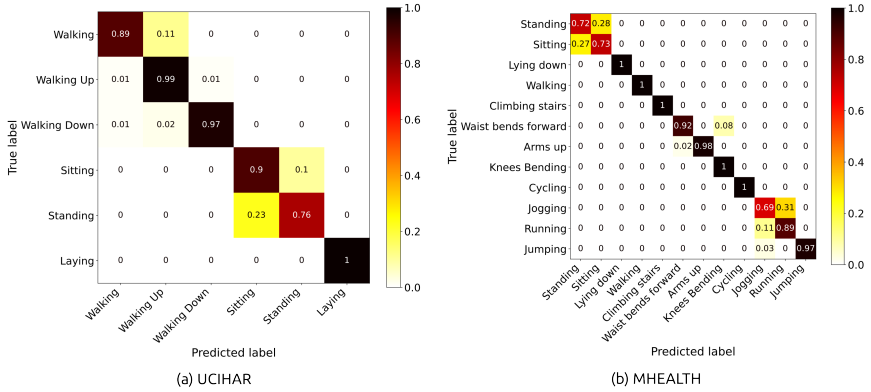


Figure 4.2: Confusion matrices of accuracy with the Global model on all datasets.

4.6.2 Contrastive learning

The results of contrastive learning experiments also show the consistency of the GNN-based models. A single perturbation, either to the nodes or to the edges, produced the best results in all cases. Compared to the local representation alone, combining it with the global representation allows the models to learn stronger vector representations that boost the performance of the classifiers. Our contrastive learning approach is suitable for HAR as it allows the model to learn the shared and the uncommon patterns between similar activities. The walking up and walking down activities involve the same body parts; and thus, they share common patterns. We argue that while the global model learns those shared patterns, the local model refines the learning process with the

patterns that are intrinsic to each time window frame and related to the way the activity is performed. This can be observed in the confusion matrix calculated with the predictions of the *Global* model in UCIHAR. The *Global* model confuses the walking up and walking down activities (Figure 4.2a), which we attribute to the fact that these activities have similar global patterns. In contrast, the contrastive learning approach on UCIHAR improves significantly on these two activities (Figure 4.3a). Similar behavior occurs in MHEALTH. The *Global* model confuses the standing and sitting, waist bend forward and knees bending, and jogging and running activities (Figure 4.2b). In this case, the contrastive learning model improves the classification of the standing and sitting activities, and removes the confusion between waist bend forward and knees bending activities almost completely (Figure 4.3b).



(a) UCIHAR (b) MHEALTH

Figure 4.3: Confusion matrices of accuracy with the Contrastive model on UCIHAR and MHEALTH datasets.

The contrastive learning approach shows significant improvement when the global and local representations have similar performance. This may be considered a prerequisite for applying the contrastive learning approach. This is the case for UCIHAR and MHEALTH datasets where the contrastive model achieves 2.65 and 3.54 percentage-point improvements, respectively. In contrast, for PAMAP2 and REALDISP datasets, where the performance of global and local models was not comparable, the contrastive learning approach did not contribute to performance gains.

The contrastive approach shows a promising direction for further

exploration. In this approach, we contrasted different views of the graphs representing the same activities. Incorporating different graph augmentation techniques for creating a richer set of views to contrast is considered part of the future work.

4.6.3 Comparison with other deep learning models

We compared our approach with other deep learning approaches for HAR reported in the literature. Namely, CNN [222], 2-LSTM [144], 4-CNN-LSTM [144], DeepConvLSTM [160], Self-Attention [139], and ResGCNN [238]. The results presented in Table 4.2 show that our *global* representation allows our model to outperform all other models across all datasets in terms of F1-score. In terms of accuracy, our *global* model surpasses all of the models on MHEALTH, PAMAP2, and REALDISP datasets by a large margin. On UCIHAR dataset, only DeepConvLSTM is on par with our model, while the others show worse results. Moreover, leveraging the mutual information between the global and local representations of the human activities, our contrastive learning approach improves the performance even further, surpassing all other models on all of the tested datasets.

Although the proposed contrastive learning approach leads to consistent performance improvements, the experiments do not explicitly disentangle the contributions of the model architecture and the learning paradigm. In particular, the same contrastive learning strategy is not applied to the baseline methods, and no systematic ablation study is conducted to isolate the effect of contrastive learning alone. As a result, the observed gains should be interpreted as the combined effect of the proposed global–local representation and the adopted training strategy, rather than as evidence that contrastive learning is the sole or dominant factor. A comprehensive investigation of this aspect, including applying contrastive learning to competing architectures and performing controlled ablations, is left as an important direction for future work.

In terms of model complexity, our smaller model, *local*, has 0.09M parameters, on par with the CNN and 2-LSTM models. However, our model exhibits a clear performance gain in UCIHAR, MHEALTH and PAMAP datasets. Likewise, our *global* model, with 0.22M parameters,

is smaller than all other counterparts, except CNN and 2-LSTM, but outperforms all of them in accuracy and F1-scores in all datasets.

Our results confirm the performance drop of ResGCNN [238] when model evaluation is correctly implemented, fixing the issue of the performance overestimation caused by random data splits leaking training data into the test set [98, 202]. We evaluated ResGCNN ensuring the independence between training, validation, and test sets by partitioning the data by subject as described in Section 4.4.2. The best result obtained with that model was 86% accuracy for the MHEALTH dataset, which is significantly lower than the ≈98% reported in [238]. Our contrastive model shows a clear improvement with an accuracy of 90.60%. It is important to point out that the difference in complexity between our models and ResGCNN is very large. Our more complex model is the one used for Contrastive Learning with 0.43M parameters, while ResGCNN has 5.29M. Our model is $\approx 91\%$ smaller but it achieves a much higher performance. This shows that very deep models, at least for these datasets, are not necessary. The performance decrease of ResGCNN shows that, even using residual connections, the oversmoothing problem, common in GNNs, still affects the learning capabilities of the model.

4.7 Conclusions

In this chapter, we leveraged the spatial dependencies between IMU sensor data channels by modeling them as a graph. Using two GNN-based encoders, we learned global and local representations of the intra- and inter-sensor dependencies and exploited those representations by maximizing the mutual information between them, following a contrastive learning approach. We evaluated our approach on four HAR benchmark datasets, showing a significant performance increase compared to previous studies, including CNNs, LSTMs, CNN-LSTMs, Self-Attention, and GNN-based models.

The results of the experiments show that the underlying structure of the IMU sensor data channels and the strengths of their connections can be modeled as a graph. In addition, they provide evidence that such graphs can be properly encoded by a GNN-based model, and the embeddings learned by the GNNs can be used downstream for Human

Activity Recognition. The results confirm that considering other types of relationships, beyond the time dimension, and incorporating spatial dependencies into the model, allows us to uniquely characterize each human activity. The best-performing models achieved an accuracy and macro F1-score above 90% for the UCIHAR, MHEALTH, and REALDISP datasets, and an accuracy and macro F1-score above 86% for the PAMAP2 dataset. This shows that GNNs are a good alternative for Human Activity Recognition.

A promising future direction is to combine GNNs with temporal models to fully leverage the spatio-temporal relationships and explore Temporal GNNs for HAR. Augmentation techniques applied to the sensor data itself can improve the contrastive learning approach further. The effects of scaling, inverting the signals, reversing the time direction, stretching, and warping the time series, together with the graph augmentation techniques, are worth investigating.

Part II

Deep Learning on Graphs for Water Distribution Networks

Chapter 5

Graph Neural Networks for Pressure Estimation in Water Distribution Systems

Pressure and flow estimation in Water Distribution Networks (WDNs) enables water management companies to optimize control operations. For many years, mathematical simulation tools have been the most common approach for reconstructing WDN hydraulics. However, purely physics-based simulations face several challenges, such as partially observable data, high uncertainty, and extensive manual calibration requirements. Consequently, data-driven approaches have gained increasing attention as a means to overcome these limitations.

In this chapter, we combine physics-based modeling with Graph Neural Networks, a data-driven approach, to address the pressure estimation problem. Our work makes two main contributions. First, we propose a training strategy based on random sensor placement, which makes the GNN-based estimation model robust to unexpected changes in sensor locations. Second, we introduce a realistic evaluation protocol that incorporates real temporal patterns and noise injection to better mimic the uncertainties inherent in real-world scenarios.

As a result, we present a new state-of-the-art model for pressure estimation, named GATRes. Our model outperforms previous studies across multiple WDN benchmarks, achieving an average reduction in absolute error of approximately 40%.

5.1 Introduction

State Estimation in Water Distribution Networks (WDNs) is a general problem that encompasses pressure and flow estimation, often using scarce and sparsely located sensor devices. WDN management companies rely on such estimations for optimizing their operations. Knowing the

state of the network at any given time enables water managers to perform real-time monitoring and control operations. The research community and practitioners working in this field have resorted to the power of mathematical simulation tools for many years to reconstruct an estimate of the system hydraulics [7, 114, 121, 146, 180, 206]. However, pure physics-based simulation approaches have to overcome the challenges of (i) data scarcity, which translates to partially observable systems, (ii) high uncertainty introduced by the large number of parameters to configure, unexpected changes in consumers' behavior reflected in uncertain demand patterns, and noisy sensor measurements, and (iii) extensive manual configuration for model calibration using metered data, which requires expert knowledge and usually hinders model reusability in a different WDN [161, 224]. The challenges associated with physics-based modeling of WDNs have motivated researchers to investigate the use of data-driven approaches, or a combination of both, to address the state estimation problem [134, 143].

Graph Neural Networks are a data-driven approach that has shown successful results in several estimation problems where data can be modeled as a graph. Since WDNs can be naturally modeled as a graph, GNNs can exploit the relational inductive biases imposed by the graph topology. As a result, GNNs have also attracted the attention of researchers in the field of WDNs. For example, Tsiami and Makropoulos, in [211], used temporal graph convolutional neural networks, a combination of Convolutional Neural Networks (CNNs) and GNNs, to extract temporal and spatial features simultaneously to detect cyber-physical attacks in WDNs. Zanfei *et al.*, in [246], used GNNs for implementing burst detection algorithms. GNNs are also used for integrated water network partitioning and dynamic district metered areas [75]. In the context of Digital Twins of WDNs, a GNN-based model is used for Pump Speed-Based State Estimation [25]. Water demand forecasting has been also addressed using GNNs [245]. Metamodeling is another interesting area where GNNs have been leveraged for the estimation of pressures and flows [115, 134, 247]. GNN-based models has been also used to solve problems related to water quality. For example, *Li et al.*, in [132], propose GNNs for identifying contamination sources in water distribution systems, and for water quality prediction [133]. Other recent works

on GNNs for pressure estimation are [8, 92]. The main differences of those approaches with ours are the training strategy and the evaluation protocol to assess the model performance. Our proposed techniques in this regard are more realistic and give the model the ability to adapt to unexpected changes.

In this chapter, we focus on pressure estimation by leveraging both physics-based simulation models and GNN-based data-driven approaches. We rely on a data generation method that leverages the EPANET simulation tool to overcome the lack of data required for model training. However, in our approach we include all dynamic parameters (e.g., reservoir total heads, tank levels, roughness coefficient) which were not considered in previous works. This contributes to data variety and avoids that uncertainties propagate due to model simplification errors [66]. Our main contributions are twofold. First, our GNN-based estimation model is robust to unexpected sensor's location changes due to the proposed training strategy that relies on random sensor placement. Second, our evaluation protocol considers real time-dependent patterns and additionally injects the uncertainties intrinsic to real-world scenarios. The outcome is a new state-of-the-art GNN-based model, GATRes, for pressure estimation in WDNs.

GATRes is able to reconstruct the junction pressures of Oosterbeek, a large-scale WDN in the Netherlands, with an average 1.94m absolute error, which represents an 8.57% improvement with respect to other models. Similarly, our model outperformed previous approaches on other WDNs benchmark datasets. The highest improvement was seen for C-Town WDN [161] with an absolute error decrease of 52.36%, for Richmond [213] an error deacrease of 5.31%, and 40.35% error decrease for L-Town [219]. In addition, our first attempt on model generalization shows that a multi-graph pretraining followed by fine-tuning helps to increase the model performance. The absolute error on Oosterbeek network was reduced by $\approx 2\%$ following our generalization strategy.

The remainder of this chapter is as follows. Section 5.2 presents the problem statement, describes the issues that need to be addressed by pressure reconstruction models and defines the criteria to assess the model capabilities. Section 5.3 depicts the related work in the field, narrowed to GNNs for node-level regression tasks and how previous

work on GNN-based pressure estimation satisfies the criteria defined in the Section 5.2. The methodology is presented in Section 5.4, including the data generation process, a detailed description of our model architecture, and the details of the proposed approach for model training and evaluation. Section 5.5 describes the setup of the experimental phase. It includes a description of WDNs benchmark datasets used in this chapter, the base model configurations, and the evaluation metrics. Section 5.6 describes all the empirical evaluations of our approach. First, the experiments on the main use case of this study, Oosterbeek WDN, are depicted. Then, the experiments towards model generalization are shown. Next, the performance of the proposed model on different benchmark WDNs is presented. This section concludes with an ablation study to identify the contribution of the different components of the model architecture. A discussion of the most salient findings are presented in Section 5.7. Finally, the conclusions are presented in Section 5.8.

5.2 Pressure estimation in water distribution networks

5.2.1 Problem statement

Hydraulic experts have managed WDNs using essential measurements such as flow, demand, and pressure. These measurements offer a comprehensive perspective of a WDN, forming a foundation for various supervisory tasks like forecasting [108], leak detection [77], and operational control [155]. For this reason, this study focuses on addressing the fundamental challenge of approximating measurements across all nodal locations within water networks. In this investigation, we initially assume that the prior knowledge (i.e., historical data) is unavailable, and network states are discretely recorded under typical conditions, disregarding unforeseen events like leaks, earthquakes, or fires. Additionally, we presume that measured states between two neighborhoods within the network exhibit a degree of similarity. These assumptions are crucial for employing a data-driven machine learning model trained from valid cases while avoiding corruption from the complexities of water networks.

A real-life WDN consists of diverse components such as tanks, valves,

pumps, reservoirs, and thousands of customer junctions whose measurement states are crucial for management. In this study, we narrow the scope and favor pressure as the primary measurement due to the ease of meter installation and the more affordable price compared to flow ones [259]. Nevertheless, these pressure sensors are limited in practice due to infrastructural limits and privacy concerns. Consequently, the gathered data, known as the pressure states of the junction nodes, are scarce. These states play crucial roles as samples in training data-driven approaches aligned with machine-learning models that often require a large amount of data. As a prerequisite, pressure states should be fully observable to minimize the estimation loss of the trained model in all customer positions throughout the water network. This contradiction poses a challenge in applying a machine-learning approach to solve pressure estimation tasks.

The application context concerns what and when the trained model should be applied. Generally, a model is often associated with a specific water network and fixed sensors previously seen during training. Also, the training environment may exclude noisy, uncertain conditions that could affect the model's decision-making. In other words, these challenges result in worse model performance when faced with unfamiliar network topologies or uncertain situations. Consequently, model retraining is inevitable, albeit such training is an expensive and unsustainable approach. This concern enhances the necessity of the generalization ability of pressure estimation models, which has not been addressed in prior research. Before addressing this research gap, we will first delve into the specific problem within water networks and lay out the criteria necessary for a robust pressure estimation model.

5.2.2 Partially-Observable Data and Realistic Model Evaluation

The water Distribution Networks domain is characterized by partial observability due to the limited sensor coverage. This imposes an additional challenge because the reconstruction models need to be trained on fully observable network operation snapshots. The common approach to overcome this limitation is to rely on mathematical hydraulic simulation tools,

e.g. EPANET [178], to generate full views of the network operation and use them for model training [8, 92, 233, 260].

Although the hydraulic simulations solve the lack of training data for the reconstruction models, the remaining challenge is how to create a valid and reliable evaluation protocol and the data used for it. Sampling from the data generated from the simulation models and splitting them into training and test sets is not enough. Ideally, the assumption behind machine learning models is that the training data is ruled by the exact same distribution of the data on which the model will be evaluated. However, having absolute control over the data generation process and meeting such a perfect match between both distributions is unrealistic, and the assumption is violated under real-world conditions [20, 71, 101]. Thus, the prediction models should be robust to distribution shifts between training and testing samples, i.e., be able to generalize to out-of-distribution (OOD) data [72].

We observed that the data distribution of pressures in the training and test sets created by the simulation models is identical, which is unnatural in practice. The density distributions of pressures in the training and test sets from different WDNs, generated by the hydraulic simulation tool EPANET, are shown in Figure 5.1.

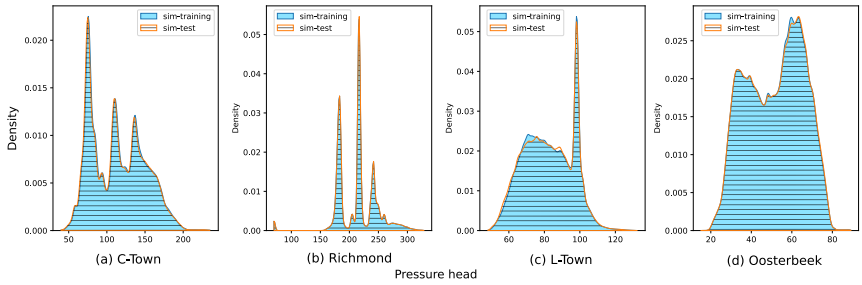


Figure 5.1: Density Distribution of training and test sets in C-Town, Richmond, L-Town and Oosterbeek WDNs, generated by the Hydraulic Simulation tool EPANET.

In this case, the simulation's dynamic parameters, e.g., reservoir total heads, junction demands, pump speed, were randomly adjusted for every run. As we described before, we consider only WDN states under normal conditions. Thus, all static parameters, e.g., node elevation, pipe length,

pipe diameter, etc., are fixed. Each simulation run produces an arbitrary state of the network (snapshot) given a particular combination of input parameter values. The algorithm ran until 50,000 stable network states were reached. Then, the data were randomly partitioned into training, validation, and test sets in a 60:20:20 ratio, respectively. The plots were generated taking 2,000 snapshots at random from the training and test sets, where each snapshot represents the pressures at all junctions. Nonetheless, as evident from the image, the distributions of the training and test sets created by the hydraulic simulations are identical in all examples. In this case, evaluating the reconstruction models on data generated by the same algorithm is simply evaluating the ability of the model to reconstruct the signals already seen during the training process.

In our work, we propose a realistic test set generation process that relies on time-based demand patterns. In addition, Gaussian noise is injected before the simulation to mimic the uncertainty intrinsic to real-world scenarios. Combining time-based demand patterns and noise injection allows us to create realistic scenarios to evaluate the ability of the models to generalize to OOD data, with visible differences in density distribution between training and test sets.

5.2.3 Criteria for model assessment

The out-of-distribution problem may originate from the uncertainty in measured sensor data, the fluctuation in hydraulic parameters, and the diversity of water network topologies. Within the context of these factors, we delve into the limitations of physical models. First, the uncertainty from sensors is due to the gradual sensor coverage plan, maintenance periods, and sensor malfunction. This uncertainty impacts the performance of physical models and often requires a human-involved recalibration process. Similarly, unforeseen fluctuations in hydraulic parameters can be triggered by changes in socioeconomic conditions, aging infrastructure, or unexpected events like pandemics. While existing techniques on top of the physical model, such as calibration and skeletonization, can mitigate these issues, they often require further human intervention. Additionally, encountering unseen water networks with different topologies inevitably leads to a redesign process that significantly increases time and resources.

Furthermore, the physical model can only perform a proper simulation if all hydraulic parameters of all components in the new network are fully identified, posing a practical challenge and emphasizing the persistence of the generalization problem.

Some previous studies have proposed utilizing more efficient data-driven machine-learning models (discussed in Section 5.3) to tackle the challenges above. However, it is critical to note that these works have often overlooked some of these problems. This oversight essentially motivates us to have a list of criteria that indicate the desirable capabilities of a data-driven model addressing the pressure estimation task on WDNs. We then propose the criteria of generalizability, adaptability, and robustness as follows.

($\mathbb{C}$1) Generalizability: The model is able to perform the pressure estimation task across diverse WDNs, regardless of their topology, even when encountering networks not seen during training. This is an important aspect of generalizability, making it more useful in practice.

($\mathbb{C}$2) Adaptability: The model should efficiently maintain its estimation capabilities in diverse contextual circumstances, typically involving the positional variation of sensor measurements. Sensor-related events like periodic maintenance, gradual coverage plans, and malfunctions can cause changes in sensor locations within the network, leading to exhaustive retraining for conventional machine-learning models in severe cases. Therefore, the criterion favoring model flexibility is essential to uphold model performance during the network's life cycle.

($\mathbb{C}$3) Robustness: This criterion considers the model's robustness against inherent distortion in real-world settings. The reasons involve noise in observations, data transmission, and parameter discrepancies between simulated and actual environments. Notably, this uncertainty has been overlooked in existing approaches, as testing data was often evaluated in a noise-free, laboratory environment.

In the following section, the provided list is crucial in assessing the state-of-the-art methods for the pressure estimation task. Accordingly, we outline their accomplishments and limitations before presenting our solution that purposely fulfills all specified criteria.

5.3 Related Work

5.3.1 GNNs for node-level regression task

Most of the GNNs have originated from a feed-forward neural network involving multiple hidden layers of interconnected neurons. Each layer performs a transformation of the input data and then passes it through a series of mathematical operations. In terms of GNNs, the favored input data is the graph. As it can naturally represent an abstract level of a water distribution network, we purposely investigated existing GNN approaches for solving our core problem, known as node-level regression.

Wu *et al.*, in [231], categorized GNNs based on their purpose into graph-level, link-level, and node-level tasks. These categories indicate the versatility and primary focus of GNNs to provide results across various domains. For instance, graph-level and link-level GNNs have been employed in domains such as chemistry [174], bioinformatics [158], and recommendation systems [43].

On the other hand, the node classification task has gained prominence in the node-level category, demonstrating its dominance in practical fields such as physics [187] and finance [234].

As a result of the prevalence of node classification, well-known GNN architectures have been developed to excel at this specific task [41, 55, 215]. This has resulted in limited attention to node regression tasks and created uncertainty about the effectiveness of these popular GNNs in handling continuous values within the node-level regime. While early research has explored node-level regression in narrow domains [62, 240], comprehensive comparisons across these popular architectures, especially within the water sector, are lacking. Addressing this issue, we delve into exploring the capability of popular GNNs in tackling a fundamental challenge in node regression: state estimation.

5.3.2 State Estimation with GNNs

Early research integrated Graph Neural Networks (GNNs) into calibration processes [247]. With this development, recent advancements have introduced calibration-free GNNs, showcasing notably superior performance compared to classical models in state estimation tasks [92]. State estimation is a process of inferring the current state of a water distribution network. For example, it involves predicting the water pressure or flow rate at different nodes based on sensor measurements within the network. Its application is crucial for subsequent management tasks, including leak localization [154], optimal control [142], and cyber-attack detection [198].

It is worth distinguishing state estimation from a related concept known as surrogate modeling, as both approaches may produce similar outputs, such as pressure, flow rate, and velocity [115]. The key distinction lies in their objectives and input features. Surrogate models strive to replicate the behavior of a physical simulation model across diverse input parameters—such as customer demand, nodal elevation, and pump head patterns. Conversely, state estimation focuses on reconstructing signals at unmeasured nodes based on existing ones and the network's topology. In other words, state estimation models have fewer requirements to provide outcomes, leading to greater convenience in the practical application. Expanding from this point, we supply a comprehensive list of representative works and assess them against the predefined criteria outlined in Section 5.2.3.

Hajgató *et al.* is the first work to propose training a GNN model on a well-defined synthetic dataset [92]. **($\mathbb{C}$2)Adaptability** is satisfied because the authors trained the model on various snapshots concerning different sensor locations. On the other hand, achieving **($\mathbb{C}$3)Robustness** is unclear as all reports were based on time-irrelevant and synthetic data. Also, the model cannot deal with the generalization problem due to the limitation of spectral-based GNNs in which the learned features extracted from Fourier space are closely associated with the corresponding network, yielding a lack of reusability [254]. For this reason, it fails to satisfy **($\mathbb{C}$1)Generalizability**.

Ashraf *et al.*, in [8], improved the above work on historical data.

In this case, working with a spatial-based GNN can help extract features from any encountered topology, so it is possible to satisfy **(ℂ1)Generalizability**. In addition, testing on noisy time-relevant data could be seen as an uncertainty consideration. However, this work heavily depends on historical data generated from a pure mathematical simulation with “unchanged" dynamic parameters (e.g., customer demand patterns). In practice, this approach does not apply to the cases where those parameters are prone to error or are unknown [123]. Hence, we consider that it weakly satisfies **(ℂ3)Robustness**. However, the authors fixed sensor positions during training, which could negatively affect the model observability of other regions in the WDN. For this reason, stacking very deep layers that increase model complexity is inevitable to ensure the information propagation from far-away neighbors to fixed sensors [13]. Additionally, retraining the model is mandatory whenever a new measurement is introduced, which has detrimental effects on its flexibility and scalability. Thus, the model violates **(ℂ2)Adaptability**.

Note that we exclude the heuristic-based methods as they do not consider the topology in decision-making. Also, several graph-related approaches [123, 233] exist in this field. However, they relied on historical data, and neither attempted to solve the task in a generalized manner. Alternatively, we assume that prior knowledge (i.e., historical data) is unavailable. In this chapter, we delve into the capability of GNNs in a general case, in which the trained model can be applied to any WDN and any typical scenario.

5.4 Methodology

5.4.1 Water network as graph

A water distribution network is a complex infrastructure that provides safe and reliable access to clean water for individual use. We define an immediate state of measurements recorded in a water network as a *snapshot* at a particular timestamp. A sequence of snapshots yields a scenario expressing a form of temporal correlation across *snapshots*. Due to initial assumptions, this temporal information is unavailable, leading us to consider a sampled snapshot as a representation of the entire scene

for a particular network.

Mathematically, a *snapshot* is represented as a finite, homogeneous, and undirected graph $G = (\mathbf{X}, \mathbf{E}, \mathbf{A})$ that has N nodes and M edges. Edges represent pipes, valves, and pumps, while nodes can be junctions, reservoirs, and tanks. The nodal features are stored in the matrix $\mathbf{X} \in \mathbb{R}^{N \times d_{node}}$, where d_{node} is the nodal feature dimension. In this work, pressure is the unique node feature because it is recognized as the most vital stable factor in monitoring the WDN [46] and aligns with prior research [8, 92]. Accordingly, we refer $\mathbf{X}$ to a pressure matrix, and the feature dimension d_{node} is fixed to 1.

$\mathbf{E} \in \mathbb{R}^{M \times d_{edge}}$ is an edge feature matrix, in which d_{edge} is the edge dimension. Depending on a particular model, we set d_{edge} to 0 if none of these edge attributes is used or 2, which indicates pipe lengths and diameters are supported. The node connection is represented in an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where $a_{ij} = 1$ means node i and j are connected by a link, and $a_{ij} = 0$ otherwise.

Observing accurate pressure $\mathbf{X}$ for an entire water network is challenging due to partial observability. Hence, we rely on a physics-based simulation model to construct synthetic pressure as training samples for the model. Concretely, the simulation encompasses a variety of parameters, both static and dynamic. Static parameters, such as nodal elevation and pipe diameter, remain constant, while dynamic parameters, such as junction demands and tank settings, fluctuate over time. Then, it solves a hydraulic equation to estimate the pressure and flow at unknown nodes in a Water Distribution Network [188]. For more details on solving hydraulic optimization, we refer the reader to the EPANET engine, which serves as our default mathematical simulation [178].

Despite the usability of conventional simulations, they demand a manual calibration process to stay synchronized with the actual physical water network. Also, they suffer from the OOD problem mentioned in Section 5.2. In light of these limitations, we adopt a strategic alternative. In particular, we merely leverage a simulation model to generate synthetic samples for training our calibration-free model. The trained model can infer the pressure of any water network in the deployment.

5.4.2 Dataset creation

Throughout this chapter, GNNs take WDN *snapshots* as inputs. In particular, each *snapshot* provides a global view of a WDN graph representing pressure values at an arbitrary time. Additionally, it contains topological information (e.g., node connectivity and edge attributes) from a corresponding water network. We denote a *clean snapshot* the one that describes an instantaneous pressure state without any hidden information. In contrast, a *masked snapshot* portrays a partially observable network in which the target feature known as pressure is mostly undetermined except for a small number of metered areas.

The conventional generation requires temporal patterns to create a set of *clean snapshots*. A pattern records a time series of a specific simulation parameter, such as customer demand or pump curve, in a fixed period. In other words, such a series is often recorded in ordinary scenarios. However, it is not guaranteed that these patterns include all events, and real-world data is highly volatile. For example, a model trained on data created from past patterns can fail to estimate the pressure of a WDN during the long-term COVID-19 pandemic due to the unexpected sudden change in water consumption that was never found in such patterns [32, 205]. Furthermore, the number of available patterns is seldom provided or partially accessible due to privacy-related concerns, especially in public benchmark WDNs. For this reason, they are often repetitively overused in modeling large-scale water networks where the number of nodes is exponential compared to the required patterns. This significantly impacts dataset diversity and, therefore, limits the model's capability to satisfy criteria **($\mathbb{C}$3) Robustness**.

We then scrutinize existing generation approaches that consider the characteristics of hydraulic parameter volatility and variability in Table 5.1. The underlying simulation (i.e., EPANET [178]) still plays a crucial role in creating *clean snapshots* given an arbitrary set of parameters. Still, each approach has a specific selection and adjustment of dynamic parameters with respect to a design space. It is worth noting that we recognize the presence of other methods that intentionally distort topology or create unforeseen scenarios (such as leakage, earthquakes, etc.) [147]. However, such approaches contradict our initial assumptions, leading to

their dismissal.

Table 5.1: The selection of dynamic parameters between conventional simulations and sampling-based generations. Parameters marked with a check can exhibit varying values, whereas others remain constant throughout the generation process. Note that the dynamic parameter selection also depends on component availability in a particular network and dataset creation stability to prevent abnormal results.

Dataset Creation	Reservoir Total Heads	Junction Demand	Pump Speed	Pump Status	Tank Levels	Valve Settings	Valve Status	Pipe Roughness
[178]	✓	✓	✓[a]					
[93]		✓	✓	✓	✓			
Our	✓	✓	✓	✓	✓	✓	✓	✓

[a] *Pump speed* is implicitly adjusted by a *pump curve* pattern.

The conventional simulation method, EPANET [178], operates in a time-dependent manner and relies primarily on fixed patterns. Excessive use of these patterns results in temporal correlations among snapshots, primarily due to their inherent seasonal factors. This issue becomes inevitable, especially in large-scale networks, where numerous unmeasured nodes require pattern assignments to complete a simulation process. Consequently, this leads to information leakage between snapshots within the same scenario (see after-splitting data distribution in training and testing sets in Figure 5.1).

Alternatively, Hajgató *et al.*, in [93], eliminate time patterns and consider a single snapshot as an instantaneous scenario. This way is more suitable to provide more observations for data-hungry models. However, their work focuses only on pump optimization, so half of the listed parameters remain untouched.

Both available generations assume the remaining parameters are deterministic and unchanged. Nevertheless, these parameters (e.g., pipe roughness) can be critical factors affecting the simulation result [247]. Thus, neglecting any of these parameters can restrict the model from learning representations of WDN snapshots.

Intuitively, we consider a comprehensive modification of all dynamic parameters as data augmentation to ensure the simulation quality and

address the generalization problem. Our main objective is to design a sufficient search space to provide different pressure views from flexible sensor positions. This approach helps alleviate the data-hungry issue when training deep learning models and benefits model robustness thanks to the augmented data space [53].

In particular, we adopt a brute-force approach to explore the full range of available dynamic parameters. To ensure the simulation quality, we exclude parameter sets that generate pressure ranges surpassing practical limits within the range of $[0\,m,\ 151\,m]$ [162]. Subsequently, our generation takes these sets of dynamic parameters, an unchanged static set, and the topology of a particular water network to generate a single snapshot using the conventional simulation (refer to Figure 5.2). Note that it only performs a single simulation step that removes the essence of temporal patterns. The outcome of this process is a set of distinct immediate pressure states, which are more versatile and independent in time. In contrast to the classical usage, this approach eliminates the temporal correlation concern and leverages all dynamic

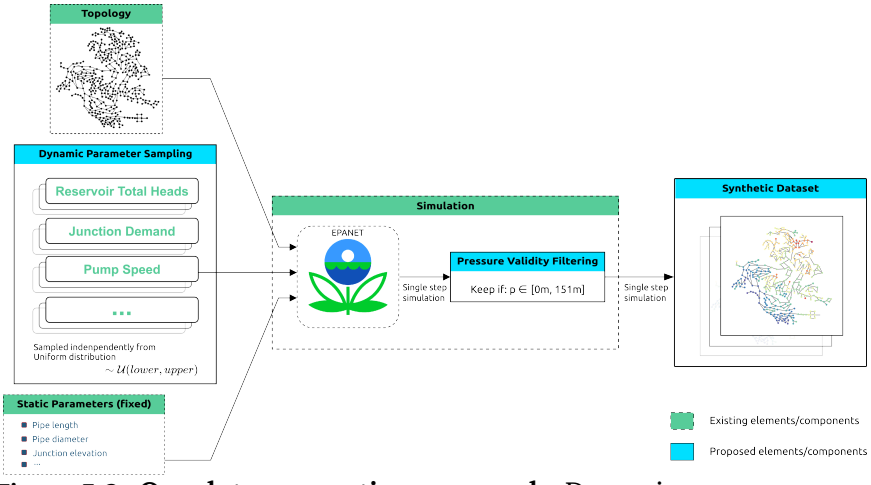


Figure 5.2: Our data generation approach. Dynamic parameters are sampled from a uniform distribution and passed to the mathematical simulation with static values and WDN topology. The result is a synthetic dataset containing legit snapshots whose pressure range should be close to reality.

parameters to generate training samples designed to cover the entire input space.

5.4.3 Model Architecture

The model is expected to learn a graph representation from known signals to estimate unknown pressures. In addition, to accurately estimate the pressure at a distant location from sensors, it is crucial to have an approach to gather the (inferred) measurements from the metered nodes and intermediate nodes to the distant neighbor efficiently. We first recap Message Passing Neural Networks (MPNN) [82], the generic framework for spatial GNNs. Then, we discuss Graph Attention Network (GAT) [215] as one of our fundamental components. In light of this, we propose *GATRes* as a principal block and devise the overall architecture illustrated in Figure 5.3.

5.4.4 Preliminaries

Considering a GNN as a series of stacked layers, MPNN describes a specific layer to transform previous representations into subsequent ones using message propagation. We omit the layer index for simplicity and denote representations of a target node i as x_i. Notably, the first representations are input features known as pressure values. Then, the output representations of a subsequent layer are computed as follows:

$$z_i = \textit{UPDATE}\left(x_i, \bigoplus_{j \in \mathcal{N}(i)} \textit{MSG}(x_j)\right) \tag{5.1}$$

where z_i is the corresponding output of the target node i, $\mathcal{N}(i)$ denotes the 1-hop neighbors, *MSG* and *UPDATE* are differentiable functions describing messages received from neighbors and the way to update that information concerning its previous representations, respectively. $\bigoplus$ is a differentiable, permutation-invariant function, ensuring the gradient flow backward for model optimization and addressing concerns related to node ordering [82]. This function plays a critical role in aggregating neighbor messages into the target node.

Depending on the task-specific purpose, numerous ways exist to define the message aggregator, such as mean, max, sum [236], or Multilayer Perceptron [248]. Ideally, $\bigoplus$ is designed to propagate messages from surrounding nodes in a sparse fashion, which only applies to non-zero values. Thus, this scheme efficiently scales when dealing with enormous graphs and reduces memory consumption.

Next, we explain GAT in view of a target node i. Concretely, GAT focuses on the intermediate representation relationship between the target node i and one of its 1-hop neighbors j. If a node pairs with itself, it forms a self-attention relationship. Hence, we strategically establish a virtual self-loop link in every node to assign weights to its own representations compared to the aggregated ones from the neighborhood. Mathematically, we can rewrite the GAT formula according to Equation 5.1 as:

$$z_i = \Big\Vert_{h}^{H} \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^{h} \mathbf{\Theta} x_j = GAT(x_i) \tag{5.2}$$

where H is the number of attention heads, $||$ is a concatenation operator, $\mathbf{\Theta} \in \mathbb{R}^{d_{in} \times d_{out}}$ is the layer weight matrix with d_{in} and d_{out}, which are the input and output representation dimensions, respectively. Specifically, a summation is chosen as the *UPDATE* function to aggregate nodal messages. For each node, its *MSG* function is computed by multiplying α with the linear combination of learnable weights $\mathbf{\Theta}$ and nodal input x_j. In addition, α is referred to as the attention coefficient and is defined as:

$$\alpha_{ij} = \textit{softmax}(\sigma(a^T[\mathbf{\Theta} x_i || \mathbf{\Theta} x_j])) \tag{5.3}$$

where $\textit{softmax}(x) = \frac{e^{x_i}}{\sum_{j \in \mathcal{N}(i) \cup \{i\}} e^{x_j}}$ is used to compute the importance score between the target node i and a neighbor j. Before calculating *softmax*, the concatenation of both nodal representations is parameterized by learnable weights $a \in \mathbb{R}^{2d_{out}}$ and passed through a non-linear activation function $\sigma(.)$ (e.g., *ReLU*, *GELU*, or *LeakyRELU* [235]).

Inspired by Vaswani *et al.* [214], GAT leverages multiple attention heads to perform parallel computation and produce diverse linear views. In the original work, those head views should be joined to "merge" all-in-one representations, thanks to a linear layer mapping the concatenated

heads. However, the conventional approach is to stack numerous concatenated GAT layers sequentially (hence, without any head joint) except for the last layer, where a final mean view is computed but only for the final logit in a classification task [215]. The postponed head joining could preserve irrelevant views in the consecutive layer that double the detrimental effect of the nodal feature sparsity due to the high masking rate in an unsupervised setting. In other words, irrelevant head views quickly saturate the impact of final nodal representations and accelerate the smoothing process (that is, oversmoothing [37]). Extra propagation layers are ineffective because they worsen the situation. Thus, we hypothesize that merging head views could complete the original design and suppress unrelated information. Intuitively, it raises a question of whether to linearly transform the concatenated head view as Vaswani *et al.* in [214] or merely take an average of head representations.

5.4.5 GATRes

Alternatively, we propose using an additional GAT layer to refine head views generated from the previous one. We name this approach the **GAT** with **Res**idual Connections (GATRes). Mathematically, we define our *GATRes* as follows:

$$z_i = x_i + \frac{1}{|\mathcal{N}(i)+1|} \sum_{j \in \mathcal{N}(i) \cup \{i\}} GAT(GAT(x_j; \alpha, \mathbf{\Theta}); \beta, \Psi) \qquad (5.4)$$

where attention coefficients $\alpha \in \mathbb{R}^{N \times H}$ computed in Equation 5.3 and learnable weight matrix $\mathbf{\Theta} \in \mathbb{R}^{d_{in} \times H d_{out}}$ belong to the first GAT. Identically, $\beta \in \mathbb{R}^{\mathbb{N}}$ and $\mathbf{\Psi} \in \mathbb{R}^{H d_{out} \times d_{out}}$ are from the second GAT but differ in shape.

As shown in the middle image in Figure 5.3, we feed the intermediate input x_i to two GAT layers sequentially. For a target node i, the inner $GAT : \mathbb{R}^{N \times d_{in}} \rightarrow \mathbb{R}^{N \times H d_{out}}$ produces multi-head views and weighs them among its 1-hop neighbors. In other words, it additionally enriches the diversity of multi-head views using the message aggregation from surrounding nodes.

Then, the outer $GAT : \mathbb{R}^{N \times H d_{out}} \rightarrow \mathbb{R}^{N \times d_{out}}$ creates a bottleneck in the feature dimension and, again, reweights the target representation

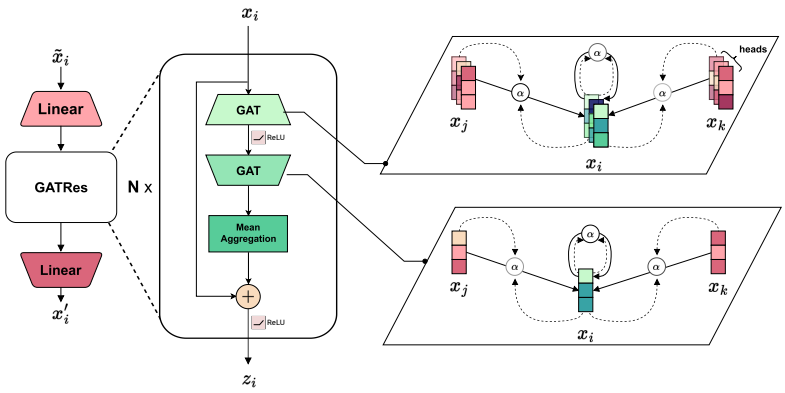


Figure 5.3: ***GATRes* architecture.** The left image indicates the overall architecture consists of two linear layers interleaving with *GATRes* blocks. The middle figure illustrates the abstract view in each block. The right-side ones explain the message aggregation mechanism between neighbor nodes.

considering those of its neighbors. Note that the second GAT has exactly one head to transform all previous heads into a consistent view. We call this process the *squeezing technique*. Since most initial features are noise or zeros, squeezing can reduce the duplicate sparsity in feature space caused by head concatenation and, therefore, benefits second attention mechanism among nodal pairs. Moreover, we consider the distribution of pressure values in the neighborhood, so we empirically apply a mean aggregator to the current representations [236]. Afterward, we use a non-parametric residual connection with the intermediate input x_i that allows depth extension and mitigates overfitting [99].

As shown in Figure 5.3, the overall structure is a stack of numerous *GATRes* blocks. As each block considers 1-hop neighborhoods, stacking multiple blocks allows message propagation to distant neighbors in the graph. Before message propagation layers, we employ a shared-weight linear transformation to project the masked input nodal features to a higher-dimensional space. The details of masked inputs will be explained in the following subsection. We refer to the first linear layer as the stem layer, which is well-known in computer vision tasks [65, 196]. After message propagation, the final linear layer acts as a decoder to project

higher-dimensional representations back to the original dimension (i.e., $d_{node} = 1$). The end-to-end model will then output an immediate snapshot in which all pressure values at all junctions are recovered.

5.4.6 Model training

This section introduces our solution to the pressure estimation task. We begin by outlining a general training scheme applicable to any GNN model. Figure 5.4 depicts this scheme, which leverages the synthetic datasets generated in the previous stage (Section 5.4.2) for training. Following this, we detail an approach to evaluate the trained model using time-relevant data.

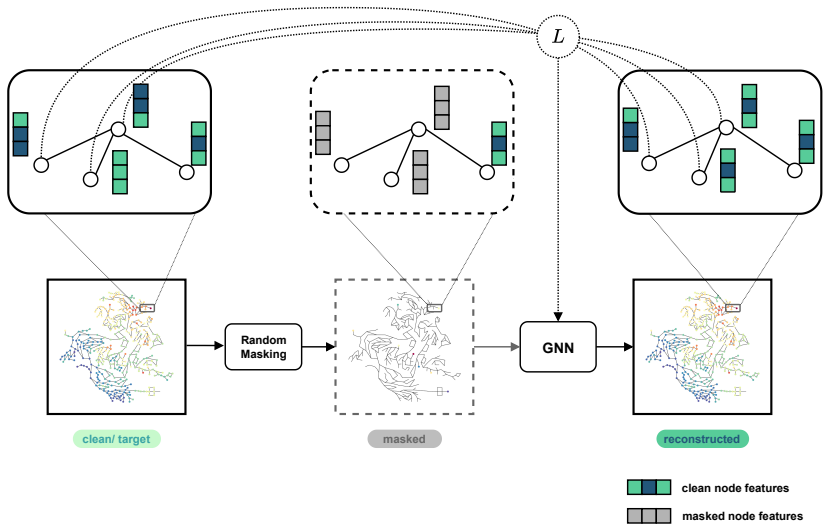


Figure 5.4: **Graph Neural Network training scheme.** Given clean snapshots, we mask out a significant number (95%) of node features. The remaining data is then sent into a GNN playing as an autoencoder to rebuild missing values with regard to graph properties (such as topology and edge attributes). GNN weights are updated using the loss derived by the predicted and ground truth values at masked places.

5.4.7 Training details

To begin with, we sample a binary mask vector $m = \{m_1, m_2, ..., m_N\}$ where each $m_i \in \{0, 1\}$. We then construct a feature subset of the masked node $\tilde{\mathbf{X}} \subset \mathbf{X}$ in which its element $\tilde{x}_i$ is denoted as:

$$\tilde{x}_i = \begin{cases} 0 & m_i = 1 \\ x_i & m_i = 0 \end{cases} \tag{5.5}$$

There are various masking strategies, such as learnable [MASK] tokens, feature permutation, arbitrary vector substitution, and mixup of nodal features, which could be helpful for future work [250]. In this work, we opt for a simple masking approach: replacing the node features with zeros in the masked positions to create the masked $\tilde{\mathbf{X}}$ (Equation 5.5). We then formalize the pressure estimation as follows:

$$\mathbf{X}' = f_{GNN}(\tilde{\mathbf{X}}, \mathbf{E}, \mathbf{A}; \mathbf{\Theta}) \tag{5.6}$$

where $f_{GNN} : \mathbb{R}^{N \times d_{node}} \times \mathbb{R}^{N \times N} \times \mathbb{R}^{M \times d_{edge}} \mapsto \mathbb{R}^{N \times d_{node}}$ is a generic GNN function that takes partially observable feature matrix $\tilde{\mathbf{X}}$, the topology $\mathbf{A}$, and edge attributes $\mathbf{E}$ as inputs and yields reconstructed features $\mathbf{X}'$ characterized by model weights $\mathbf{\Theta}$. The key idea is to find the optimal weights that minimize the error between predicted and ground truth nodal features. Mathematically, the objective is formalized as follows:

$$\mathbf{\Theta}^* = \underset{\mathbf{\Theta}}{\operatorname{argmin}} \, \mathbf{L}(\mathbf{X}', \mathbf{X}) \tag{5.7}$$

Empirically, we use the mean square error (MSE) as a default loss function $\mathbf{L}$ for *GATRes* because it yields the best result in our tests. In addition, inspired by BERT [63], the loss is computed on masked positions.

After computation, model weights $\mathbf{\Theta}$ are updated by the partial derivatives with respect to the computed loss. For details, we refer to gradient descent optimization techniques [117, 179]. The training process is repeated with different masked features $\tilde{\mathbf{X}}$ derived from the original features $\mathbf{X}$ until the model converges.

5.4.8 Testing details

In testing, the test graphs can be represented as $\mathbf{G}_{test} = (\tilde{\mathbf{X}}_{test}, \mathbf{E}_{test}, \mathbf{A}_{test})$. By default, topology and edge attributes are retained as in training, while $\tilde{\mathbf{X}}_{test}$ is varied and its data distribution is undetermined. For the generalization problem, $\mathbf{G}_{test}$ can differ from the training graphs in any property. In other words, the model should be able to estimate pressure on unseen topologies and an unknown data distribution.

When the temporal dimension is involved, the test graph at a particular time t is denoted as $\mathbf{G}^t_{test}$. As the designed model takes only one snapshot as input, we feed temporal $\mathbf{G}^t_{test}$ into the trained GNN model sequentially and individually. In other words, previously inferred outcomes do not impact any reconstruction result within a testing scenario. These scenarios are considered real-world situations associated with inherent uncertainty due to dynamic factors [260]. To reflect this uncertainty in our testing, we adopt a distortion method applied to junction demands during the testing phase, drawing inspiration from [154, 260]. Specifically, we outline two strategies as follows:

Clean test. We assume no uncertainty, ensuring that baseline models observe clear, precisely calibrated pressure. As these models treat each snapshot as an independent sample, we gauge the model adaptability across different sensor configurations using a distinct mask for each snapshot in every trial. The statistical outcomes from 100 trials, encompassing diverse metered locations, illustrate the model performance under typical conditions.

Noisy test. Following [260], we inject Gaussian noise into junction demands before processing the simulation and then pair each outcome snapshot with a random mask for a test case. Pipe roughness, another considered parameter, is intentionally excluded due to its inconsistency across different headloss formulas used for each water network. Notably, the Hazen-Williams and Darcy-Weisbach formulas are sensitive to pipe roughness, whereas the Chezy-Manning formula remains unaffected by this parameter [120]. In the interest of generalization, we have opted to exclude this parameter.

Nevertheless, we apply a stronger noise level to junction demands beyond the original tests. The new noisy test uses a mean of 10% and a standard deviation of 100% of the initial demands. We run 100 test cases and report statistical findings.

5.5 Experiment settings

5.5.1 Datasets

The main use case in this study was conducted on a private large-scale WDN in the Oosterbeek area of the Netherlands. The network comprises 5855 junctions and 6188 pipes. Figure 5.5(a) shows the topology and the pressures at the nodes from a random snapshot of the Oosterbeek WDN.

We also used four publicly available WDN benchmarks, namely Anytown [221], C-Town [161], L-Town [219], and Richmond [213], to provide a baseline for evaluation and reproducibility of our work . Finally, in the experiments related to model generalization, we used two additional public datasets, Ky13 [102] and an anonymized WDN called "Large" [190]. The WDNs used in this study vary in size and structure, ranging from small- and medium-sized to large-scale networks like "Large" and Oosterbeek, as shown in Figure 5.5. Table 5.2 shows the main characteristics of each network.

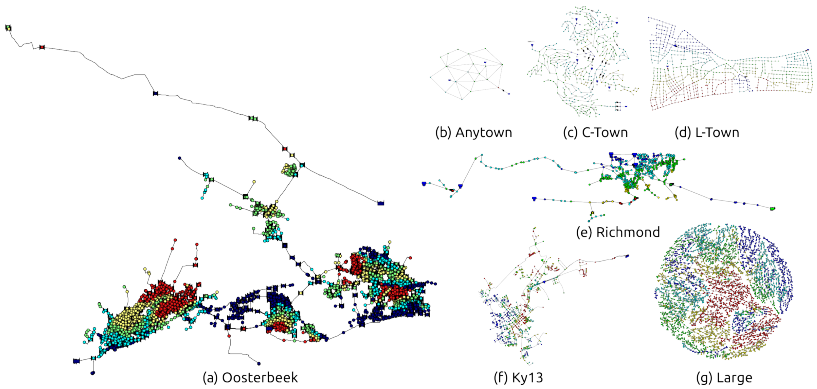


Figure 5.5: Water Distribution Networks used in this study.

Table 5.2: Properties of WDNs used in this study.

WDNs:	Oosterbeek	Anytown	C-Town	L-Town	Richmond	Ky13	Large
junctions	5855	22	388	785	865	775	3557
pipes	6188	41	429	909	949	915	4021

5.5.2 Baseline models settings

Generally, two goals dictate the baseline selection. Section 5.3 mentions the first goal: to evaluate popular GNN architectures on a node-level regression problem. The aim is to achieve acceptable errors when the models are tested on data points that are from an unknown "nature" distribution. The second goal is to explore existing frameworks, from synthesized data querying and model training to evaluation phases. We aim to establish a reliable benchmarking framework for pressure estimation tasks, or problems related to water distribution networks. In other words, the model that performs better in our tests should be more useful in practical applications.

For this purpose, we compare our *GATRes* architecture to popular GNNs, including *GCNii* [41] and *GAT* [215]. In addition, *GraphConWat* (GCW) [92] and *mGCN* [8], which are leading approaches to solving pressure estimation using GNNs with sparse information, are also considered in our comparison. Table 5.3 summarizes the model settings.

Table 5.3: Baseline settings.

	GCNii	GAT	GCW	GCW tuned (ours)	mGCN	GATRes small (ours)	GATRes large (ours)
#blocks	64	10	4	4	45	15	25
#hidden.channels	32	{32,64}	{120,60,30}	32	{98,196}	{32,64}	{128,256}
Coefficient K	-	-	{240,120,20,1}	{24,12,10,1}	-	-	-
Edge Attribute	binary	binary	binary	binary	pipe.len, pipe.dia[a]	binary	binary
Norm Type	znorm	znorm	minmax	znorm	minmax	znorm	znorm
Loss	MSE	MSE	MSE	MSE	MAE	MSE	MSE
Learning Rate	3e-4	3e-4	3e-4	3e-4	1e-5	5e-4	5e-4
Weight Decay	1e-6	1e-6	1e-6	1e-6	0	1e-6	1e-6

[a] Pipe lengths and pipe diameters. They are static parameters gathered from the corresponding Water Distribution Network.

GATRes-small is the optimal version after the optimization process, which will be carefully explained in a later section. To study the impact of the model size, we also introduce *GATRes-large*, which scales to a size comparable to *mGCN* in terms of the number of parameters.

GAT remained at a shallow depth to prevent the oversmoothing prob-

lem [37]. Precisely, neighbor features encoded by an excessively deep GNN converged to indistinguishable embeddings that harm the model performance. Empirically, we balance the trade-off between performance and efficiency for each model to select the appropriate hyperparameters.

In *GraphConvWat* models, we detached binary masks from the input features as these masks did not improve model performance, which aligns with the findings in [8]. Furthermore, *GraphConvWat tuned* is a lightweight version in which the degrees of the Chebyshev polynomial K_i are set to smaller values to reduce complexity, and it works surprisingly well in our experiments.

Training *mGCN* slightly diverged from its original work with respect to sensor positions. Concretely, Ashraf *et al.*, in [8], trained *mGCN* using fixed sensors with extensive historical data. However, as we explained in Section 5.4.6, this data was inaccessible throughout training. Therefore, we fed different random masks into the model in each epoch. Considering a synthetic dataset, the model had an opportunity to capture meaningful patterns in various sensor positions. Additionally, *mGCN* incorporated static edge attributes such as pipe length and diameter in its final decision-making process.

We then provide an overview of other hyperparameters, applicable to any mentioned model. Each model was trained in a fixed number of iterations, so-called epochs, throughout the entire training set. In addition, the whole dataset, including the training set, was pre-processed by z-score transformation (z-norm) or min-max feature scaling. Within an iteration, the model was sequentially fed data batches whose size determined the number of training examples. Both epochs and batch size were detailed in each specific experiment. Following each iteration, we computed the loss, typically Mean Squared Error (MSE) or MAE, comparing predicted outputs to ground truth. Subsequently, this loss guided the optimizer algorithm in updating the model's weights. The optimizer's behavior was influenced by additional hyperparameters, including the learning rate, regularization (defaulting to L2), and weight decay, which penalized model weights to mitigate overfitting. For comprehensive definitions, we recommend referring to [21].

5.5.3 Evaluation metrics

The most common evaluation metrics used for assessing the performance of regression models are Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and, Mean Absolute Percentage Error (MAPE) [62,110,256]. Following the insights from [128], the models should be evaluated using both relative and absolute error metrics. Thus, our model is evaluated using MAE and MAPE. Additionally, we included the Nash and Sutcliffe Coefficient of Efficiency (NSE), which is widely used to evaluate the performance of hydrologic models [128]. Finally, we used an accuracy metric defined as the ratio of positive predictions to the total number of predicted values. The positive predictions are those that deviates by at most a certain threshold (δ_{thresh}) from the true value. Thus, the evaluation metrics used in this chapter are defined as follows:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{5.8}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i} \tag{5.9}$$

$$\text{NSE} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2} \tag{5.10}$$

$$Acc(@\delta_{\text{thresh}}) = \frac{1}{N}\sum_{i=1}^{N} positive_i\,; \quad positive = \begin{cases} 1, & \text{if } |y_i - \hat{y}_i| \leqslant \delta_{\text{thresh}} * y_i \\ 0, & \text{otherwise} \end{cases} \tag{5.11}$$

where $\boldsymbol{y}$ denotes the true values, $\hat{\boldsymbol{y}}$ denotes the predicted values, $\overline{\boldsymbol{y}}$ is the mean of the true values, and $\boldsymbol{N}$ is the number of values to predict.

5.6 Experiments

The experimental evaluation focuses on methods that are directly comparable to the proposed approach in terms of task formulation and evaluation protocol. While the Introduction and Related Work sections review a

broad range of related studies, including physics-based simulation methods and graph-based approaches addressing different objectives, only a subset of these works target pressure estimation in water distribution networks under a supervised learning setting. Accordingly, the proposed approach is compared against the most relevant state-of-the-art GNN-based approaches for pressure estimation, namely GraphConvWat [92], and mGCN [8]. In addition, well-established GNN architectures such as GCN and GAT are included as canonical baselines to provide a broader methodological reference. This selection enables a fair and meaningful comparison while clearly positioning the proposed approach within the current state of the art.

5.6.1 Baseline comparison on Oosterbeek WDN

In this experiment, we investigated the proposed model's performance against GNN variants on a large-scale WDN benchmark called Oosterbeek. Specifically, given the topology and hydraulic parameters, our dataset generation provided 10,000 synthetic snapshots divided into 6000, 2000, and 2000 for training, validation, and testing sets, respectively. However, as we discussed in Section 5.2.2, these synthetic sets might not reflect real-world scenarios. Therefore, we merely used them to keep track of model learning during the training process.

Alternatively, we performed the comparison on the Oosterbeek dataset recorded every five minutes for 24 hours. We relied on mathematical simulation to produce reproducible results that resembled real-world conditions. The topology and predefined parameters set by hydraulic experts under a calibration process made them valid for our analysis. As a result, we considered the simulated outcomes of time-relevant data as ground truth.

Regarding the experimental setting, we trained all models for 500 epochs with a batch size of 8 for a fair comparison among the baseline models. Early Stopping was applied to terminate training if the validation error showed no improvement in 100 steps. We used the Adam optimizer [117] and set the default masking rate at 95%, leaving only 5% of nodes unmasked.

For evaluation, we tested the baseline models on the 24-hour Ooster-

beek dataset, repeating the process 100 times. The mean and standard deviation of the results are presented in Table 5.4. Unless otherwise specified, the default setting in our experiments is a clean test. The results of the noisy test are given in Table 5.5.

Table 5.4 shows that *GATRes-small* achieved accurate junction pressure reconstruction with a MAPE of 7% and a MAE of 1.93m, even with a sparse masking ratio of 95%. Notably, the testing data were time-sensitive and originated from an unfamiliar distribution that our models were not exposed to during training. The good results on snapshot-based models suggest that in cases where temporal data is not available, snapshot-based models seem to be good alternatives.

Table 5.4: Model comparison in the clean test performed on 24-hour Oosterbeek WDN at 95% masking rate.

Model	#Milion Params(↓)	MAE(↓)	MAPE(↓)	NSE(↑)	Acc(@0.1)(↑)
GCNii [41]	0.65	6.357±0.0197	0.2147±0.0008	-0.0137±0.0061	38.48±0.1351
GAT [215]	0.35	3.726±0.0120	0.1287±0.0008	0.3276±0.0037	73.52±0.0900
GraphConvWat [92]	0.92	3.067±0.0077	0.1160±0.0004	0.6938±0.0020	69.92±0.1205
GraphConvWat-tuned	**0.23**	2.293±0.0087	0.0821±0.0005	0.7518±0.0024	83.03±0.1025
mGCN [8]	2.48	2.111±0.0085	0.0806±0.0003	0.7100±0.0030	84.05±0.0693
GATRes-small (ours)	0.66	**1.937**±0.0074	**0.0703**±0.0005	0.7773±0.0025	**87.48**±0.0761
GATRes-large (ours)	1.67	2.020±0.0132	0.0711±0.0003	**0.7864**±0.0031	84.33±0.1347

Table 5.5: Model comparison in the noisy test performed on 24-hour Oosterbeek WDN at 95% masking rate.

Model	#Milion Params(↓)	MAE(↓)	MAPE(↓)	NSE(↑)	Acc(@0.1)(↑)
GCNii [41]	0.65	6.696±0.0838	0.2484±0.0552	-0.1064±0.0266	36.02±0.4684
GAT [215]	0.35	4.397±0.3052	0.2112±0.0767	0.1490±0.1153	66.98±1.6290
GraphConvWat [92]	0.92	3.611±0.1234	0.1551±0.0376	0.5877±0.0370	62.99±1.1600
GraphConvWat-tuned	**0.23**	2.347±0.0252	0.0963±0.0363	0.749±0.0086	81.09±0.3877
mGCN [8]	2.48	2.188±0.0558	0.0948±0.0155	0.6993±0.0213	82.83±0.4199
GATRes-small (ours)	0.66	**1.964**±0.0301	0.0802±0.0458	**0.778**±0.0113	**86.56**±0.2826
GATRes-large (ours)	1.67	2.115±0.0503	**0.0799**±0.0207	0.7417±0.0140	83.43±0.5044

In addition, we assessed the efficiency of baseline models in the Oosterbeek experiment using an Nvidia RTX 3060 Laptop GPU during inference only. The results are presented in Table 5.6.

Table 5.6: Throughput comparison in the clean test performed on 24-hour Oosterbeek WDN at 95% masking rate.

Model	Throughput(↑) (Snapshots per second)
GCNii [41]	663.80
GAT [215]	2320.37
GraphConvWat [92]	90.39
GraphConvWat-tuned	2026.65
mGCN [8]	44.94
GATRes-small (ours)	749.38
GATRes-large (ours)	31.21

In this evaluation, we measured throughput, which counts the number of processed snapshots per second. This metric can demonstrate the efficiency of baselines in terms of large-scale matter that demands continuous processing of massive data streams from sensors.

Notably, lightweight models such as *GAT* and *GraphConvWat-tuned* achieved the highest throughput, with our *GATRes-small* model ranking third. When we consider both efficiency and performance in Table 5.6, *GATRes-small* is a balanced option as this model delivers the best results while maintaining sufficient efficiency, a critical factor in saving computation resources and ensuring the sustainability of the environment.

Our next focus was on analyzing the robustness of the baseline models. Specifically, we assessed each baseline on clean and noisy tests using an individual snapshot with 100 randomly initialized masks. Our primary objective was to measure the model's robustness in these contrasting scenarios. A superior model should exhibit minimal error discrepancy between between clean and noisy tests. As illustrated in Figure 5.6, both versions of *GATRes* consistently maintained similar error levels even under conditions of high uncertainty. In contrast, other models exhibited a noticeable gap in their results when transitioning from clean to noisy environments.

Finally, we conducted a detailed analysis of our top-performing model, *GATRes-small*, based on the evaluations conducted earlier. In this analysis, we intentionally masked sensor locations to observe model inference

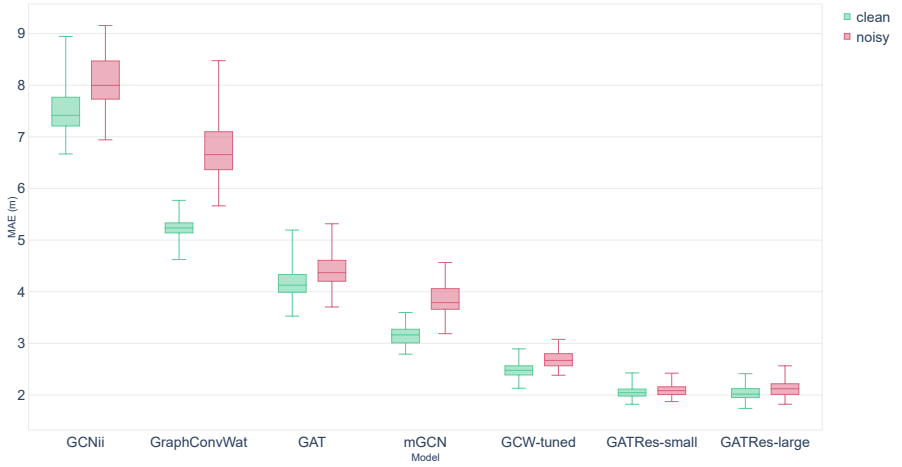


Figure 5.6: Baseline Mean Absolute Errors measured for a single snapshot under both clean and noisy conditions.

performance on those nodes. Figure 5.7 illustrates time series data from the predictions of *GATRes-small*, a well-calibrated simulation, and actual meter readings. As expected, *GATRes* closely mirrors the behavior of the hydraulic simulation. Although a slight difference exists between them, both time series were bounded within the range of actual measurements.

5.6.2 Generalization

This set of experiments is the first attempt to achieve generalization capabilities of our model. We aim to evaluate whether training a model on different topologies simultaneously can equip the model with generalization capabilities. For this, we chose three different WDNs, whose topologies vary in structure and size. We trained *GATRes-small* on L-Town, Ky13, and "Large" WDNs simultaneously. This model is named the *Multi-Graph model*. This model was trained with the *ReduceLROnPlateau* learning rate scheduler from the PyTorch library. The scheduler reduces the learning rate if the model does not improve for a certain number of epochs. The initial learning rate was set to $5e$-3 and was reduced by a factor of 0.1 if the validation loss does not improve for 30 consecutive epochs. The batch size was 16, and the model was trained for 500 epochs.

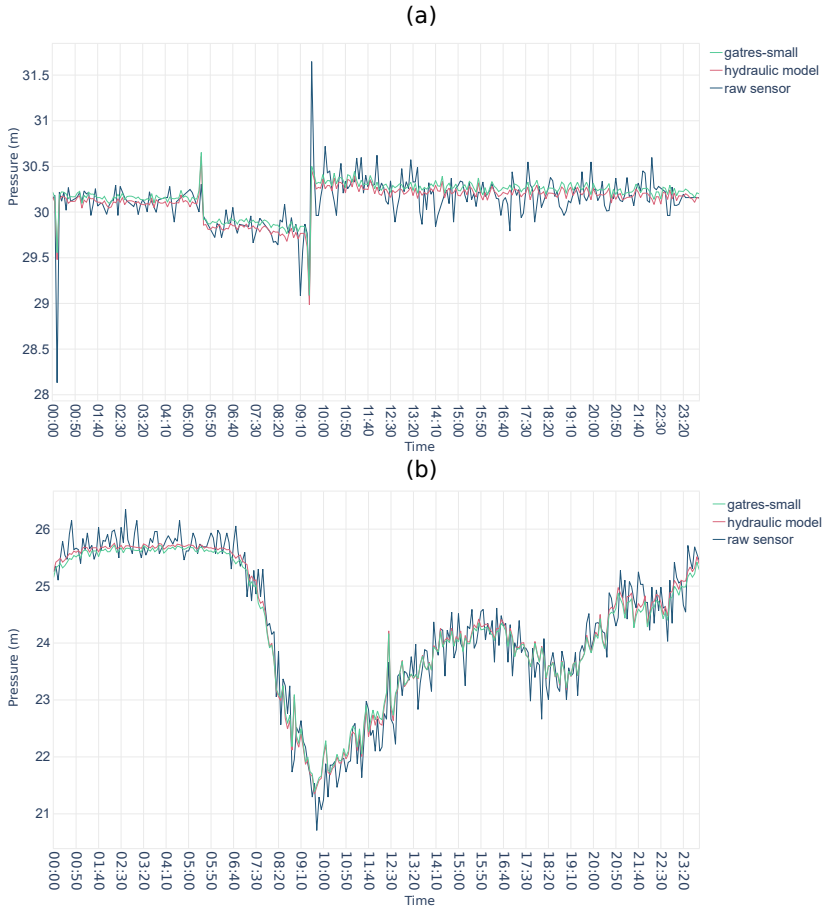


Figure 5.7: Pressure estimates derived from a hydraulic simulation model and our GATRes-small, validated against real sensor data from points (a) and (b) within the Oosterbeek WDN.

Then, to assess the generalization capabilities of the pretrained model, we executed two different experiments: zero-shot inference, and transfer learning with fine-tuning.

Zero-shot inference implies evaluating the pretrained *Multi-Graph model* on a fully unseen topology. Hence, we used the pretrained model directly to reconstruct the pressures of our use case, Oosterbeek, a WDN topology not seen during training.

The second experiment is transfer learning with fine-tuning. Transfer

learning is a technique motivated by the fact that humans use previously learned knowledge to solve new tasks faster or better [163]. Hence, the weights learned by a model trained on some particular network(s) can be transferred to train and improve (fine-tune) the prediction capabilities of a model on a new, previously unseen, WDN.

In our work, the pretrained model is the *Multi-Graph model*, trained on L-Town, Ky13, and "Large", and the fine-tuned model targets the Oosterbeek WDN. Usually, during fine-tuning, the top layers of the pretrained model are frozen and reused as feature extractors for the target data. We empirically found that unfreezing the entire model and retraining all layers produces better results. Thus, we initialized the weights of the target model with those of the pretrained one. Then, we reduced the learning rate for training the target model to avoid completely changing the pretrained weights during fine-tuning. The learning rate was reduced from $5e$-3 in the source model to $1e$-4 during fine-tuning. The target model was trained with a batch size of 8 for 200 epochs. Fine-tuning can help practitioners reduce the implementation time of a predictive model for a completely new and unseen WDN. The more WDNs are added to the Multi-Graph model, the broader and more diverse the training data become. This can reduce the amount of samples (snapshots) required for training. Analyzing the number of snapshots with respect to the number of WDNs included in the training procedure is an interesting path for future work.

The results of both experiments are shown in Table 5.7. They reflect those of Yosinski *et al.*, [242] who also found that combining transfer learning with fine-tuning yields better performance than a model trained directly on a target dataset.

Table 5.7: Generalization evaluation on 24-hour Oosterbeek WDN. Results of GATRes-small model, trained directly of Oosterbeek, are compared against those produced by the zero-shot and transfer learning with fine-tuning experiments.

5.6.3 The effect of masking ratios

To explore the model capabilities, we investigated the *GATRes-small* under myriad masking rates. As is the previous experiment, the model was trained on the synthetic dataset generated from our algorithm and was evaluated using a clean test on the 24-hour data. Both were devised from the Oosterbeek WDN. In addition, each *GATRes-small* corresponding to a specific mask rate was trained for 200 epochs with the default settings. For convenience, we replaced the model name with fixed masking rates in this experiment.

Figure 5.8 shows the influence of the masking ratio on the proposed model. Each ratio indicates a specific probability of missing nodal features (i.e., the pressure signals in a snapshot graph). Due to the sensor density being exceptionally sparse in real-world scenarios, the typical benchmark of 95%, commonly found in previous studies, fails to reflect this practical issue. Therefore, we report errors occurring in all cases with lower and more extreme ratios beyond the standard. As expected, we observe a decreasing trend in performance with increasing masking rates. This pattern also appears across other metrics in Table 5.8.

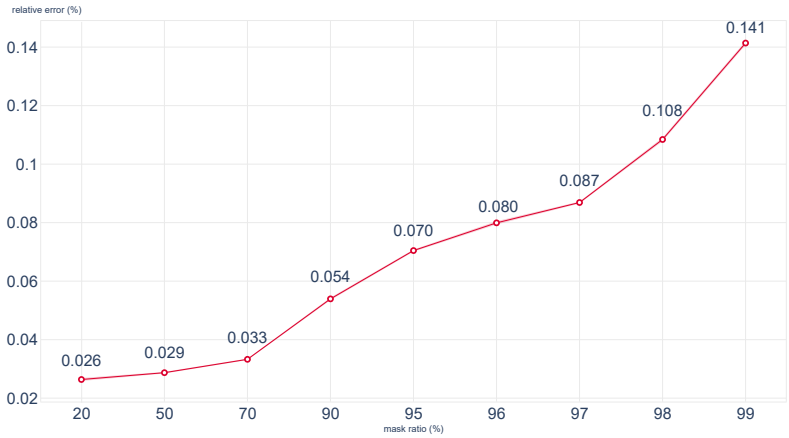


Figure 5.8: Relative errors (MAPE) for nodal pressure on different masking ratios(lower is better).

We conducted an additional investigation into the discrepancy between the masking ratios of train-test pairs. Our approach involved eval-

Table 5.8: Detailed performance of *GATRes-small* on different masking ratios.

Mask ratio(%)	MAE(↓)	MAPE(↓)	NSE(↑)	Acc(@0.1)(↑)
20	0.610	0.0265	0.9599	0.9760
50	0.798	0.0286	0.9372	0.9654
70	0.900	0.0331	0.9301	0.9586
90	1.457	0.0544	0.8603	0.9167
95	1.939	0.0703	0.7770	0.8746
96	2.213	0.0800	0.7185	0.8467
97	2.415	0.0867	0.7059	0.8148
98	3.075	0.1091	0.5648	0.7465
99	4.087	0.1414	0.3424	0.6396

uating a trained model on the 24-hour Oosterbeek with various masking rates rather than just the specific rate initially trained on. Through this exploration, we found that the best model of a specific testing masking rate was unnecessary to be trained on this rate. Table 5.9 shows this phenomenon in extreme ratios. Considering the trade-off between sparsity and performance, the model trained with a 97% masking rate demonstrated a Pareto-optimal solution. In addition, it surprisingly achieved the best results in extremely sparse testing rates (i.e., $> 98\%$). This means that at most 3% of the total nodes would be sufficient for a high-quality model to monitor the Oosterbeek WDN, a large-scale network. Further analysis is highly recommended for WDN authorities to balance the trade-off between efficiency and measurement resources.

5.6.4 Baseline comparison on benchmark WDNs

In this set of experiments, we compared the performance of *GATRes-small* against two state-of-the-art baseline models, GraphConvWat [92] and mGCN [8]. We evaluated the three models on four benchmark WDNs: Anytown, C-Town, Richmond, and L-Town, described in Section 5.5.1.

The experiments were conducted following the method proposed by Hajgató *et al.* in [93] to ensure a fair comparison. Specifically, 1,000, 10,000, and 20,000 snapshots were generated for the C-Town, L-Town, and Richmond WDNs, respectively. Then, the datasets were split into

Table 5.9: Confusion matrix of relative mean errors (MAPE) between different train and test masking ratios(lower is better). Bold and underline are used to highlight the best and second-best results for a specific test mask, respectively.

test mask(%)	train mask (%)				
	95	96	97	98	99
95	**0.0702**	0.0723	0.0725	0.0772	0.0814
96	**0.0766**	0.0797	0.0784	0.0843	0.0882
97	**0.0858**	0.0901	0.0870	0.0934	0.0970
98	0.1031	0.1077	**0.1018**	0.1090	0.1109
99	0.1454	0.1494	**0.1388**	0.1450	0.1414

training, validation, and test sets in a 6:2:2 ratio.

We used the same experimental settings as proposed in the baseline approaches to guarantee a fair comparison. Thus, in all experiments the models were trained for 2,000 epochs with early stopping, using the Adam gradient-based optimization algorithm [117]. The GraphConvWat model training was stopped if the validation loss did not improve for 50 consecutive epochs. In the case of mGCN, the training was stopped if no improvement was seen after 250 epochs. Likewise mGCN, our model training was stopped after 250 epochs if no improvement was observed. In all cases, it is considered an improvement when the validation loss decreases by at least *1e-6*.

The evaluation of the models' performance on each WDN, with the exception of Anytown, was performed using data that included realistic demand patterns per node. In the case of C-Town and Richmond, the WDN snapshots for evaluation were created using a 24-hour demand pattern time series sampled at 5-minute interval. L-Town evaluation snapshots were created using a 1-week demand pattern time series sampled at 5-minute interval. Table 5.10 shows the results of the performance comparison of ten runs per WDN, and the mean and standard deviation are reported. In order to make the test deterministic, we hard-coded 10 different seeds, one for each run. Then, the same seed was used per run across all three models to ensure a fair comparison of the results. As

Table 5.10: Models performance comparison on 24-hour demand pattern time series data.

WDN	Metrics	Models		
		GraphConvWat	mGCN	GATRes-small (ours)
C-Town	MAE (↓)	14.8860 ±0.1418	19.9138 ±0.0948	**9.4860** ±0.1822
	MAPE (↓)	0.1028 ±0.0009	0.1318 ±0.0005	**0.0690** ±0.0010
	NSE (↑)	0.7870 ±0.0046	0.6310 ±0.0030	**0.8480** ±0.0075
Richmond	MAE (↓)	4.3501 ±0.0170	2.9690 ±0.0283	**2.8114** ±0.0899
	MAPE (↓)	0.0196 ±0.0001	**0.0128** ±0.0002	0.0133 ±0.0005
	NSE (↑)	0.9500 ±0.0000	**0.9630** ±0.0046	0.9390 ±0.0030
L-Town	MAE (↓)	3.4505 ±0.0129	1.5928 ±0.0050	**0.9501** ±0.0086
	MAPE (↓)	0.0611 ±0.0002	0.0305 ±0.0001	**0.0157** ±0.0002
	NSE (↑)	0.5040 ±0.0049	0.8000 ±0.0000	**0.9000** ±0.0000

can be seen from the table, our model *GATRes-small* achieved the lowest MAE in all WDNs and the lowest MAPE in all networks except Richmond. Likewise, *GATRes-small* achieved the highest NSE in all WDNs except Richmond.

One limitation of previous approaches is the evaluation of model performance on unrealistic data, i.e., an exact copy of the training data distribution (Section 5.2.2). In previous approaches, the snapshots representing random WDN states used for training, validation, and test were created by the same algorithm. Consequently, the distribution of the data used for testing is a fidelity copy of the data used for training. However, in practice, the distribution of the real data differs from the data used for training the reconstruction models, as explained in Section 5.2.2. Therefore, it is important that the models adapt to handle such uncertainties. Previous approaches achieve impressive performance when tested on replicas of the training data (see Table 5.11), but a performance drop is evident when they are evaluated on a realistic scenario (see Table 5.10).

The density distributions of training and test sets in C-Town, Richmond and, L-Town WDNs are shown in Figure 5.9. It is clear that the distributions of the training and testing data created by the same algorithm (mathematical simulation) are identical, while the distribution of the test data with a demand pattern differs substantially from the

Table 5.11: Models performance comparison on synthetic sampling-based snapshots following the approach presented by Hajgató *et al.* [93].

WDN	Metrics	Models		
		GraphConvWat	mGCN	GATRes-small (ours)
Anytown	MAE (↓)	5.1044 ±0.0714	3.9460 ±0.0642	**3.9245** ±0.1056
	MAPE (↓)	0.0654 ±0.0012	0.0497 ±0.0009	**0.0491** ±0.0012
	NSE (↑)	0.7440 ±0.0049	**0.8020** ±0.0075	0.7980 ±0.0189
C-Town	MAE (↓)	4.1619 ±0.0170	**1.6963** ±0.0133	1.8928 ±0.0149
	MAPE (↓)	0.0354 ±0.0001	**0.0148** ±0.0001	0.0169 ±0.0001
	NSE (↑)	0.9640 ±0.0049	**0.9900** ±0.0000	**0.9900** ±0.0000
Richmond	MAE (↓)	2.3999 ±0.0069	**0.6363** ±0.0061	1.5979 ±0.0106
	MAPE (↓)	0.0110 ±0.0000	**0.0029** ±0.0000	0.0080 ±0.0001
	NSE (↑)	0.9805 ±0.0003	**0.9900** ±0.0000	0.9750 ±0.0009
L-Town	MAE (↓)	1.2970 ±0.0036	**0.2441** ±0.0014	0.4930 ±0.0028
	MAPE (↓)	0.0159 ±0.0000	**0.0030** ±0.0000	0.0061 ±0.0000
	NSE (↑)	0.9700 ±0.0000	**1.0000** ±0.0000	**1.0000** ±0.0000

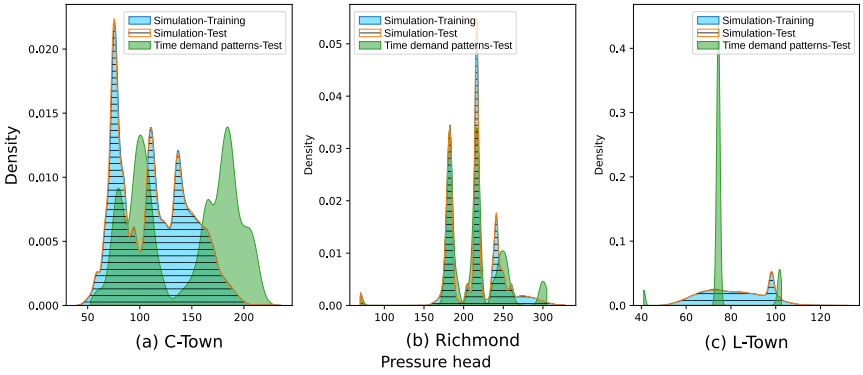


Figure 5.9: Comparison of density distributions of synthetic and time-based datasets in C-Town, Richmond and L-Town WDNs. The simulation-training and simulation-test curves represent the density of pressure heads generated without including demand patterns. In contrast, the time demand patterns-test represent the data created using the demand-patterns time series.

distribution used during training. This shows the ability of *GATRes* to adapt to the changes that occur in real-world scenarios. It also shows that other models achieve better results only when evaluated on fidelity copies of the training data, a consequence of overfitting caused by the large model complexity of previous approaches. The density distribution plot in Figure 5.9(b) explains the good performance of mGCN on Richmond WDN in terms of MAPE and NSE (Table 5.10), as the time-based demand pattern test dataset has a similar distribution to the one used for training.

5.6.5 Ablation study

The ablation study presented in this section evaluates the importance of the different components of the *GATRes* model architecture, and the effect of their removal or alteration on performance. In each run, a specific component is removed or altered, and *GATRes* is restored to its original version before a new change is made. The different variants used in the ablation study are as follows:

Without Residual Connections (woResCon). The residual connections used within each *GATRes* Block are removed.

Without Mean Aggregation (woMeanAggr). The mean aggregation applied after the second convolution within each *GATRes* Block is removed, and the residual connection is added to the output of the second convolution.

Without Residual Connection and Without Mean Aggregation (woResCon-woAggr). Both the residual connection and the mean aggregation are removed from the *GATRes* Block.

Mean Aggregation Outside the Block (MeanAggrOut). Instead of applying a mean aggregation within each block, it is applied only once in the forward pass, after the output of the last *GATRes* Block and before the final linear layer.

These experiments were performed by training *GATRes-small* on the C-Town WDN. As can be seen in Table 5.12, removing the Residual

Connections produced the highest negative impact on model performance for all metrics.

Table 5.12: Ablation study of *GATRes* evaluated on C-Town 24-hour time series data.

Variants	MAE (↓)	MAPE (↓)	NSE (↑)
GATRes-small	**09.4860** ±0.1822	0.0690 ±0.001	**0.8480** ±0.0075
woMeanAggr	09.7479 ±0.1444	**0.0686** ±0.0009	0.8473 ±0.0046
woResCon-woAggr	10.0934 ±0.1644	0.0735 ±0.0010	0.8150 ±0.0062
MeanAggrOut	10.3735 ±0.1251	0.0735 ±0.0006	0.8333 ±0.0058
woResCon	11.5362 ±0.1697	0.0815 ±0.0008	0.7694 ±0.0089

5.7 Discussion

In this section, we discuss our findings and technical changes that affect our model in estimating pressures in Water Distribution Networks. We first review changes that made *GATRes* versions outperform other baselines and discuss their limitations. Then, we discuss the role of synthetic data and the relationship between hydraulic simulation and data-driven models. Finally, we address the question of generalizability in the context of our research.

5.7.1 General findings and limitations

We first remark that our *GATRes* satisfies the predefined criteria: **($\mathbb{C}$1)Generalizability**, by *GATRes* being a spatial-based GNN approach that has topology awareness in its decision; **($\mathbb{C}$2)Adaptability**, by random masking that dynamically changes sensor positions and myriad contextual snapshots from our data generation tool; and **($\mathbb{C}$3)Robustness**, by effectively evaluating the model on unseen time-relevant data with respect to uncertainty conditions.

In addition, *GATRes* achieved pressure reconstruction with an average relative error of 7% and an absolute error of 1.93 water column meters on a 95% masking rate (see Table 5.4). We attribute its success primarily to the fundamental blocks and training strategy. These blocks update the connection weights using nodal features and, therefore, relax the original topology in a given Water Distribution Network. This relaxation provides

robustness and generalizability to *GATRes* in uncertain conditions and across diverse network topologies, which may vary in size, headloss formula, and component configurations. Furthermore, *GATRes* utilizes a random sensor replacement strategy, eliminating the need for time-consuming retraining when a new sensor is introduced in the future. For these reasons, both blocks within the architecture and training strategy make *GATRes* a highly reusable and sustainable solution for predicting pressures in numerous Water Distribution Networks.

The significance of the improvement in performance of our model with respect to previous approaches was assessed using the Wilcoxon signed-rank and the paired t-tests. The results obtained for C-Town, Richmond, and L-Town benchmark datasets are statistically significant with a value $p \ll 0.05$, and comparable to the performance of the mGCN model only for Richmond dataset. In terms of applicability, previous studies found a relationship between leakage and pressure, and suggested that proper pressure management is crucial for reducing water loss [131, 259]. Therefore, having a model that improves the pressure estimation, and whose improvement is statistically significant, the applicability of our model for solving problems related to leak detection and localization is worth considering as future work.

However, it is essential to acknowledge the limitations when *GATRes* scales. The performance limit becomes apparent when comparing the larger and smaller versions of *GATRes* in Tables 5.4 and 5.5. The larger *GATRes* eventually reaches a saturation point of performance and is surpassed by its smaller counterpart. The same finding is available in both GATConvWat variants. They likely originate from inherent issues in graph neural networks, such as over-smoothing and over-squashing, where nodes tend to propagate redundant information excessively [64]. While *GATRes* employs residual connections that partially mitigate over-smoothing and contribute to the model performance, as shown in Section 5.6.5, they are unable to eliminate this phenomenon completely [118]. To address these issues in the future, potential solutions may include exploring graph rewiring strategies and subgraph sampling techniques.

While graph rewiring and subgraph sampling techniques are promising directions to mitigate over-smoothing and over-squashing, their practical feasibility involves important trade-offs. Graph rewiring strategies

typically require additional preprocessing steps and may alter the original graph structure, potentially affecting the physical or semantic interpretability of the modeled system. Moreover, identifying optimal rewiring schemes often introduces extra hyperparameters and computational overhead. Subgraph sampling techniques can reduce message-passing depth and alleviate information bottlenecks. However, they may lead to information loss and require careful design to preserve global context, particularly in dense or highly interconnected graphs. Integrating either approach into *GATRes* would therefore require substantial architectural and experimental modifications, as well as extensive validation, which fall beyond the scope of this thesis. Consequently, these strategies are identified as relevant but non-trivial directions for future research rather than immediate solutions.

5.7.2 Benefit of synthetic data

Incorporating generated snapshots into the training of deep models not only enhances their capabilities but also highlights the value of synthetic data, especially in situations where dynamic data is limited or sensor placements change.

These common issues have been found in many public benchmarks, including the five water networks reviewed in Section 5.5.1 due to missing historical patterns and privacy issues. They have made reproducibility a persistent challenge in water network research.

As a solution, our data generation tool extends the limits of working with these public networks without raising confidentiality concerns. For practical purposes, the synthesized training set could include a wide range of cases, reducing the risk of long-term incidents that may not have occurred in historical records. Thus, it boosts model robustness when dealing with unforeseen scenarios.

5.7.3 Relationship between hydraulic simulations and *GATRes*

Yet, an intriguing question arises: Can we replace traditional mathematical simulations with data-driven models like *GATRes*? Conventional simulations bridge the interaction between hydraulic experts and the

Water Distribution Network in water management. Such an interaction should be preserved in the design or analysis phase. In deployment, especially for Digital Twin or water systems on big data, pressure estimation models often deal with heavy computation and require a low response time [167]. In this case, *GATRes* and GNN variants can be alternative approaches due to their competitive results and impressive throughput (see Table 5.6). However, these deep models may involve the risk of over-relaxation of energy conservation laws and other constraints within the actual networks. The risk is often minimal in pure physics-based simulations.

Accordingly, these simulations still play a critical role in data synthesis as they define a valid boundary for newly created models through their generated training samples and testing environments. When fast computation is required, *GATRes* is a good alternative to estimate pressure in a large WDN with unlimited sensor streams. In the future, it is promising to focus on physics-inspired models that can regularize *GATRes* to preserve fundamental physical laws and yield more confident results.

5.7.4 Generalization

GATRes is able to generalize to previously unseen WDNs by design, given the ability of spatial methods (e.g. *GAT*) to generalize across graphs [29]. On the contrary, previous works that rely on spectral approaches suffer from the generalization problem because their convolutions (e.g., ChebNet) depend on the eigenfunctions of the Laplacian matrix of a particular graph [254].

GATRes trained on multiple WDNs simultaneously produced a MAE of 3.06m and a MAPE of 9.98%, on average, during zero-shot inference on the 24-hour Oosterbeek WDN. These results show a promising direction given that pressure estimation was performed on a completely different, previously unseen, WDN and they do not deviate significantly from those obtained with the GATRes-small model. The zero-shot inference offers immediate monitoring of a target WDN given a sufficiently pretrained model, eliminating the need for data generation and model training specific to each WDN. These results can be improved by adding more WDNs in the pretraining data and exploring self-supervised pretext tasks

like nodal degree estimation, link prediction, and shortest paths. Moreover, the fine-tuned model on the target Oosterbeek dataset produced a reduction in MAE of 1.41% relative to the model trained directly and only on Oosterbeek.

The results of our first attempts towards generalization (see Table 5.7) show that our approach is worth further exploration. GNN models fail to generalize when the local structures of the graphs in the training data differ from the local structures in the test data [239]. In our case, the term "local structures" does not refer only to topology but also to water system components. By training the model with several WDNs and myriad scenarios, we can better prepare for unforeseen configurations and potentially include or approximate them within our dataset. This can mitigate the impact of heterogeneity in terms of number and type of system components. Then, a possible explanation for the generalization capabilities of *GATRes* is the training on several WDNs simultaneously. Using graphs that differ in size and structure for training allows *GATRes* to learn a richer set of local structures that may be present in the target WDN.

Despite these promising results, several questions remain unanswered. For example, how to choose the right WDNs in order to enrich the training data in terms of local structures diversity? How to design a pretraining task that can effectively capture the local-level patterns and extrapolate those to unseen larger graphs? How to train a GNN-based foundation model, in the Water Management Domain, that can be applied to different downstream tasks on any WDN topology? All these questions open paths for future research.

5.8 Conclusions

In this chapter, we presented a hybrid physics-based and data-driven approach to address the problem of state estimation in WDNs. We leveraged mathematical simulation tools and GNNs to reconstruct the missing pressures at 95% of the junctions in the network, using only 5% of junctions observed during training. We also tested our approach on more extreme cases of sensor sparsity, reaching up to a 99% masking rate. The outcome of our work is a new state-of-the-art for pressure

estimation in WDNs, with two main contributions. First, a training strategy that relies on random sensor placement making our GNN-based estimation model robust to unexpected sensor location changes. Second, a realistic evaluation protocol that considers real temporal patterns and noise injection to mimic the uncertainties intrinsic to real-world scenarios.

Our model was evaluated on a large-scale network in The Netherlands, showing a clear improvement over previous approaches. *GATRes* obtained an average MAE of 1.94m, which represents an 8.57% improvement with respect to other models. Similarly, it showed an average MAE reduction of $\approx$40% on other WDN benchmarks with respect to previous approaches. We attribute the high performance of *GATRes* to its building blocks and training strategy. These blocks relax the original topology, leveraging nodal features to re-weight the connections by means of an attention mechanism. Despite its success, there are still some aspects that demand further exploration. On the one hand, while the residual connections mitigate the over-smoothing problem inherent to GNNs, the phenomenon is not completely removed. Therefore, other techniques such as graph rewiring and subgraph sampling would be fruitful directions for further work. On the other hand, our multi-graph pretraining strategy is a promising direction towards model generalization and transferability in the WDN domain.

Nonetheless, further research is needed to explore the connection between network topologies and pretraining tasks, as well as the applications of state estimation models such as pipe burst identification, leak localization, and anomaly detection.

Chapter 6

Graph Foundation Models for Water Distribution Networks

The ever-increasing computational power, the enormous volumes of data available, the advances in Artificial Intelligence (AI), among other recent technological developments have given rise to the so-called Foundation Models. These special types of deep learning models are trained on diverse and vast amounts of data to solve generic tasks [24]. Then, the models are adapted to tackle specific problems from a particular domain. Foundation models have accelerated scientific discoveries in biology, medicine, chemistry, atmospheric research to name a few. Our view is that water research can also benefit from these technological advancements to boost scientific exploration and optimize day-to-day operations of water distribution networks. We envision that current deep learning methods for graphs, tailored to the water domain, can be the enablers of Graph Foundation Models for Water Distribution Networks. Since these models are trained on generic tasks, they learn more accurate data representations based on the topology of Water Distribution Networks and the properties of junctions, pipes, and other components. As a result, Graph Foundation Models will enable faster and better solutions for specific problems, e.g., network state estimation, water quality control, leak detection and localization, sensor placement, among others. This chapter describes our first approach to Graph Foundation Models for the WDNs domain. We provide the DiTEC-WDN dataset to satisfy the need for data in the water domain, which is essential for training data-hungry deep learning models. Our experiments show the feasibility of GFM for solving WDN-related tasks and especially their potential in terms of model generalization.

6.1 Data and AI shaping research and industry practices

The data and AI revolution witnessed today has brought to life what a few years ago was considered pure science fiction. Self-driving cars, talking to a virtual assistant in any possible language, and chatting with a bot that makes you doubt whether it is a real person are just a few examples of what these disruptive technologies have made possible. All these applications have been developed using techniques that involve generic models, trained on broad data, which can be adapted for solving a wide range of particular tasks, i.e., Foundation Models [24]. Large Language Models (LLMs) such as GPT-3 [30], Llama [67], or Gemini [79] are prominent examples of Foundation Models for Natural Language Processing. DALL-E 3 [18], CLIP [171], and SAM [119] are multimodal Foundation Models in the computer vision domain that combine text and image data.

Importantly, models of this type are not only leveraged by industry to develop commercial applications, but they are improving research and boosting scientific discovery. BERT [63], an early predecessor of large language models, also influenced the development of AlphaFold, a foundation model for biology that achieved major breakthroughs in protein structure prediction, a problem open for more than 50 years [112]. Another example is CHGNet [60], a model that allows researchers to run atomistic simulations giving insights into charge-transfer phenomena in computational chemistry, physics, biology, and materials science.

Although there are several state-of-the-art vision, language, and multimodal Foundation Models, less attention has been paid to Foundation Models for graph-structured data. There are ongoing efforts to Universal Graph Foundation Models, but they are still task-specific or domain-specific approaches in practice [141]. For example ULTRA [76] is a Graph Foundation Model exclusively for link prediction in Knowledge Graphs, or MolGPS [195] for molecular graphs. Moreover, existing general-purpose graph foundation models are primarily designed around discrete prediction objectives, including node or graph classification, and link prediction. Those models rely on inductive biases that are not well aligned with the characteristics of water distribution networks. In

particular, WDNs involve continuous-valued node states, physics-driven constraints, and regression tasks that require preserving hydraulic consistency across the network. Applying current graph foundation models to such settings would require substantial architectural modifications, including changes to output heads, loss functions, and training objectives, as well as retraining on domain-specific data. This motivates the development of a domain-specific foundation model that is explicitly designed to capture the structural and physical properties of water distribution networks.

We strongly believe that water research and the operations of water distribution networks can be substantially improved if Foundation Models are integrated into their tooling and common practices. In our vision, Graph Foundation Models (GFMs) for Water Distribution Networks are seen as the ultimate goal of data-driven approaches. Data-driven methods are the straightforward solution for several water management problems such as leakage localization [259], demand forecasting [104, 168], state estimation [8, 209, 233]. Although traditional data-driven models can be very accurate for a particular task on a particular WDN, their performance is challenged when certain conditions change. They are engineered as independent and isolated solutions for a particular problem, attached to a particular WDN, a specific water management company, or a specific research lab experiment. GFMs, on the contrary, are not focused on solving a particular task for a specific WDN, but their focus is to learn accurate generic data representations based on the WDN topologies and the properties of their junctions, reservoirs, tanks, pipes, and all other components. This equips them with generalization capabilities that allow researchers to apply them to solve prediction tasks in unseen WDNs.

For example, applying a data-driven model for leakage detection to a different WDN implies retraining the model, recalibration, and repeating the overall process from scratch. This limits model reusability and efficient use of resources. On the contrary, having a GFM will allow practitioners, in many cases, to reuse the model and apply it directly to find leakages in new WDNs. Most importantly, the ability of such a GFM for detecting leakages will be better than a model that was trained only on that task for a particular WDN. In cases where GFMs need to be adjusted for a tailored solution, retraining time will be lower, and the

performance will exceed that of a single-purpose data-driven approach.

6.2 Implementation of Graph Foundation Models tailored to the Water Domain

We propose that GFMs for WDNs can be materialized by harnessing synthetic data generation, a proper pretraining strategy for Graph Neural Network (GNNs) models, and model adaptation to solve the main task using existing fine-tuning techniques.

An overview of the envisioned approach is presented in Figure 6.1. The first stage implies the use of several WDN topologies and physics-based simulation tools for synthetic data generation. Then, GFMs are pretrained on generic tasks to learn data representations from the structural properties of WDNs and the features of nodes and links in the network. Finally, the pretrained model is adjusted to learn representations related to particular problems in the water domain. The adapted models can be applied to solve specific tasks, e.g., state estimation, water quality control, leak detection, and localization.

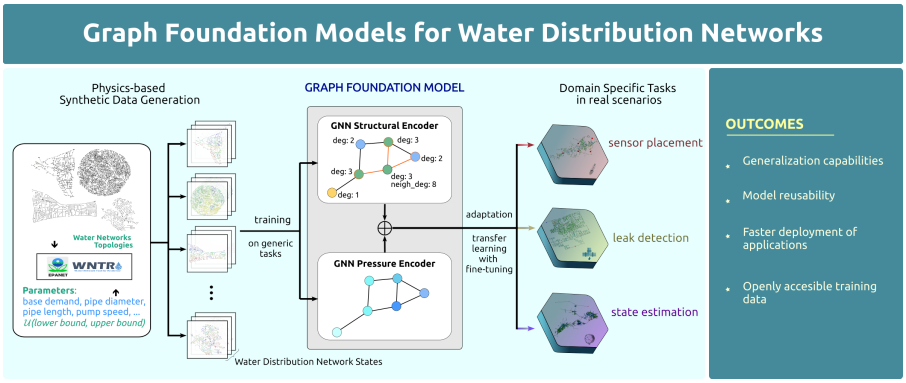


Figure 6.1: Graph Foundation Models for Water Distribution Networks. (Best viewed in color)

6.2.1 Synthetic data generation for data-hungry models

Deep learning data-driven models are known for being data-hungry, i.e., they require vast amounts of data for model training. Ironically, while

in data-centric approaches data is treated as the most important asset in water management [74], in practice data is scarce and seldom shared, let alone outside water management companies. Sensors for monitoring the state of the networks are expensive. Thus, it is common that only a few of them are installed in sparse locations along the network. In addition, privacy concerns limit data sharing among practitioners. Moreover, as researchers applying data-driven approaches, we rarely find ready-to-use data to train our models. Instead, we find configuration files of specific WDNs that serve as input for mathematical tools to generate data. Therefore, the first challenge in the era of data-driven methods is how to obtain the data needed for model training. We argue that synthetic data is the key for data-hungry models. This will not only allow researchers to materialize GFMs for WDS, but also facilitate data sharing and faster model development.

The use of physics-based simulation tools (e.g., EPANET, WNTR) during design, optimization, and control phases in WDS is a proven approach [1, 44, 77, 190]. Generating synthetic data using these tools is computationally expensive but unbounded in terms of dataset size. Simulation tools allow generating an unlimited number of scenarios, which represent arbitrary WDN states during network operation. Those scenarios can be used at the next stage for training GFMs.

Following existing approaches for data generation, we propose random adjustment of the dynamic configuration parameters [93, 209]. All the dynamic parameters (e.g., reservoir total heads, tank levels, junction demand, etc.) required as input for the simulation tools can be sampled from a uniform distribution. These, together with the static parameters (e.g., elevation, pipe diameter) and the topology, are passed to the data generation process to create arbitrary WDN states.

In addition, we provide an extended version of the dataset by also varying the static parameters such as elevation, pipe diameters, or pipe length [210]. Although unrealistic, the results show that it can contribute to more diverse and varying scenarios, covering a larger subset of the entire input space. Figure 6.2 shows a comparison of the data distribution between baseline networks (orange) and the DiTEC-WDN dataset (cyan) along the *demand* and *pressure* axes. It is clear that our dataset covers a much higher spectrum in terms of demand, contributing to data variety.

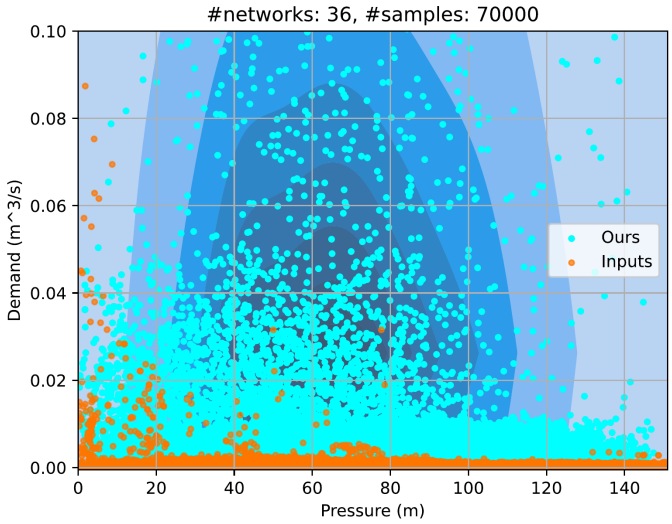


Figure 6.2: Density distribution of *pressure* and *demand* across WDNs in DiTEC-WDN (cyan) and baselines from existing input files (orange). The contours denote the data point density of the DiTEC-WDN dataset, with darker blue indicating higher concentration at the center and lighter blue showing lower density when going outward. In baseline networks, data points whose pressure is outside the range of $(0, 151]$ in meters, are excluded due to the impractical operation conditions [209].

This variability is essential for training cutting-edge data-driven algorithms. Moreover, augmenting the search space can help GFMs to adapt to the uncertainties intrinsic to real-world use cases. On the left, baseline data points correspond to high demand and low pressure, indicating that only a few nodes receive sufficient supply while most experience pressure drops. Similarly, on the right side, the pressure of baseline points is stable only when their corresponding demand approaches zero. This reflects the demand scarcity and suboptimal simulation parameters.

Demand is one of the most important inputs for solving the WDN hydraulics [85]. Surprisingly, it is one of the inputs that is rarely found in the WDN configuration files. It is common to find just a few demand patterns reused many times on several nodes in the network [161], or no demand patterns at all [173]. The stochastic nature of water

demand explains some of the uncertainties found in WDNs [247], which should be properly modeled and considered in the simulations [33]. Hence, reusing the same demand patterns on multiple nodes assumes that several users/consumers have exactly the same water consumption behavior, which is unrealistic. This not only harms the robustness of the models to uncertainties, but also limits the variety of the data, which is especially important for deep learning data-driven approaches. Therefore, we propose an Automatic Demand Generation algorithm to create unique demand patterns for each node in the WDN.

Automatic Demand Generator

The Automatic Demand Generator (ADG) algorithm aims to generate the junction demand patterns for each node in a WDN. The demand pattern is defined following an additive model of three components: a daily pattern, a yearly seasonal pattern, and noise, as expressed in Equation 6.1.

$$D = \text{daily}(x_t) + \text{yearly}(x_t) + \epsilon_t : \ t \in T \tag{6.1}$$

where D is the demand pattern for each node in the network, daily(x_t) is the daily pattern, yearly(x_t) is the yearly seasonal pattern, and ϵ_t is white noise. The demand patterns generated are multiplier time series, i.e., a factor that is multiplied by the *base demand* of each node specified in the configuration file of each WDN. The generated time series are normalized in the range $[0, 1]$.

Daily Pattern. The daily pattern defines the water consumption per day based on consumption profiles: household, commercial, extreme-demand, and zero-demand. The consumption profiles are determined by splitting the 24-hour day into four 6-hour segments. Thus, starting at midnight, these segments represent the water consumption from 00:00 to 06:00, 06:00 to 12:00, 12:00 to 18:00, and 18:00 to 00:00. Each segment is assigned either a low, medium, or high consumption. The range for low, medium, or high consumption is defined by lower and upper bounds determined at random. Thus, from N random numbers in the range $[0.00, 1.00]$ we compute the quantiles Q1 and Q3. Then, the

low consumption goes from $[0.00, Q1)$, the medium consumption goes from $[Q1, Q3)$, and the high consumption is in the range $[Q3, 1]$. For example, the household profile is represented as (low, high, medium, low). It is assumed that there is a low consumption between midnight and six in the morning, with a peak consumption in the morning when people are preparing for work. Then, after noon, the demand gradually decreases during the day because people are at work, and finally the demand is low again at the end of the day when people are going to bed. In a similar way, the commercial profile is defined as (high, high, high, medium). In this case, assuming a high consumption most of the time with a small decrease at the end of the day.

Using the consumption ranges described before, we generate random samples for each of the 6-hour segments. The number of $\mathrm{samples\_per\_hour}$ is determined based on the sampling frequency ($\mathrm{time\_step}$) defined in the configuration file. Those 6-hour segments are then concatenated to compose the 24-hour period corresponding to one day. Then, these 24-hour samples are repeated to span the entire $\mathrm{duration}$ of the demand pattern. The daily demand pattern is generated using the periodic function described in Equation 6.2.

$$\text{daily}(x_t) = cos(x_t) + sin(x_t) + z_t : \ t \in T \tag{6.2}$$

where the $cos(.)$ and $sin(.)$ terms introduce the daily periodicity in the time series, x_t represents the previously generated random sample at time t, and the z_t term represents white noise. The noise component guarantees that each repetition of the 24-hour pattern along the time series is not a fidelity copy of the previous one. Finally, we use the Savitzky-Golay filter [138, 183] to smooth the generated time series.

After the consumption profiles are defined, they have to be assigned to each node in the network. Hence, we need to determine which nodes belong to the household profile and which ones to the commercial. Domain knowledge indicates that commercial nodes are grouped in certain regions of the WDN. In order to resemble this characteristic, we propose to cluster the nodes into two main groups, household and commercial. The clusters are computed using the Louvain community detection algorithm, a heuristic approach that maximizes the modularity of the network [22]. This algorithm works in two phases. In the first

phase, each node i is isolated and belongs to a community C. Then, the modularity gain is computed after each node is moved to its neighboring communities. If there is no positive gain in modularity, the node remains in its original community. This phase is repeated until no individual move can improve the modularity. For directed graphs, the modularity gain is computed as follows [22, 68, 208]:

$$\Delta Q = \frac{k_{i,\text{in}}}{m} - \gamma \frac{k_i^{\text{out}} \cdot \Sigma_{\text{tot}}^{\text{in}} + k_i^{\text{in}} \cdot \Sigma_{\text{tot}}^{\text{out}}}{m^2} \tag{6.3}$$

where m is the size of the network, γ is the resolution parameter that controls the size of the communities [156], k_i^{out}, k_i^{in} are the outer and inner weighted degrees of node i, and $\Sigma_{\text{tot}}^{\text{in}}$, $\Sigma_{\text{tot}}^{\text{out}}$ are the sum of in-going and out-going links incident to nodes in community C.

In the second phase, the communities found in the previous step become nodes in the network, and the weights of links in the new graph are the sum of the weights of the links between nodes in the corresponding communities. Then the whole algorithm is applied again. The algorithm stops when no modularity gain is achieved or when the modularity is lower than a certain *threshold*.

At this stage, we have coherent communities within each WDN. Now, we need to define the number of nodes from those communities that will be assigned to either commercial or household profiles. According to the statistics provided by the association of water companies in the Netherlands, about 28% of the users belong to the commercial sector [216–218]. We randomly choose the $\text{percentage\_commercial}$ from the range $(0.25, 0.35)$. This allows us to resemble commercial consumption profiles in other countries around the world. We set the number of nodes that will be assigned to the commercial consumption profile as $\text{num\_nodes\_commercial} = \textit{floor}(\text{percentage\_commercial} \times \text{total\_number\_of\_nodes})$. After that, we iterate the communities found in the previous stage and sequentially assign the nodes in each community to the commercial consumption profile until we reach the $\text{num\_nodes\_commercial}$. Finally, the remaining nodes will be assigned to the household profile at this stage. While household and commercial profiles are self-explanatory, extreme and zero-demand are a special type of consumption profile.

Extreme-demand is a special case for some nodes with a very high water consumption. Thus, the extreme-demand profile is represented as (high, high, high, high). Usually, an extreme node represents a group of nodes, commonly external to the water network, but also connected to it. We set the $\mathrm{extreme\_dem\_rate} = 0.02$, i.e., 2% of the scenarios will have nodes whose demand is always high. In addition, we limited the number of nodes per scenario that can have extreme demand values; specifically, we set $\mathrm{max\_extreme\_dem\_junctions} = 2$. Domain knowledge can help to determine this parameter if the number of extreme nodes is known beforehand. The nodes to be assigned an extreme-demand profile are chosen at random and excluded from the nodes in the household or commercial profile. Then, for these nodes, the demand is randomly generated in the range $[Q3, 1]$, as described before.

Zero-demand is another special case that represents nodes that do not consume water, but which are part of the network. Thus, these nodes always have a zero-demand value. These nodes are used for monitoring and control of the network operation, or they are modeled due to a planned expansion of the network. We set the $\mathrm{zero\_dem\_rate} = 0.05$, i.e., 5% of the scenarios will have nodes whose demand is zero. Likewise, 5% of the total number of nodes in the WDN will be assigned the zero-demand profile. Alternatively, the $\mathrm{zero\_dem\_rate}$ can be set to the ratio between the number of nodes in the baseline network whose *base demand* is zero with respect to the total number of nodes, and accordingly, the number of nodes belonging to this profile. The zero-demand nodes are chosen at random and excluded from the remaining household or commercial profiles.

The presence and use of both extreme-nodes and zero-demand nodes when modeling WDNs are seen in the baselines and also confirmed by experts in the water management domain. Including these two profiles in the generated data enables us to cover a wider range of pressures and demands compared to the baselines. Otherwise, if the baselines have those types of nodes but those are not included in our generation algorithm, there is a mismatch between the baselines and our data. Our goal is to extend the range of the generated data but still cover and resemble the WDN baselines.

Yearly Pattern. The yearly pattern defines the trend of water consumption in the entire year, considering a seasonal component with a peak consumption in summer. The default configuration assumes the European summer season starting in June with a 3-month span. In addition, to introduce variability in the data, beneficial for training deep learning models, we randomly move the summer period throughout the year for approximately 20% of simulated scenarios. This approach introduces the seasonal patterns from other regions across the globe. The yearly pattern is composed of a yearly component, a seasonal component, and noise, as described in Equation 6.4.

$$\text{yearly}(x_t) = y(x_t) + s(x_t) + z_t : \ t \in T \tag{6.4}$$

where $y(x_t)$ is the yearly component generated using a Fourier timeseries as described by Equation 6.5, $s(x_t)$ is the seasonal pattern generated using a periodic cosine function as described by Equation 6.6, and z_t is white noise.

$$y(x_t) = A_0 + \sum_{n=1}^{H} \left(A_n \cos\left(2\pi \frac{n x_t}{\text{num\_samples}}\right) + B_n \sin\left(2\pi \frac{n x_t}{\text{num\_samples}}\right)\right) : \ t \in T \tag{6.5}$$

where the Fourier coefficients A_n and B_n determine the amplitude of the signal, and they are randomly sampled from a uniform distribution in the range $[0, 1)$, the value of H represents the number of harmonics used for the time series, and the periodicity of the signal is 24-hour for the short-term dataset and 1-year for the long-term. The periodicity is given by the number of samples parameter num_samples.

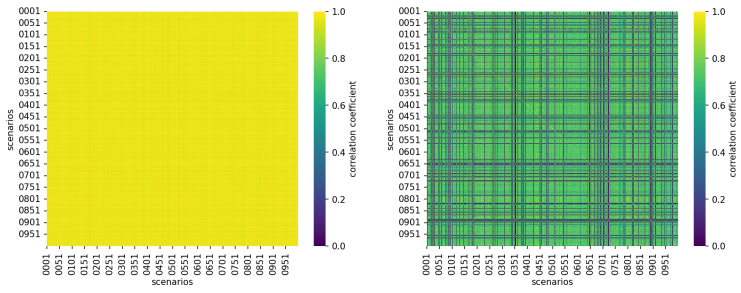
$$s(x_t) = C\left(cos\left(2\pi \frac{x_t - s_{\text{peak}}}{\text{num\_samples}}\right)\right) : \ t \in T \tag{6.6}$$

where C is a constant that represents the amplitude of the signal, reaching its maximum value in the summer peak s_{peak}, num_samples defines the periodicity of the signal. Finally, the yearly time series are normalized in the range $[0, 1]$.

Noise. The noise component ϵ_t, from Equation 6.1, is used to model the high and unexpected fluctuations in water consumption. Such

variations can be caused by unpredictable changes in consumer behavior, network maintenance, transients, or other unforeseen circumstances [220]. The noise component was sampled from a Gaussian normal distribution centered at zero, with a standard deviation randomly sampled from a uniform distribution in the range $[\text{min\_noise}, \text{max\_noise}]$.

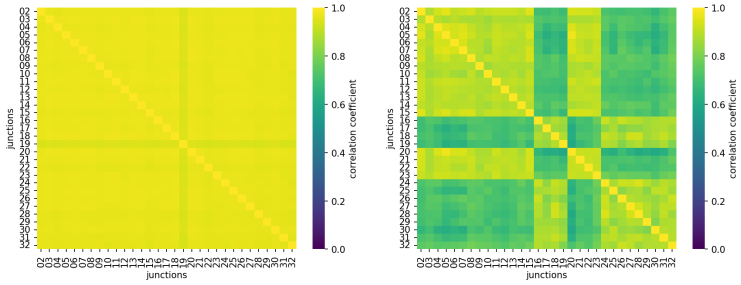
In order to validate our DiTEC-WDN dataset, we compare it against LeakDB, a dataset commonly used in research currently, which only includes the Hanoi WDN. Figure 6.3 shows the demand correlation matrices between the 1,000 scenarios generated in LeakDB and our generated data. As can be seen in Figure 6.3a, the scenarios generated in LeakDB are highly correlated. The correlation matrix shows only slight variations between some scenarios, which implies data redundancy. This limits the capacity of deep learning models to learn from such data. On the contrary, in our dataset, the correlation between scenarios is much lower, as can be seen in Figure 6.3b. This confirms the diversity of the generated data, allowing the models to see a larger space of solutions during the training process.



(a) LeakDB (b) DiTEC-WDN

Figure 6.3: Correlation matrices of generated demands between all scenarios in Hanoi WDN. (a) Correlation between scenarios in LeakDB [220]. (b) correlation between the scenarios in our dataset. Both matrices include all 1,000 scenarios, each containing 1-year of demand data. The low correlation between scenarios in our dataset shows the diversity of the data, contrary to the similarity observed across LeakDB scenarios.

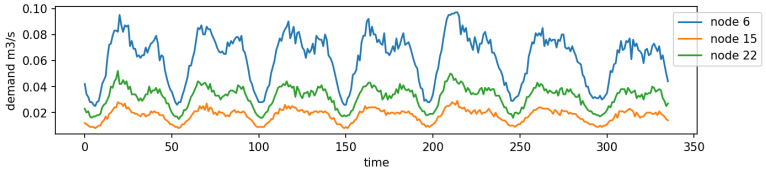
Similar conclusions can be drawn from the correlation matrices between the junction demands in an arbitrary scenario (see Figure 6.4). The high correlation shown in Figure 6.4a exposes the negative effect of demand pattern overuse in existing approaches. In contrast, Figure 6.4b shows a moderate correlation between junction demands in our data, implying there is a pattern in consumption demand, but this is not identical for every node in the WDN. In addition, the block patterns shown in Figure 6.4b display the difference between households and commercial consumption profiles described in Section 6.2.1: Automatic Demand Generator.



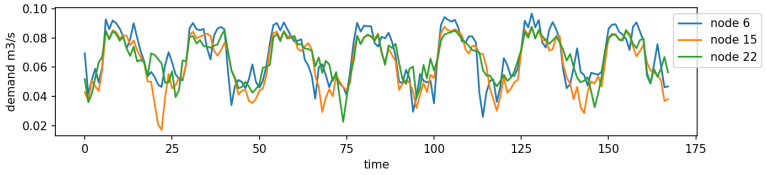
(a) LeakDB (b) DiTEC-WDN

Figure 6.4: Correlation matrices of generated demands between junction nodes in a randomly chosen scenario from Hanoi WDN. (a) Correlation between junction demands in LeakDB [220]. (b) Correlation between the junction demands in our dataset. The high correlation in LeakDB shows the overuse of demand patterns for several nodes, contrary to what it is observed in our dataset. The blocks in our dataset highlight the difference between household and commercial profiles.

Finally, the time series of one-week demand for three nodes from a random scenario in LeakDB and our dataset are shown in Figure 6.5. The time series depicted in Figure 6.5a show how a single demand pattern is reused for the three nodes in LeakDB. While the noise shows some subtle variations, each time series looks like a translated and scaled version of the others. In contrast, our data exhibit consumption patterns, but the fluctuations in each time series are independent, as shown in Figure 6.5b.



(a) LeakDB



(b) DiTEC-WDN

Figure 6.5: Time series of the generated demands of three randomly chosen junction nodes from Hanoi WDN. (a) A week of demands generated in LeakDB [220], sampled every 30 minutes. The reuse of a single pattern for different nodes is clearly observed in LeakDB. (b) A week of demands from our dataset, sampled every 60 minutes. In this case, the fluctuations observed in the data show a different consumption pattern per node.

6.2.2 Water Distribution Networks and Graphs

Water Distribution Networks can be naturally modeled and represented as graphs. The application of graph theory for modeling WDNs and solving water management-related problems has shown to be successful in several studies. For example, WDN clustering [91, 166], water quality assessment [189], WDN design and optimization [190]. In addition, Graph Neural Networks have shown high performance at tasks where data can be modeled as a graph, including evidenced success for different WDN-related problems such as leak detection and localization [77, 246], pump speed-based state estimation [25], and pressure estimation [8, 92, 209]. These two facts make Graph Neural Network approaches stand out as the most suitable candidates to be the backbone for Graph Foundation Models.

In the context of WDS, it is important to determine generic tasks to

train GFMs that can be tailored to the water domain. We identified some links between WDS research and GNNs literature. In water research, the node degree (the number of connections to other nodes) distribution is used to classify WDNs [84]. There is evidence that the average node degree is indicative of the generalization capabilities of GNNs to unseen graph topologies [127, 239, 261]. Thus, predicting the node degree or the neighborhood node degree, as described in [84], can be a suitable generic task for pretraining GFMs.

Besides node degree, other concepts from graph theory (e.g., shortest path lengths, eccentricity) have been used in the water domain to characterize WDNs based on their structural properties [83, 190, 191]. Thus, another pretraining task could be to predict the shortest path between demand nodes and source nodes. In addition, as already proposed in the WDN literature, the weights for the shortest paths can be calculated based on different properties, e.g., flow, diameter, length, roughness coefficient, or using the actual demand at each node. Similarly, the GNN literature shows that models trained on classical graph algorithms, e.g. shortest paths, can generalize to bigger graphs than those seen during training [52, 127].

The previous analysis of existing techniques in the GNNs literature and their relationship to WDS opens paths for further research. Identifying and leveraging such connections will provide an effective pretraining strategy tailored to the WDS domain. As a result, researchers and practitioners in the field would have reusable models, which do not need recalibration when applied to a different WDN. Ideally, with a proper pretraining strategy, GFMs are trained once, on a large number of networks, and they are applicable off-the-shelf to make inferences on a new, unseen water network topology.

6.2.3 Model adaptation for specific water management related tasks

Model adaptation refers to the process of adjusting a pretrained GFM to solve a specific problem. The "knowledge" previously acquired is then transferred to a new model ("*target model*") aimed at solving a specific problem ("*target task*") in a different WDN ("*target network*").

For example, let us assume that the *target task* is pressure estimation. During pretraining, the GFM was trained on a generic task, e.g. to reconstruct the node centrality. At this stage, the pretrained GFM model is trained to reconstruct the pressures at every node in the network. GFM models learn better data representations of junctions and pipes based on their features and the WDNs' topology. This knowledge is transferred to the pressure estimation task. This will reduce training time and improve the performance of the target model.

Model adaptation can be achieved by applying transfer learning, a technique that is very successful in the Computer Vision and Natural Language Processing domains, indicating that its use in GFMs for WDS deserves further analysis. Transfer learning techniques have proved to accelerate and/or improve the performance of a model with respect to the ones trained to solve a particular task for a particular dataset [163, 242].

In our vision, model adaptation will accelerate the learning process and improve models' capabilities for a range of downstream tasks in the WDS domain, e.g. state estimation, leakage detection and localization, sensor placement, demand forecasting, water quality assessment, surrogate modeling, among others.

6.2.4 Experiments

We evaluated the performance of the proposed GFM on pressure estimation. The model inputs are node pressure, demand, and elevation, together with pipe diameter, length, and roughness coefficient. We trained a model to reconstruct the pressure at all nodes under conditions in which only 5% of the nodes have their pressure and demand known, e.g., measured directly by sensors.

We used the DiTEC-WDN dataset [210] in our experiments. We selected 6 different WDNs for training the proposed GFM based on network size, ranging from 36 to 1325 nodes, namely: Fossolo, Federally Owned Water Main (FOWN), Oberlin (OBCL), C-Town, Ky3, and Ky8. For model testing, we used Bellingham (WA1), North Penn Water Authority System (NPCL), D-Town, Ky13, and Ky14 WDNs.

We used a 5-layer GATRes model [209] for pressure reconstruction as a baseline. A separate model for each WDN, in the test list, was

trained on 1,000 snapshots of WDN states divided into a 60:20:20 ratio for training, validation, and test, respectively. Then, each model was evaluated on an independent, complete test set of 4,800 snapshots per WDN.

The GFM model training relies on a dual encoder. A structural encoder is trained on masked node degree as input for node degree reconstruction and neighborhood node degree estimation, and a pressure encoder is trained for pressure reconstruction. We combine the output of both encoders by adding the embeddings learned from the structure of the graph to the embeddings of the pressure encoder. This treats the structural information as a bias term for the main task. Both encoders use the same GATRes model architecture as the one used in the baseline. We control the contribution of the pretraining task with a composed loss function (Eq. 6.7) that applies a cosine weight decay (Eq. 6.8) to regulate the weighting λ parameter. In that way, the importance of the degree reconstruction smoothly shifts towards the main task.

$$loss = \lambda \times \textit{degree\_loss} \; + \; (1 - \lambda) \times \textit{pressure\_loss} \tag{6.7}$$

$$\lambda = \lambda_{min} + 0.5 \times (\lambda_{max} - \lambda_{min}) \times \left(1 + cos\left(\pi \frac{epoch}{total\_epochs}\right)\right) \tag{6.8}$$

We trained a Dual Encoder Model per WDN in the test list, and evaluated again on the same 4,800 snapshots as in the baseline. This allows us to evaluate the contribution of the pretraining strategy to the performance of the main task. Additionally, the Dual Encoder GFM model was trained simultaneously on the six WDNs in the training list. Then, we applied transfer learning with fine-tuning. We used the weights of the pretrained model to initialize the target models. We fine-tuned one GFM model per target WDN and evaluated it again on the same 4,800 snapshots used before.

Finally, we ran an additional test: we fine-tuned our GFM on the real hydraulics-based models, defined by the original EPANET INP files, of the WDNs in the test list. We applied 1-shot learning for D-Town, WA1 and

NPCL-1, i.e., we used only one sample for fine-tuning. Since the ky13 and ky14 WDNs are provided as a single hydraulic timestep, there is no data available for fine-tuning.

6.2.5 Results and Discussion

The empirical results show that a model pretrained on an auxiliary task focused on the structural properties of the WDNs can contribute positively to the performance of the model on a main downstream task, e.g., pressure estimation. Table 6.1 shows the results of the baseline model trained only on pressure estimation compared with the results of the proposed Dual Encoder model trained on individual networks, and the GFM model trained on multiple WDNs. The proposed GFM model outperformed the baseline in all metrics for all WDNs except WA1.

Table 6.1: Performance comparison of our approach vs baseline on pressure estimation in five different WDNs. Best values are **bold**; second-best are underlined.

Dataset	Model	MAE	MAPE	R^2
D-Town	Baseline	1.9900 ± 0.008	0.1020 ± 0.002	0.9793 ± 0.000
	DualEncoder (ours)	1.5642 ± 0.005	0.0795 ± 0.001	0.9857 ± 0.000
	GFM fine-tuned (ours)	**1.3782** ± 0.006	**0.0705** ± 0.001	**0.9865** ± 0.000
Ky14	Baseline	1.8684 ± 0.005	0.0516 ± 0.000	0.9698 ± 0.000
	DualEncoder (ours)	1.5846 ± 0.003	0.0432 ± 0.000	0.9764 ± 0.000
	GFM fine-tuned (ours)	**1.5083** ± 0.005	**0.0407** ± 0.000	**0.9780** ± 0.000
WA1	Baseline	**0.0660** ± 0.000	**0.0093** ± 0.000	**0.9999** ± 0.000
	DualEncoder (ours)	0.3258 ± 0.001	0.0275 ± 0.000	0.9998 ± 0.000
	GFM fine-tuned (ours)	0.3506 ± 0.000	0.0313 ± 0.000	0.9997 ± 0.000
NPCL-1	Baseline	1.1893 ± 0.002	0.0384 ± 0.000	0.9916 ± 0.000
	DualEncoder (ours)	1.0985 ± 0.004	0.0372 ± 0.000	0.9921 ± 0.000
	GFM fine-tuned (ours)	**0.8841** ± 0.002	**0.0284** ± 0.000	**0.9945** ± 0.000
ky13	Baseline	1.6430 ± 0.005	0.0289 ± 0.000	0.9871 ± 0.000
	DualEncoder (ours)	1.5639 ± 0.004	0.0275 ± 0.000	0.9892 ± 0.000
	GFM fine-tuned (ours)	**1.3602** ± 0.003	**0.0238** ± 0.000	**0.9915** ± 0.000

Notably, the GFM also shows good performance on real hydraulics-based models after fine-tuning on only one sample per network. For WA1 and NPCL-1 WDNs, a *MAPE* < 0.0742 and a $R^2 > 0.9546$ with just 1-shot learning shows the validity of a pretrained GFM. Table 6.2 shows the

results of these experiments.

Table 6.2: GFM evaluation on the real hydraulics-based models of the WDNs in the test list, defined by the original EPANET INP files.

Dataset	Model	MAE	MAPE	R^2
D-Town	GFM fine-tuned on EPANET INP	4.4824 ± 0.036	0.1394 ± 0.001	0.8840 ± 0.002
WA1	GFM fine-tuned on EPANET INP	3.8638 ± 0.030	0.0742 ± 0.001	0.9546 ± 0.001
NPCL-1	GFM fine-tuned on EPANET INP	2.5410 ± 0.076	0.0541 ± 0.002	0.9798 ± 0.002

6.3 Impact on the Drinking Water Sector

Until now, water utilities have mainly gained their insights into the WDN based on hydraulic simulation tools. The extent to which these models can handle complexity and resilient requirements is limited. Experience shows that it is laborious and time-consuming to implement them at full scale. GFMs and GNNs are proven not to have these drawbacks. GFMs ensure an essentially fast model generation time. GNNs demonstrate better accuracy than other models [209, 246].

GFMs alleviate the dependency on hidden errors in the proprietary dataset. It is also possible to train the models without privacy concerns. The synthetic datasets eliminate the need for customer data. This facilitates greater capabilities and flexibility. As a result, the model development can be accelerated and improved. GFMs can support advanced analytics, intelligent recommendations to engineers, and set points for autonomous decision-making. Using synthetic training data instead of extensive investment in sensors will save money. GFMs can provide additional capabilities for adequate leak detection and localization, and for the prevention of unnecessary pumping of water. These are some of the practical implications of GFMs applied in the drinking water sector.

With these possibilities, awareness is growing. GFMs have to prove themselves worthy of trust, and in order to do this they need to generate outcomes that guarantee an even higher quality and certainty. Additional development of continuity and security controls and measures should ensure this. Initially, water utilities will continue to use conventional models to compare with the GFMs' outcomes. Growing confidence in GFMs will gradually replace the current methods, which will fade into

the background for analytic purposes. For this to happen, there should also be a shift in expertise towards a more data- and artificial intelligence-driven drinking water operation. The turnaround extends beyond the technological development of GFMs alone. This change will drive the ability of GFMs to support water utilities in guaranteeing the supply of sufficient and clean drinking water to millions of customers in an era of highly fluctuating conditions.

6.4 Conclusions

Foundation Models have become a game-changer in several fields. We postulate that water research and water distribution network operations can be substantially improved if Graph Foundation Models are integrated into their tooling and common practices. Harnessing the power of physics-based simulation tools, e.g., EPANET, will allow practitioners to generate large and broad WDN datasets that can be used to train Graph Foundation Models. The DiTEC-WDN dataset is one of the main contributions towards the realization of GFM for the WDN domain. Most importantly, these data are openly accessible and can be shared without limitations or concerns.

Graph Foundation Models can become the umbrella for data-driven approaches due to their generalization capabilities, allowing model reusability and faster deployment of water management applications. Nonetheless, drinking water professionals need to be aware that along with the benefits of using Graph Foundation Models new challenges will arise. Their adoption has to be gradual towards a more data-driven drinking water operation.

Chapter 7
Conclusions

In this thesis, we focused on designing deep learning architectures that enable inference models for smart environments to learn useful representations of sensor data modeled as a graph. In this Chapter, we summarize the contributions of this thesis and present the answers to the questions raised in Section 1.2. Finally, we discuss the limitations of our approaches and propose directions for future research.

7.1 Summary

We addressed graph representation learning from two different perspectives, each focusing on a different problem domain, a different nature of the graph topology, and a different type of prediction task.

In Part I of this thesis, we addressed HAR based on IMU sensors embedded in wearable and mobile devices. In this case, the input data are time series or sensor data readings. Thus, the graph topology is not given and it needs to be inferred from the data. We framed HAR as a graph classification problem in which the model learns a single representation for entire graphs. In Chapter 3, we showed that conventional approaches to HAR can bias prediction outcomes, regardless of the representation learning technique. Next, in Chapter 4, we proposed a GNN-based architecture for HAR and a contrastive learning approach that helps the models to learn better representations of IMU sensor data. The first part of this thesis allowed us to answer the research questions RQ-1 and RQ-2.

RQ-1. *How common are methodological flaws in the application of (Deep) Machine Learning models for Human Activity Recognition, and how do they affect the validity of claimed or reported results?*

As shown in Chapter 3, the data segmentation and feature extraction techniques based on sliding windows, followed by random train/test splits of the windows-based data samples, produce misleading results. The biases introduced by blindly following those well-defined steps within the HAR pipeline arise because continuous windows of time series data are highly correlated. As a result, the new window samples are not independent anymore, breaking the *i.i.d* assumption of machine learning models.

Under these conditions, we need to enforce the independence of training and test sets. Therefore, the subsets of data used to train the model and those used to test it, cannot be chosen at random. Random cross-validation methods for evaluating classification models were shown to be biased due to the situation described in Chapter 3. Hence, the learned representations are not indicative of powerful predictive capabilities that disentangle the explanatory factors of the observed input. It was shown that regardless of the feature extraction method and the type of classification model, the results were highly overestimated. Additionally, it was shown that using a third-party parameter for partitioning the data into train/test sets gives the flexibility to create unbiased scenarios that ensure an objective model evaluation.

RQ-2. *Can a graph representation of sensor data be used to improve the performance of prediction models on problems where there is no existing underlying graph topology? How can these models be trained to learn better data representations for Human Activity Recognition?*

Chapter 4 shows that, based on existing literature and empirical results, the Pearson correlation coefficient is a good strategy to define the graph topology. We proposed a graph creation method based on the global and local correlations in the time series data. Another challenge is to apply a training strategy for GNN-based models on the graph topology inferred from the data, which diminishes the spurious correlations that might be introduced by Pearson's correlation between time series.

Next, a contrastive learning approach enforces two GNN-based encoders to produce embeddings closer in the latent space for those global and local graphs that represent the same activity. On the contrary,

the learned representations are pulled apart in the latent space when the global and local graphs belong to different activities. The effectiveness of the proposed training approach is observed in the predictions for activities with inter-class similarity, e.g., waist bend forward and knee bending. While the global and local models confuse those activities, the confusion is removed almost completely by the representations learned with the contrastive learning approach. This shows that the proposed contrastive learning approach is a valid training strategy that clearly enables better representation learning and improves the prediction capabilities.

In Part II of this thesis, we focused on graph representation learning in water distribution networks. In this domain, the graph topology is explicitly given by the layout of the WDN. In Chapter 5, we addressed the pressure estimation task, which aims to reconstruct the pressure signal for all nodes in the network from a few sparsely located sensors. This is a node-level regression task where the model has to learn representations for each node in order to estimate the missing pressure values. In Chapter 6, we proposed Graph Foundation Models for WDNs. We discussed the challenges and proposed the strategies to implement the models, train them, and adapt them to solve WDN-specific tasks. The second part of this thesis allowed us to answer the research questions RQ-3 and RQ-4.

RQ-3. *Can graph-based neural network models be employed for state estimation in Water Distribution Networks, and what capabilities should they possess in this case? How can we enforce such capabilities from design time, and how can we incorporate them in the training strategies of graph-based state estimation models?*

The empirical results presented in Chapter 5 show that GNN-based models can be used for state estimation in WDNs. In addition, the literature review allowed us to identify the gaps and determine what is missing in existing works related to GNN-based models for pressure estimation, and to think of the capabilities that graph-based state estimation models should have.

In Chapter 5, we described the capabilities required for pressure

estimation models based on the challenges that need to be addressed. The models should be capable of estimating the pressures for several WDN topologies, even those not seen during model training. Thus, *generalizability* is one of the capabilities required by pressure estimation models. In addition, sensors may fail, or due to planned maintenance or network expansion, sensors can be arbitrarily added or removed. Hence, *adaptability* to sensor locations is required. Another challenge faced in pressure estimation in WDNs is the high uncertainty of this domain. The uncertainty is due to several external and internal factors such as changing consumption behavior, noise in sensor measurements, data transmission errors to name a few. This requires *robustness* to unexpected circumstances.

We presented the GATRes model for pressure estimation in Chapter 5, describing how the required capabilities are incorporated into the proposed model architecture and the training strategy. *Generalizability* is introduced by design through the GNN implementation based on the GAT spatial convolution. *Adaptability* is achieved by a training strategy that relies on random sensor placement, making GATRes robust to unexpected sensor location changes. Finally, *robustness* is achieved by a realistic evaluation protocol that considers real temporal patterns and noise injection to mimic the uncertainties intrinsic to real-world scenarios.

RQ-4. *How can Graph Foundation Models be designed and implemented to be tailored to the WDNs domain? How can the scarcity of data in this domain can be solved in order to train such data-hungry models? How can these models learn generalizable and transferable representations useful for solving specific downstream tasks?*

In Chapter 6, we proposed the first approach to Graph Foundation Models for WDNs. Graph Foundation Models can be achieved by harnessing synthetic data for training generalist GNN models, and using existing fine-tuning techniques to adapt the model for solving domain-specific tasks, e.g., network state estimation, water quality control, leak detection and localization.

We identified the links between GNNs research and WDNs literature.

Finding those common points is the first step towards tailored models for the WDN domain. Node degrees are used in both the GNNs and WDNs domains. In the former, they are used as an indicator of the generalization capabilities, in the latter they are used to classify WDNs. We proposed that predicting the neighborhood node degree can be a suitable pretraining strategy tailored to the water domain.

In Chapter 5, we showed that generalizability can be enforced by choosing the right GNN convolutional layer implementation. In addition, the models need to be trained at scale, i.e., not only trained on large amounts of data but on diverse scenarios across multiple WDNs. This allows the models to learn local and global structural patterns based on the topology and the features of nodes and edges in the graph. The DiTEC-WDN dataset [210], described in Chapter 6, is the first contribution to large-scale data for WDNs, which is an important milestone towards Graph Foundation Models for this domain.

Finally, transfer learning by fine-tuning is proposed as the model adaptation technique. A pretrained model is adapted to learn the specifics of the target task. We compared a model trained on a single WDN for the pressure estimation task given a few pressure sensors versus a Graph Foundation Model trained on multiple WDNs and fine-tuned for pressure estimation in a specific WDN. The adapted model via fine-tuning shows a clear performance improvement compared to a model trained from scratch on a single WDN and without considering a pretraining task for learning the structural properties of the graph.

Our first attempt at Graph Foundation Models for the WDNs domain shows its feasibility in terms of model generalization to new WDN topologies. It shows that Graph Foundation Models can contribute to research and practice to provide better solutions for specific problems in the WDNs domain.

7.2 Limitations and future work

GNN-based architectures and other methods proposed in this dissertation solve the problems addressed in each chapter. Nonetheless, as with all (deep) machine learning models, the proposed methods can always be improved.

For example, in Chapter 4, we used the Pearson correlation coefficient between sensor data signals to create the activity graphs used for HAR. Although this approach is shown to be successful and widely used in other fields, e.g., electroencephalography data analysis, it only considers linear correlations in the data. An important improvement can be to train end-to-end models that are able to learn jointly the graph topology and the embeddings used for the target task. Such approaches can discover hidden inter- and intra-relationships between sensor data channels, and discard spurious correlations that might be introduced by the Pearson correlation.

Another room for improvement lies in the corruption method applied to the local graph representations prior to contrastive learning. The current approach randomly removes or permutes nodes and edges in the graph. A further study could assess the effects of applying augmentation techniques to the sensor data itself before activity graph generation. Various techniques from the signal processing domain could be used to augment the original input, e.g., inverting the signals or stretching and warping the time series.

Over-squashing and over-smoothing are known limitations of GNN-based models. Our proposed architecture, GATRes, mitigates over-smoothing by the use of residual connections. Further research should focus on exploring other techniques to alleviate the over-squashing problem. For example, experiments to evaluate different graph rewiring techniques to enable long-range interactions between nodes in the graph would be an interesting addition to this work.

Finally, Graph Foundation Models for WDNs were proposed. We presented some ideas for pretraining these models and how to adapt them for solving a downstream domain-specific task. Further work needs to be done to evaluate more pretraining strategies considering other topological features of the data apart from node degrees. A natural progression of this work is to adapt the proposed GFM to solve multiple downstream tasks in the WDNs domain.

Bibliography

[1] C. Abdelbaki, M. M. Benchaib, S. Benziada, H. Mahmoudi, and M. Goosen. Management of a water distribution network by coupling gis and hydraulic modeling: a case study of chetouane in algeria. *Applied Water Science*, 7:1561–1567, 2017.

[2] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pages 1–3, 2024.

[3] M. A. R. Ahad, A. D. Antar, and M. Ahmed. Iot sensor-based activity recognition. *IoT Sensor-based Activity Recognition. Springer*, 2020.

[4] D. Anguita, A. Ghio, L. Oneto, F. X. Llanas Parra, and J. L. Reyes Ortiz. Energy efficient smartphone-based activity recognition using fixed-point arithmetic. *Journal of universal computer science*, 19(9):1295–1314, 2013.

[5] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, and J. L. Reyes Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proc. 21th international European symposium on artificial neural networks, computational intelligence and machine learning*, pages 437–442, 2013.

[6] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, pages 40 – 79, 2010.

[7] C. T. Arsene and B. Gabrys. Mixed simulation-state estimation of water distribution systems based on a least squares loop flows state estimator. *Applied Mathematical Modelling*, 38(2):599–619, 2014.

[8] I. Ashraf, L. Hermes, A. Artelt, and B. Hammer. Spatial graph convolution neural networks for water distribution systems. In *International Symposium on Intelligent Data Analysis*, pages 29–41. Springer, 2023.

[9] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[10] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas. Window size impact in human activity recognition. *Sensors*, 14(4):6474–6499, 2014.

[11] O. Banos, R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez, and C. Villalonga. mhealthdroid: a novel framework for agile development of mobile health applications. In *Int. workshop on ambient assisted living*, pages 91–98. Springer, 2014.

[12] O. Banos, M. Toth, and O. Amft. REALDISP Activity Recognition Dataset. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5GP6D.

[13] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. Reutter, and J. P. Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2020.

[14] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[15] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[16] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[17] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[18] J. Betker, G. Goh, L. Jing, T. Brooks, J. Wang, L. Li, L. Ouyang, J. Zhuang, J. Lee, Y. Guo, et al. Improving image generation with better captions, 2023.

[19] V. Bianchi, M. Bassoli, G. Lombardo, P. Fornacciari, M. Mordonini, and I. De Munari. Iot wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment. *IEEE Internet of Things Journal*, 6(5):8553–8562, 2019.

[20] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88, 2007.

[21] M. Biehl. *The Shallow and the Deep: A biased introduction to neural networks and old school machine learning*. University of Groningen Press, 2023.

[22] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[23] M. Bock, A. Hoelzemann, M. Moeller, and K. Van Laerhoven. Investigating (re) current state-of-the-art in human activity recognition datasets. *Frontiers in Computer Science*, 4:924954, 2022.

[24] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[25] C. A. Bonilla, A. Zanfei, B. Brentan, I. Montalvo, and J. Izquierdo. A digital twin of a water distribution system by using graph convolutional networks for pump speed-based state estimation. *Water*, 14(4):514, 2022.

[26] D. Bouchabou, S. M. Nguyen, C. Lohr, B. Leduc, and I. Kanellos. Fully convolutional network bootstrapped by word encoding and embedding for activity recognition in smart homes. In *Int. Workshop on Deep Learning for Human Activity Recognition*, pages 111–125. Springer, 2021.

[27] D. Bouchabou, S. M. Nguyen, C. Lohr, B. LeDuc, and I. Kanellos. Using language model to bootstrap human activity recognition ambient sensors based in smart homes. *Electronics*, 10(20), 2021.

[28] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[29] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[30] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[31] A. Bulling, U. Blanke, and B. Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):1–33, 2014.

[32] M. A. S. Campos, S. L. Carvalho, S. K. Melo, G. B. F. R. Gonçalves, J. R. dos Santos, R. L. Barros, U. T. M. A. Morgado, E. da Silva Lopes, and R. P. Abreu Reis. Impact of the covid-19 pandemic on water consumption behaviour. *Water Supply*, 21(8):4058–4067, 2021.

[33] G. H. Cassiolato, J. R. Ruiz-Femenia, R. Salcedo-Diaz, and M. A. Ravagnani. Water distribution networks optimization considering uncertainties in the demand nodes. *Water Resources Management*, 38(4):1479–1495, 2024.

[34] S. S. Chakravarthy, N. Bharanidharan, V. V. Kumar, T. Mahesh, S. B. Khan, A. Almusharraf, and E. Albalawi. Intelligent recognition of multimodal human activities for personal healthcare. *IEEE Access*, 2024.

[35] W. Chan, Z. Tian, and Y. Wu. Gas-gcn: Gated action-specific graph convolutional networks for skeleton-based action recognition. *Sensors*, 20(12):3499, 2020.

[36] C. Chatzaki, M. Pediaditis, G. Vavoulas, and M. Tsiknakis. Human daily activity and fall recognition using a smartphone's acceleration sensor. In *Information and Communication Technologies for Ageing Well and e-Health: 2nd Int. Conf., ICT4AWE 2016, Revised Selected Papers 2*, pages 100–118. Springer, 2017.

[37] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3438–3445, 2020.

[38] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys (CSUR)*, 54(4):1–40, 2021.

[39] M. Chen, Y. Jiang, X. Lei, Y. Pan, C. Ji, and W. Jiang. Drug-target interactions prediction based on signed heterogeneous graph neural networks. *Chinese Journal of Electronics*, 33(1):231–244, 2024.

[40] M. Chen, J. Liang, and H. Liu. Multi-stream p&u adaptive graph convolutional networks for skeleton-based action recognition. *The Journal of Supercomputing*, 80(8):11614–11639, 2024.

[41] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and deep graph convolutional networks. In H. D. III and A. Singh, editors, *Proceedings*

of the 37th International Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, pages 1725–1735. PMLR, 2020.

[42] Y. Chen and Y. Xue. A deep learning approach to human activity recognition based on single accelerometer. In *IEEE int. conf. on systems, man, and cybernetics*, pages 1488–1492. IEEE, 2015.

[43] Z. Chen, Y. Wang, B. Zhao, J. Cheng, X. Zhao, and Z. Duan. Knowledge graph completion: A review. *Ieee Access*, 8:192435–192456, 2020.

[44] M. Cheng and J. Li. Optimal sensor placement for leak location in water distribution networks: A feature selection method combined with graph signal processing. *Water Research*, 242:120313, 2023.

[45] H.-g. Chi, M. H. Ha, S. Chi, S. W. Lee, Q. Huang, and K. Ramani. Infogcn: Representation learning for human skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20186–20196, 2022.

[46] S. E. Christodoulou, M. Fragiadakis, A. Agathokleous, and S. Xanthos. Chapter 1 - introduction. In S. E. Christodoulou, M. Fragiadakis, A. Agathokleous, and S. Xanthos, editors, *Urban Water Distribution Networks*, pages 1–20. Butterworth-Heinemann, 2018.

[47] A. C. Cob-Parro, C. Losada-Gutiérrez, M. Marrón-Romera, A. Gardel-Vicente, and I. Bravo-Muñoz. A new framework for deep learning video based human action recognition on the edge. *Expert Systems with Applications*, 238:122220, 2024.

[48] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan. Casas: A smart home in a box. *Computer*, 46(7):62–69, 2012.

[49] D. J. Cook and S. K. Das. How smart are our environments? an updated look at the state of the art. *Pervasive and mobile computing*, 3(2):53–73, 2007.

[50] D. J. Cook and M. Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of information in medicine*, 48(05):480–485, 2009.

[51] D. J. Cook, M. Youngblood, E. O. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. Mavhome: An agent-based smart home. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(PerCom 2003).*, pages 521–524. IEEE, 2003.

[52] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

[53] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. Randaugment: Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020.

[54] G. Dai, X. Wang, X. Zou, C. Liu, and S. Cen. Mrgat: multi-relational graph attention network for knowledge graph completion. *Neural Networks*, 154:234–245, 2022.

[55] M. Defferrard, X. Bresson, and P. 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, page 3844–3852, 2016.

[56] V. Degeler. Dynamic rule-based reasoning in smart environments, 2014.

[57] V. Degeler, M. Hadadian, E. Karabulut, A. Lazovik, H. van het Loo, A. Tello, and H. Truong. Ditec: Digital twin for evolutionary changes in water distribution networks. In *International Symposium on Leveraging Applications of Formal Methods*, pages 62–82. Springer, 2024.

[58] A. Dehghani, T. Glatard, and E. Shihab. Subject cross validation in human activity recognition. *arXiv preprint arXiv:1904.02666*, 2019.

[59] A. Dehghani, O. Sarbishei, T. Glatard, and E. Shihab. A quantitative comparison of overlapping and non-overlapping sliding windows for human activity recognition using inertial sensors. *Sensors*, 19(22):5026, 2019.

[60] B. Deng, P. Zhong, K. Jun, J. Riebesell, K. Han, C. J. Bartel, and G. Ceder. Chgnet as a pretrained universal neural network potential for charge-informed atomistic modelling. *Nature Machine Intelligence*, 5(9):1031–1041, 2023.

[61] V. Dentamaro, V. Gattulli, D. Impedovo, and F. Manca. Human activity recognition with smartphone-integrated sensors: A survey. *Expert Systems with Applications*, 246:123143, 2024.

[62] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire, P. W. Battaglia, V. Gupta, A. Li, Z. Xu, A. Sanchez-Gonzalez, Y. Li, and P. Velickovic. Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 3767–3776, New York, NY, USA, 2021. Association for Computing Machinery.

[63] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[64] F. Di Giovanni, L. Giusti, F. Barbero, G. Luise, P. Lio, and M. M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, pages 7865–7885. PMLR, 2023.

[65] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[66] K. Du, R.-y. Ding, Z.-h. Wang, Z.-g. Song, B.-f. Xu, M. Zhou, Y. Bai, and J. Zhang. Direct inversion algorithm for pipe resistance coefficient calibration of water distribution systems. *Journal of Water Resources Planning and Management*, 144(7):04018027, 2018.

[67] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[68] N. Dugué and A. Perez. *Directed Louvain: maximizing modularity in directed networks*. PhD thesis, Université d'Orléans, 2015.

[69] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[70] G. Eiger, U. Shamir, and A. Ben-Tal. Optimal design of water distribution networks. *Water resources research*, 30(9):2637–2646, 1994.

[71] Z. Fang, Y. Li, J. Lu, J. Dong, B. Han, and F. Liu. Is out-of-distribution detection learnable? *Advances in Neural Information Processing Systems*, 35:37199–37213, 2022.

[72] S. Farquhar and Y. Gal. What 'out-of-distribution' is and is not. In *NeurIPS ML Safety Workshop*, 2022.

[73] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE transactions on Neural Networks*, 9(5):768–786, 1998.

[74] G. Fu, D. Savic, and D. Butler. Making waves: Towards data-centric water engineering. *Water Research*, 256:121585, 2024.

[75] M. Fu, K. Rong, Y. Huang, M. Zhang, L. Zheng, J. Zheng, M. W. Falah, and Z. M. Yaseen. Graph neural network for integrated water network partitioning and dynamic district metered areas. *Scientific Reports*, 12(1):19466, 2022.

[76] M. Galkin, X. Yuan, H. Mostafa, J. Tang, and Z. Zhu. Towards foundation models for knowledge graph reasoning. *arXiv preprint arXiv:2310.04562*, 2023.

[77] G. Ö. Garðarsson, F. Boem, and L. Toni. Graph-based learning for leak detection and localisation in water distribution networks. *IFAC-PapersOnLine*, 55(6):661–666, 2022. 11th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2022.

[78] D. Garcia-Gonzalez, D. Rivero, E. Fernandez-Blanco, and M. R. Luaces. New machine learning approaches for real-life human activity recognition using smartphone sensor-based data. *Knowledge-Based Systems*, page 110260, 2023.

[79] T. Gemini", R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[80] D. Gholamiangonabadi, N. Kiselov, and K. Grolinger. Deep neural networks for human activity recognition with wearable sensors: Leave-one-subject-out cross-validation for model selection. *IEEE Access*, 8:133982–133994, 2020.

[81] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[82] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1263–1272. JMLR.org, 2017.

[83] O. Giustolisi, L. Ridolfi, and A. Simone. Tailoring centrality metrics for water distribution networks. *Water Resources Research*, 55(3):2348–2369, 2019.

[84] O. Giustolisi, A. Simone, and L. Ridolfi. Network structure classification and features of water distribution systems. *Water Resources Research*, 53(4):3407–3423, 2017.

[85] O. Giustolisi and T. M. Walski. Demand components in water distribution network analysis. *Journal of Water Resources Planning and Management*, 138(4):356–367, 2012.

[86] R. Gómez Ramos, J. Duque Domingo, E. Zalama, and J. Gómez-García-Bermejo. Daily human activity recognition using non-intrusive sensors. *Sensors*, 21(16):5270, 2021.

[87] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[88] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005.

[89] W. Guo, S. Yamagishi, and L. Jing. Human activity recognition via wi-fi and inertial sensors with machine learning. *IEEE Access*, 12:18821–18836, 2024.

[90] S. Gupta. Deep learning based human activity recognition (har) using wearable sensor data. *Int. Journal of Information Management Data Insights*, 1(2):100046, 2021.

[91] S. Hajebi, E. Roshani, N. Cardozo, S. Barrett, A. Clarke, and S. Clarke. Water distribution network sectorisation using graph theory and many-objective optimisation. *Journal of Hydroinformatics*, 18(1):77–95, 2015.

[92] G. Hajgató, B. Gyires-Tóth, and G. Paál. Reconstructing nodal pressures in water distribution systems with graph neural networks. *arXiv preprint arXiv:2104.13619*, 2021.

[93] G. Hajgató, G. Paál, and B. Gyires-Tóth. Deep reinforcement learning for real-time optimization of pumps in water distribution systems. *Journal of Water Resources Planning and Management*, 146(11):04020079, 2020.

[94] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. *arXiv preprint arXiv:1706.05674*, 2017.

[95] W. L. Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[96] W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, pages 1024–1034, 2017.

[97] N. Y. Hammerla, S. Halloran, and T. Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*, 2016.

[98] N. Y. Hammerla and T. Plötz. Let's (not) stick together: pairwise similarity biases cross-validation in activity recognition. In *Proc. 2015 ACM international joint conference on pervasive and ubiquitous computing*, pages 1041–1051, 2015.

[99] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[100] J. F. Hemphill. Interpreting the magnitudes of correlation coefficients. 2003.

[101] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[102] E. Hernadez, S. Hoagland, and L. Ormsbee. *Water Distribution Database for Research Applications*, pages 465–474. 2016.

[103] C. Hu, L. Cheng, J. Sepulcre, K. A. Johnson, G. E. Fakhri, Y. M. Lu, and Q. Li. A spectral graph regression model for learning brain connectivity of alzheimer's disease. *PloS one*, 10(5):e0128136, 2015.

[104] P. Hu, J. Tong, J. Wang, Y. Yang, and L. de Oliveira Turci. A hybrid model based on cnn and bi-lstm for urban water demand prediction. In *2019 IEEE Congress on evolutionary computation (CEC)*, pages 1088–1094. IEEE, 2019.

[105] W. Huang, L. Zhang, W. Gao, F. Min, and J. He. Shallow convolutional neural networks for human activity recognition using wearable sensors. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2021.

[106] W. Huang, L. Zhang, H. Wu, F. Min, and A. Song. Channel-equalization-har: a light-weight convolutional neural network for wearable sensor based human activity recognition. *IEEE Transactions on Mobile Computing*, 2022.

[107] A. Ignatov. Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915–922, 2018.

[108] O. Iwakin and F. Moazeni. Improving urban water demand forecast using conformal prediction-based hybrid machine learning models. *Journal of Water Process Engineering*, 58:104721, 2024.

[109] A. R. Javed, R. Faheem, M. Asim, T. Baker, and M. O. Beg. A smartphone sensors-based personalized human activity recognition system for sustainable smart cities. *Sustainable Cities and Society*, 71:102970, 2021.

[110] W. Jiang and J. Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, 2022.

[111] A. Jordao, A. C. Nazare Jr, J. Sena, and W. R. Schwartz. Human activity recognition based on wearable sensor data: A standardization of the state-of-the-art. *arXiv preprint arXiv:1806.05226*, 2018.

[112] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[113] S. Jun and K. E. Lansey. Convolutional neural network for burst detection in smart water distribution systems. *Water Resources Management*, 37(9):3729–3743, 2023.

[114] D. Kang and K. Lansey. Real-time demand estimation and confidence limit analysis for water distribution systems. *Journal of Hydraulic Engineering*, 135(10):825–837, 2009.

[115] B. Kerimov, R. Bentivoglio, A. Garzón, E. Isufi, F. Tscheikner-Gratl, D. B. Steffelbauer, and R. Taormina. Assessing the performances and transferability of graph neural network metamodels for water distribution systems. *Journal of Hydroinformatics*, 25(6):2223–2234, 2023.

[116] S. Khalifa, G. Lan, M. Hassan, A. Seneviratne, and S. K. Das. Harke: Human activity recognition from kinetic energy harvesting data in wearable devices. *IEEE Transactions on Mobile Computing*, 17(6):1353–1368, 2017.

[117] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[118] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[119] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[120] K. A. Klise, R. Murray, and T. Haxton. An overview of the water network tool for resilience (wntr). In *Proceedings of the 1st International WDSA/CCWI Joint Conference, Kingston, Ontario, Canada*. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2018.

[121] M. M. Koşucu, E. Albay, and M. C. Demirel. Extending epanet hydraulic solver capacity with rigid water column global gradient algorithm. *Journal of Hydro-environment Research*, 42:31–43, 2022.

[122] N. Krishnan and D. Cook. Activity recognition on streaming sensor data. *Pervasive and mobile computing*, 10:138–154, 2014.

[123] S. M. Kumar, S. Narasimhan, and S. M. Bhallamudi. State estimation in water distribution networks using graph-theoretic reduction strategy. *Journal of Water Resources Planning and Management*, 134(5):395–403, 2008.

[124] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[125] C. Lamnatou, D. Chemisana, and C. Cristofari. Smart grids and smart technologies in relation to photovoltaics, storage systems, buildings and the environment. *Renewable Energy*, 185:1376–1391, 2022.

[126] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[127] H. Lee and K. Yoon. Towards better generalization with flexible representation of multi-module graph neural networks. *arXiv preprint arXiv:2209.06589*, 2022.

[128] D. R. Legates and G. J. McCabe Jr. Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation. *Water Resources Research*, 35(1):233–241, 1999.

[129] J. Li, Z. Wang, Z. Zhao, Y. Jin, J. Yin, S.-L. Huang, and J. Wang. Tribogait: A deep learning enabled triboelectric gait sensor system for human activity recognition and individual identification. In *Adjunct Proc. of the 2021 ACM Int. Joint conf. on Pervasive and Ubiquitous Computing and Proc. of the 2021 ACM Int. Symposium on Wearable Computers*, pages 643–648, 2021.

[130] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian. Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):3316–3333, 2021.

[131] Y. Li, J. Gao, C. Shen, Y. Guan, and W. Wang. Estimation of leak area-pressure relationship for cracks on water pipes using models based on linear-elastic fracture mechanics. *Water Research*, 221:118692, 2022.

[132] Z. Li, H. Liu, C. Zhang, and G. Fu. Gated graph neural networks for identifying contamination sources in water distribution systems. *Journal of Environmental Management*, 351:119806, 2024.

[133] Z. Li, H. Liu, C. Zhang, and G. Fu. Real-time water quality prediction in water distribution networks using graph neural networks with sparse monitoring data. *Water Research*, 250:121018, 2024.

[134] G. M. Lima, B. M. Brentan, D. Manzi, and E. Luvizotto Jr. Metamodel for nodal pressure estimation at near real-time in water distribution systems using artificial neural networks. *Journal of Hydroinformatics*, 20(2):486–496, 2018.

[135] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[136] F. Luo, S. Khan, Y. Huang, and K. Wu. Binarized neural network for edge intelligence of sensor-based human activity recognition. *IEEE Transactions on Mobile Computing*, 22(3):1356–1368, 2023.

[137] F. Luo, S. Poslad, and E. Bodanese. Kitchen activity detection for healthcare using a low-power radar-enabled sensor network. In *IEEE Int. conf. on Communications (ICC)*, pages 1–7. IEEE, 2019.

[138] J. Luo, K. Ying, and J. Bai. Savitzky–golay smoothing and differentiation filter for even number data. *Signal processing*, 85(7):1429–1434, 2005.

[139] S. Mahmud, M. Tanjid Hasan Tonmoy, K. Kumar Bhaumik, A. Mahbubur Rahman, M. Ashraful Amin, M. Shoyaib, M. Asif Hossain Khan, and A. Ahsan Ali. Human activity recognition from wearable sensor data using self-attention. In *ECAI 2020*, pages 1332–1339. IOS Press, 2020.

[140] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi. Protecting sensory data against sensitive inferences. In *Proc. of the 1st Workshop on Privacy by Design in Distributed Systems*, pages 1–6, 2018.

[141] H. Mao, Z. Chen, W. Tang, J. Zhao, Y. Ma, T. Zhao, N. Shah, M. Galkin, and J. Tang. Position: Graph foundation models are already here. In *Forty-first International Conference on Machine Learning*, 2024.

[142] F. Martínez, V. Hernández, J. M. Alonso, Z. Rao, and S. Alvisi. Optimizing the operation of the Valencia water-distribution network. *Journal of Hydroinformatics*, 9(1):65–78, 2007.

[143] G. Meirelles, D. Manzi, B. Brentan, T. Goulart, and E. Luvizotto. Calibration model for water distribution network using pressures estimated by artificial neural networks. *Water Resources Management*, 31:4339–4351, 2017.

[144] S. Mekruksavanich and A. Jitpattanakul. Lstm networks using smartphone data for sensor-based human activity recognition in smart homes. *Sensors*, 21(5):1636, 2021.

[145] M. C. Melo, J. R. Maasch, and C. de la Fuente-Nunez. Accelerating antibiotic discovery through artificial intelligence. *Communications biology*, 4(1):1050, 2021.

[146] A. Menapace, D. Avesani, M. Righetti, A. Bellin, and G. Pisaturo. Uniformly distributed demand epanet extension. *Water resources management*, 32:2165–2180, 2018.

[147] A. Menapace, A. Zanfei, M. Felicetti, D. Avesani, M. Righetti, and R. Gargano. Burst detection in water distribution systems: The issue of dataset collection. *Applied Sciences*, 10(22), 2020.

[148] D. Micucci, M. Mobilio, and P. Napoletano. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10):1101, 2017.

[149] A. Mohamed, F. Lejarza, S. Cahail, C. Claudel, and E. 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 335–340. IEEE, 2022.

[150] G. Mohmed, A. Lotfi, and A. Pourabdollah. Employing a deep convolutional neural network for human activity recognition based on binary ambient sensor data. In *Proceedings of the 13th ACM international conference on pervasive technologies related to assistive environments*, pages 1–7, 2020.

[151] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[152] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proc. of the AAAI conf. on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[153] R. Mutegeki and D. S. Han. A cnn-lstm approach to human activity recognition. In *Int. conf. on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 362–366. IEEE, 2020.

[154] N. T. Mücke, P. Pandey, S. Jain, S. M. Bohté, and C. W. Oosterlee. A probabilistic digital twin for leak localization in water distribution networks using generative deep learning. *Sensors*, 23(13), 2023.

[155] D. Nerantzis, F. Pecci, and I. Stoianov. Optimal control of water distribution networks without storage. *European Journal of Operational Research*, 284(1):345–354, 2020.

[156] M. E. Newman. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Physical Review E*, 94(5):052315, 2016.

[157] E. Nguyen, M. Poli, M. G. Durrant, B. Kang, D. Katrekar, D. B. Li, L. J. Bartie, A. W. Thomas, S. H. King, G. Brixi, et al. Sequence modeling and design from molecular to genome scale with evo. *Science*, 386(6723):eado9336, 2024.

[158] T. Nguyen, H. Le, T. P. Quinn, T. Nguyen, T. D. Le, and S. Venkatesh. GraphDTA: predicting drug–target binding affinity with graph neural networks. *Bioinformatics*, 37(8):1140–1147, 2020.

[159] Q. Ni, Z. Fan, L. Zhang, C. D. Nugent, I. Cleland, Y. Zhang, and N. Zhou. Leveraging wearable sensors for human daily activity recognition with stacked denoising autoencoders. *Sensors*, 20(18):5114, 2020.

[160] F. J. Ordóñez and D. Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[161] A. Ostfeld, E. Salomons, L. Ormsbee, J. G. Uber, C. M. Bros, P. Kalungi, R. Burd, B. Zazula-Coetzee, T. Belrain, D. Kang, et al. Battle of the water calibration networks. *Journal of water resources planning and management*, 138(5):523–532, 2012.

[162] D. Paez and Y. Filion. Generation and validation of synthetic wds case studies using graph theory and reliability indexes. *Procedia Engineering*, 186:143–151, 2017. XVIII International Conference on Water Distribution Systems, WDSA2016.

[163] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[164] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[165] W. Peng, X. Hong, H. Chen, and G. Zhao. Learning graph convolutional network for skeleton-based human action recognition by neural searching. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2669–2676, 2020.

[166] L. Perelman and A. Ostfeld. Topological clustering for water distribution systems analysis. *Environmental Modelling & Software*, 26(7):969–972, 2011.

[167] J. E. Pesantez, F. Alghamdi, S. Sabu, G. Mahinthakumar, and E. Z. Berglund. Using a digital twin to explore water infrastructure impacts during the covid-19 pandemic. *Sustainable Cities and Society*, 77:103520, 2022.

[168] Z. Pu, J. Yan, L. Chen, Z. Li, W. Tian, T. Tao, and K. Xin. A hybrid wavelet-cnn-lstm deep learning model for short-term urban water demand forecasting. *Frontiers of Environmental Science & Engineering*, 17(2):22, 2023.

[169] L. Qiao, H. Zhang, M. Kim, S. Teng, L. Zhang, and D. Shen. Estimating functional brain networks by incorporating a modularity prior. *Neuroimage*, 141:399–407, 2016.

[170] B. Quigley, M. Donnelly, G. Moore, and L. Galway. A comparative analysis of windowing approaches in dense sensing environments. *Multidisciplinary Digital Publishing Institute Proc.*, 2(19):1245, 2018.

[171] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[172] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR, 2023.

[173] J. Reca and J. Martínez. Genetic algorithms for the design of looped irrigation water distribution networks. *Water resources research*, 42(5), 2006.

[174] P. Reiser, M. Neubert, A. Eberhard, L. Torresi, C. Zhou, C. Shao, H. Metni, C. van Hoesel, H. Schopmans, T. Sommer, et al. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93, 2022.

[175] A. Reiss and D. Stricker. Introducing a new benchmarked dataset for activity monitoring. In *16th int. symp. on wearable computers*, pages 108–109. IEEE, 2012.

[176] R. Riesebos, V. Degeler, and A. Tello. Smartphone-based real-time indoor positioning using ble beacons. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 1281–1288. IEEE, 2022.

[177] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, et al. Collecting complex activity datasets in highly rich networked sensor environments. In *2010 7th int. conf. on networked sensing systems (INSS)*, pages 233–240. IEEE, 2010.

[178] L. A. Rossman. The epanet programmer's toolkit for analysis of water distribution systems. In *WRPMD'99: Preparing for the 21st Century*, pages 1–10. 1999.

[179] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[180] E. Ruiz, S. Díaz, and J. González. Potential performance of hydraulic state estimation in water distribution networks. *Water Resources Management*, pages 1–18, 2022.

[181] R. San-Segundo, H. Blunck, J. Moreno-Pimentel, A. Stisen, and M. Gil-Martín. Robust human activity recognition using smartwatches and smartphones. *Engineering Applications of Artificial Intelligence*, 72:190–202, 2018.

[182] D. Sánchez, M. Tentori, and J. Favela. Activity recognition for the smart hospital. *IEEE intelligent systems*, 23(2):50–57, 2008.

[183] A. Savitzky and M. J. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[184] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[185] A. Shehzed, A. Jalal, and K. Kim. Multi-person tracking in smart surveillance system for crowd counting and normal/abnormal events detection. In *2019 International Conference on Applied and Engineering Mathematics (ICAEM)*, pages 163–168, 2019.

[186] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7912–7921, 2019.

[187] J. Shlomi, P. Battaglia, and J.-R. Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

[188] A. Simpson and S. Elhay. Jacobian matrix for solving water distribution system equations with the darcy-weisbach head-loss model. *Journal of Hydraulic Engineering*, 137(6):696–700, 2011.

[189] R. Sitzenfrei. Using complex network analysis for water quality assessment in large water distribution systems. *Water Research*, 201:117359, 2021.

[190] R. Sitzenfrei, M. Hajibabaei, S. Hesarkazzazi, and K. Diao. Dual graph characteristics of water distribution networks—how optimal are design solutions? *Complex & Intelligent Systems*, 9(1):147–160, 2023.

[191] R. Sitzenfrei, Q. Wang, Z. Kapelan, and D. Savić. Using complex network analysis for optimization of water distribution networks. *Water resources research*, 56(8):e2020WR027929, 2020.

[192] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE transactions on neural networks*, 8(3):714–735, 1997.

[193] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proc. of the 13th ACM conf. on embedded networked sensor systems*, pages 127–140, 2015.

[194] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

[195] M. Sypetkowski, F. Wenkel, F. Poursafaei, N. Dickson, K. Suri, P. Fradkin, and D. Beaini. On the scalability of gnns for molecular graphs. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 19870–19906. Curran Associates, Inc., 2024.

[196] M. Tan and Q. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[197] J. Tang, J. Li, Z. Gao, and J. Li. Rethinking graph neural networks for anomaly detection. In *International conference on machine learning*, pages 21076–21089. PMLR, 2022.

[198] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, A. Ostfeld, D. G. Eliades, M. Aghashahi, R. Sundararajan, M. Pourahmadi, M. K. Banks, B. M. Brentan, E. Campbell, G. Lima, D. Manzi, D. Ayala-Cabrera, M. Herrera, I. Montalvo, J. Izquierdo, E. Luvizotto, S. E. Chandy, A. Rasekh, Z. A. Barker, B. Campbell, M. E. Shafiee, M. Giacomoni, N. Gatsis, A. Taha, A. A. Abokifa, K. Haddad, C. S. Lo, P. Biswas, M. F. K. Pasha, B. Kc, S. L. Somasundaram, M. Housh, and Z. Ohar. Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks. *Journal of Water Resources Planning and Management*, 144(8):04018048, 2018.

[199] R. Taylor. Interpretation of the correlation coefficient: a basic review. *Journal of diagnostic medical sonography*, 6(1):35–39, 1990.

[200] A. Tello and V. Degeler. Digital twins: an enabler for digital transformation. In *The Digital Transformation handbook*. 2021.

[201] A. Tello and V. Degeler. Contrasting global and local representations for human activity recognition using graph neural networks. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing (SAC'25)*, 2025.

[202] A. Tello, V. Degeler, and A. Lazovik. Too good to be true: accuracy overestimation in (re) current practices for human activity recognition. In *IEEE Int. Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 511–517. IEEE, 2024.

[203] A. Tello, H. Truong, A. Lazovik, and V. Degeler. Large-scale multipurpose benchmark datasets for assessing data-driven deep learning approaches for water distribution networks. *Engineering Proceedings*, 69(1):50, 2024.

[204] A. Tello, H. van het Loo, A. Lazovik, and V. Degeler. Towards Graph Foundation Models for Water Distribution Networks. In the 21st. International Computing & Control in the Water Industry Conference (CCWI2025), 2025.

[205] H. R. Tiedmann, L. A. Spearing, L. Sela, K. Kinney, M. J. Kirisits, L. E. Katz, J. Kaminsky, and K. M. Faust. Modeling in the covid-19 pandemic: Overcoming the water sector's data struggles to realize the potential of hydraulic models. *Journal of Water Resources Planning and Management*, 148(6):05022003, 2022.

[206] E. Todini, S. Santopietro, R. Gargano, and L. A. Rossman. Pressure flow–based algorithms for pressure-driven analysis of water distribution networks.

Journal of Water Resources Planning and Management, 147(8):04021048, 2021.

[207] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[208] V. Traag, L. Waltman, and N. Van Eck. From louvain to leiden: guaranteeing well-connected communities. sci. rep. 9, 5233, 2019.

[209] H. Truong, A. Tello, A. Lazovik, and V. Degeler. Graph neural networks for pressure estimation in water distribution systems. *Water Resources Research*, 60(7):e2023WR036741, 2024.

[210] H. Truong, A. Tello, A. Lazovik, and V. Degeler. Ditec-wdn: A large-scale dataset of hydraulic scenarios across multiple water distribution networks. *Scientific Data*, 12(1):1733, 2025.

[211] L. Tsiami and C. Makropoulos. Cyber—physical attack detection in water distribution systems with temporal graph convolutional neural networks. *Water*, 13(9):1247, 2021.

[212] T. Van Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *Proc. of the 10th int. conf. on Ubiquitous computing*, pages 1–9, 2008.

[213] J. E. Van Zyl. *A methodology for improved operational optimization of water distribution systems*. Dataset, University of Exeter UK, 2001.

[214] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[215] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster.

[216] Vewin. Dutch drinking water statistics 2015., 2015.

[217] Vewin. Dutch drinking water statistics 2017 from source to tap. "`https://www.vewin.nl/wp-content/uploads/2024/08/Drinkwaterstatistieken-2017-EN.pdf`", 2017. "Online; accessed 05-March-2025".

[218] Vewin. Dutch drinking water statistics 2022 from source to tap. `https://www.vewin.nl/wp-content/uploads/2024/06/vewin-dutch-drinking-water-statistics-2022-eng-web.pdf`, 2022. "Online; accessed 05-March-2025".

[219] S. G. Vrachimis, D. G. Eliades, R. Taormina, Z. Kapelan, A. Ostfeld, S. Liu, M. Kyriakou, P. Pavlou, M. Qiu, and M. M. Polycarpou. Battle of the leakage detection and isolation methods. *Journal of Water Resources Planning and Management*, 148(12):04022068, 2022.

[220] S. G. Vrachimis, M. S. Kyriakou, et al. Leakdb: a benchmark dataset for leakage diagnosis in water distribution networks:(146). In *WDSA/CCWI joint conference proceedings*, volume 1, 2018.

[221] T. M. Walski, E. D. Brill Jr, J. Gessler, I. C. Goulter, R. M. Jeppson, K. Lansey, H.-L. Lee, J. C. Liebman, L. Mays, D. R. Morgan, et al. Battle of the network models: Epilogue. *Journal of Water Resources Planning and Management*, 113(2):191–203, 1987.

[222] S. Wan, L. Qi, X. Xu, C. Tong, and Z. Gu. Deep learning models for real-time human activity recognition with smartphones. *Mobile Networks and Applications*, 25(2):743–755, 2020.

[223] J. Wang, T. Zhu, J. Gan, L. L. Chen, H. Ning, and Y. Wan. Sensor data augmentation by resampling in contrastive learning for human activity recognition. *IEEE Sensors Journal*, 22(23):22994–23008, 2022.

[224] S. Wang, A. F. Taha, N. Gatsis, L. Sela, and M. H. Giacomoni. Probabilistic state estimation in water networks. *IEEE Transactions on Control Systems Technology*, 30(2):507–519, 2021.

[225] S. Wang, G. Zhou, Y. Ma, L. Hu, Z. Chen, Y. Chen, H. Zhao, and W. Jung. Eating detection and chews counting through sensing mastication muscle contraction. *Smart Health*, 9:179–191, 2018.

[226] T. Wang and D. J. Cook. Multi-person activity recognition in continuously monitored smart homes. *IEEE Transactions on Emerging Topics in Computing*, 2021.

[227] Y. Wang, X. Wang, H. Yang, Y. Geng, H. Yu, G. Zheng, and L. 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*, 2023.

[228] G. M. Weiss, K. Yoneda, and T. Hayajneh. Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access*, 7:133190–133202, 2019.

[229] F. Wong, E. J. Zheng, J. A. Valeri, N. M. Donghia, M. N. Anahtar, S. Omori, A. Li, A. Cubillos-Ruiz, A. Krishnan, W. Jin, et al. Discovery of a structural class of antibiotics with explainable deep learning. *Nature*, 626(7997):177–185, 2024.

[230] H. Wu, Z. Zhang, X. Li, K. Shang, Y. Han, Z. Geng, and T. Pan. A novel pedal musculoskeletal response based on differential spatio-temporal lstm for human activity recognition. *Knowledge-Based Systems*, 261:110187, 2023.

[231] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[232] L. Xiao, A. Zhang, B. Cai, J. M. Stephen, T. W. Wilson, V. D. Calhoun, and Y.-P. Wang. Correlation guided graph learning to estimate functional connectivity patterns from fmri data. *IEEE Transactions on Biomedical Engineering*, 68(4):1154–1165, 2020.

[233] L. Xing and L. Sela. Graph neural networks for state estimation in water distribution systems: Application of supervised and semisupervised learning. *Journal of Water Resources Planning and Management*, 148(5):04022018, 2022.

[234] B. Xu, H. Shen, B. Sun, R. An, Q. Cao, and X. Cheng. Towards consumer loan fraud detection: Graph neural networks with role-constrained conditional random field. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4537–4545, 2021.

[235] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[236] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[237] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *32nd AAAI conference on artificial intelligence*, 2018.

[238] Y. Yan, T. Liao, J. Zhao, J. Wang, L. Ma, W. Lv, J. Xiong, and L. Wang. Deep transfer learning with graph neural network for sensor-based human activity recognition. *arXiv preprint arXiv:2203.07910*, 2022.

[239] G. Yehudai, E. Fetaya, E. Meirom, G. Chechik, and H. Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR, 2021.

[240] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems.

In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 974–983, New York, NY, USA, 2018. Association for Computing Machinery.

[241] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[242] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[243] W. Yu, Z. Zhang, and Z. Qin. Low-pass graph convolutional network for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8954–8961, 2022.

[244] H. Yuan, S. Chan, A. P. Creagh, C. Tong, A. Acquah, D. A. Clifton, and A. Doherty. Self-supervised learning for human activity recognition using 700,000 person-days of wearable data. *NPJ digital medicine*, 7(1):91, 2024.

[245] A. Zanfei, B. M. Brentan, A. Menapace, M. Righetti, and M. Herrera. Graph convolutional recurrent neural networks for water demand forecasting. *Water Resources Research*, 58(7):e2022WR032299, 2022.

[246] A. Zanfei, A. Menapace, B. M. Brentan, M. Righetti, and M. Herrera. Novel approach for burst detection in water distribution systems based on graph neural networks. *Sustainable Cities and Society*, 86:104090, 2022.

[247] A. Zanfei, A. Menapace, B. M. Brentan, R. Sitzenfrei, and M. Herrera. Shall we always use hydraulic models? a graph neural network metamodel for water system calibration and uncertainty assessment. *Water Research*, 242:120264, 2023.

[248] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

[249] C. Zhang, K. Cao, L. Lu, and T. Deng. A multi-scale feature extraction fusion model for human activity recognition. *Scientific Reports*, 12(1):20620, 2022.

[250] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[251] L. Zhang, J. Yu, Z. Gao, and Q. Ni. A multi-channel hybrid deep learning framework for multi-sensor fusion enabled human activity recognition. *Alexandria Engineering Journal*, 91:472–485, 2024.

[252] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[253] M. Zhang and A. A. Sawchuk. Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proc. ACM conf. on ubiquitous computing*, pages 1036–1043, 2012.

[254] S. Zhang, H. Tong, J. Xu, and R. Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[255] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, X. Xie, and J. Tang. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*, 2023.

[256] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 21(9):3848–3858, 2019.

[257] X. Zheng, M. Wang, and J. Ordieres-Meré. Comparison of data preprocessing approaches for applying deep learning to human activity recognition in the context of industry 4.0. *Sensors*, 18(7):2146, 2018.

[258] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[259] X. Zhou, Z. Tang, W. Xu, F. Meng, X. Chu, K. Xin, and G. Fu. Deep learning identifies accurate burst locations in water distribution networks. *Water research*, 166:115058, 2019.

[260] X. Zhou, J. Zhang, S. Guo, S. Liu, and K. Xin. A convenient and stable graph-based pressure estimation methodology for water distribution networks: Development and field validation. *Water Research*, 233:119747, 2023.

[261] Q. Zhu, C. Yang, Y. Xu, H. Wang, C. Zhang, and J. Han. Transfer learning of graph neural networks with ego-graph information maximization. *Advances in Neural Information Processing Systems*, 34:1766–1779, 2021.

[262] E. G. Zimbelman and R. F. Keefe. Development and validation of smartwatch-based activity recognition models for rigging crew workers on cable logging operations. *Plos one*, 16-5:e0250624, 2021.