Physics Informed Neural Networks: Deployment and Evaluation in Sparse Data Applications

Dinh Viet Cuong, B.Sc.

Supervised by Prof Mark Roantree

A thesis presented for the degree of Doctor of Philosophy

School of Computing

Dublin City University

March 14, 2025

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, and that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed (Typed Name): Dinh Viet Cuong

Student ID No.: 20214134

Date: 05/12/2024

Acknowledgements

Four years have passed in the blink of an eye. This thesis would not have been possible without the encouragement, advice, and support of my advisors, friends, and family.

Firstly, I would like to thank my advisor, Professor Mark Roantree, for his guidance, which was essential to the completion of my thesis. I am grateful for his patience, trust, and attention during my doctoral studies. Throughout these challenging years, he granted me the freedom to pursue a variety of interesting research topics. His constructive feedback on my work taught me how to conduct meaningful research and greatly improved my critical thinking abilities.

I would also like to thank my colleagues—Dermot, Adam, Valerio, Vuong, Aidan, Danham, and Uttran—for their collaboration and assistance in my studies, and for their help in proofreading my thesis.

I wish to express my gratitude to my friends in Ireland—An, Phuc, Boi, Le, Long, Bao, Khoi, Tu, and Khiem—for their friendship and support. An, for always being by my side while working. Phuc, for being a gym companion and for our shared interests in video games. Boi, for generously cooking delicious meals for me in three years. Khoi, Le, Long, and Bao, for the enjoyable times we shared. Tu and Khiem, for your assistance in Ireland. My life here would have been very dull without your friendship.

I would like to thank my friends Phuc Hau, Thanh Nhan and Thanh Huyen. Thank you, Phuc Hau and Thanh Nhan, for playing video games with me during the most stressful time of my life during Covid. Phuc Hau is a brilliant guy; not only

did we study mathematics together, but I also appreciate his insights into Buddhism. Thanh Nhan is a good travel companion and a good Chinese teacher. I would also like to thank Hieu and Quoc for the little chats here and there.

I want to express my deep gratitude to my family, who have provided unwavering support and constant encouragement throughout my life. My father is a gentleman who has always been intelligent but had no opportunity to pursue higher education. My mother is a hardworking woman who had to drop out of school at a very early age to make a living. They have always believed that studying and knowledge are the keys to becoming a good person and achieving success in life. I hope that the studies I have undertaken so far will partly fulfill their dream of learning. Additionally, my sister has always supported me throughout my life. I cannot imagine my life without my family.

Lastly, I would like to acknowledge School of Computing at Dublin City University for providing a great environment to work in, and the financial support of Taighde Éireann - Research Ireland through the Insight Centre for Data Analytics (SFI/12/RC/2289_P2).

I dedicate this thesis

to my father, Dinh Viet Quang,

to my mother, Nguyen Thi Huong,

who always support me in my life;

to science, and to my own dream.

Contents

List of Figures

List of Tables

Physics Informed Neural Networks

Deployment and Evaluation in Sparse Data Applications

Dinh Viet Cuong

Abstract

Neural networks have demonstrated remarkable success in various domains but they often struggle with generalization beyond their training data. To address these limitations and enhance the robustness of machine learning models, this thesis explores the integration of domain knowledge into neural networks through two approaches, network analysis and ordinary differential equations (ODEs). We begin by investigating neural network performance in diverse tasks, such as hyperglycemia/hypoglycemia diagnosis using exosome profiles and oxygen uptake estimation from sensor measurements. The study then progresses to more structured data with complex networks.

Subsequently, we incorporate network structure into machine learning using graph neural networks, applying this method to an air quality forecasting task where locations and their correlations form a network. An alternative approach is then investigated by integrating ODE systems describing dynamical systems into a data-driven machine learning framework. This comprises the development of advanced techniques to enable neural networks to learn underlying physics, including ODE Normalization, Gradient Balancing, Causal Training, and Domain Decomposition. These methods address challenges in training with stiff systems across large domains.

The frameworks in this research are then validated using simulated data for the Lorenz system and a system of ODEs modelling mosquito populations. This work is further developed to accommodate real-life observations, by making adjustments to model inputs, neural network architecture, and activation functions. This extended framework is then evaluated against real-world mosquito counts in an

inverse problem setting, learning relationships between meteorological conditions and mosquito development. Our results demonstrate that incorporating domain knowledge into neural networks enhances model generalizability, improving both accuracy and extrapolation capabilities. Moreover, this approach maintains the explainability of the added knowledge while leveraging the flexibility of machine learning models.

Chapter 1

Introduction

This research dissertation explores the use of machine learning, principally neural networks in solving problems across real world application areas. It begins as a journey to understand the strengths of these machine learning functions, identifying weaknesses in the face of challenges presented by practical application areas such as health, climate and sport. Often, these challenges result from the lack of sufficient training data, an issue which is more commonplace than was expected when this research was undertaken.

In this opening chapter, we provide an introduction and motivation for the different studies presented, focusing on the advancements and applications of neural networks across various domains. We begin with a brief overview of neural networks, describing their biological inspiration, fundamental structure and learning mechanisms. We then discuss the evaluation of architectures such as recurrent, convolutional and graph neural networks, as well as the integration of physics laws that describe a real-world phenomenon. We then outline the research questions that guide this research, including the incorporation of prior knowledge, graph-based modeling and the application of physics-informed neural networks for solving dynamical systems. Finally, we present an overview of the dissertation's structure and list of the publications associated with this research.

1.1 Introduction to Neural Networks and their Application Areas

A *neural network* [1] is a mathematical model inspired by the biological neural network in animals [2]. The computational unit of the model is a neuron. A neuron receives inputs, typically real-valued numbers, either from other neurons or from data representations. It processes these inputs to produce an output, also in the form of a real valued number. Each input is associated with a weight, which represents the importance of that input with respect to the overall output. The output of the neuron is computed as the weighted sum of its inputs. This weighted sum can optionally pass through an activation function, providing nonlinearity to the neuron. A neuron can be thought as a decision maker, making decisions by weighting evidence from various inputs. For example, consider deciding whether to go outside for a run. Factors such as the current temperature outside, the probability it will rain, or your energy level, the amount of time you have, all might play a role. Each factor is assigned a weight based on its importance with the final decision based on the combined information from these weighted factors. A single neuron is too simplistic to make complex decisions on its own, combining multiple neurons together make it possible to model more sophisticated decision making. Neurons are often organized into network structures, where each node represents a neuron and a directed edge indicates that the output of one neuron is the input to another. Usually organized into *layers*, these layers consists of neurons that share a similar functionality. Layers receive inputs from other layers and produce outputs that feed into other layers. *Multi-layer Perceptrons* (MLP), described in [2], organize the neurons in a feed-forward manner, with each layer being fully connected to both preceding and subsequent layers. Thus, the first layer makes decisions directly on the input data, while subsequent layers refine these decisions by building on outputs from previous layers. This arrangement facilitates the learning of increasingly abstract representations, ultimately enabling the modeling of complex decision-making processes.

A single neuron, as well as the entire neural network as a whole, can learn to make good decisions by adjusting the weight of each input, which are referred to the *trainable parameters* of the neural network. The learning process of a neural network can be performed automatically by a computer algorithm based on the information provided by a set of samples, known as the *training set*. This training is often formulated as an optimization problem, wherein an *objective function*, also called the *loss function*, is minimized. The objective function is designed to reflect reality, specifically matching the model with the training data or with specific domain knowledge. The most common approach to training neural networks today is based on the *gradient descent* method [3]. The method step-by-step updates the trainable parameters by moving in the direction opposite to the gradient of the objective function, which is the direction that reduces the loss.

The simplest form of neural network, *linear regression* (LR), was proposed as early as 1795 by Johann Gauss [4]. Inspired by the computational model of biological neurons proposed by [2], Frank Rosenblatt introduced the multi-layer perceptron model in 1958 [1]. Rosenblatt's MLP had the first hidden layer randomized and non-trainable, while the output layer was learnt. In 1967, [5] trained MLPs with stochastic gradient descent [6]. The algorithm updates model parameters based on random subsets of training data, allowing for efficient and scalable optimization. It remains one of the most widely used methods for training neural networks today. In 1970, Seppo Linnainmaa published the *back-propagation* algorithm, which provided an efficient mechanism for calculating derivatives required in gradient-based optimization methods [7]. Today, backpropagation is implemented widely in modern deep learning frameworks, such as PyTorch [8] and TensorFlow [9].

Neural networks are now a core part of artificial intelligence (AI), machine learning (ML), and particularly deep learning today. They serve as a central driving force behind various advancements. Many impactful applications use neural networks as their machine learning model, including recommendation system [10], face recognition [11], object recognition [12], image generation [13], speech recognition [14], speech

synthesis [15], AI assistant chatbots [16, 17], machine translation [18]. Neural networks have been successfully deployed across diverse fields, including healthcare [19], bioinformatics [20], finance [21], agriculture [22], climate science [23], and information technology [24], to name a few. In many cases, neural networks have outperformed traditional machine learning models, achieving state-of-the-art results [25].

1.1.1 Issues with Neural Networks

Despite their impressive success, neural networks are inherently limited by several challenges, particularly regarding the large volume of data required for effective training [26, 27] and their poor generalization capabilities [28]. The predictions made by a machine learning model can be considered as either interpolation, predicting within the range of the training data, or extrapolation, predicting beyond the distribution of the training samples. When sufficient training data is available, neural networks have a high capacity to interpolate these examples, leading to accurate decision-making [29]. However, when the data is sparse, meaning the unseen sample is likely to fall out of the training distribution, resulting in a poor generalization from the training biases and extrapolation. Furthermore, neural networks are frequently over-parameterized in order to achieve strong performance [29]. Over-parameterization lets these models learn highly complex mappings from the input space to the output space. While this enables neural networks to model complex relationships within the data, it also results in a very large hypothesis space, which allows them to fit almost any dataset [30], irrespective of how well the learned mapping aligns with the underlying nature of the data. This phenomenon, known as *overfitting*, degrades its ability to perform well on unseen data.

1.1.2 Neural Network Architectures

To address these limitations, researchers have made significant improvements for neural networks. The authors in [26, 31] identify several ways to incorporate prior knowledge into neural networks, such as enhancing the neural network architecture

and adding regularization terms in the loss function during training. The differences in neural architectures arise from the design of individual neurons, the arrangement of neurons within layers, and the overall arrangement of the layers themselves. These structural differences significantly impact the performance and suitability of neural network models for different tasks [32].

A neural network architecture that has been gaining considerable attention in recent years is the Graph Neural Network (GNN), which is specifically designed to manage the complexity of graph-structured data. A *graph*, also referred to as a *network*, consists of a set of nodes (or vertices) and a set of edges (or links) that establish connections between pairs of nodes. Typically, nodes represent entities of interest, such as locations, stations, airports, assets, etc., while edges denote relationships or connections between these nodes. The definition of these connections can be derived from domain-specific knowledge, including physical connections, semantic similarity, statistical correlations, and more. The idea behind GNN to handle graph structures is to replicate the graph's edges as connections among neurons or layers within. As a result, GNNs can make predictions at a given node by aggregating information from the node itself as well as from the nodes to which it is connected. This aggregation mechanism allows GNNs to model interactions between nodes and to capture complex relationships in the data, thus making GNNs suitable to model structured data. In recent years, GNNs have emerged as a powerful machine learning model capable of learning from complex network systems, as well as network representations of flat data. The capabilities of GNN have been shown in many areas, including social networks [33], transportation networks [34], chemistry [35], climate [36], computer vision [37], and natural language processing [38].

1.1.3 Physics-Informed Neural Networks

Another effective approach to improving the performance of neural networks is by improving the objective function, specifically through the addition of regularization terms to the loss function. This addition encourages the model to converge to more

preferable solutions, ultimately improving its performance. Regularization terms can be utilized to incorporate scientific knowledge of the underlying processes into the model. Domain experts often have prior knowledge about the relationships present in the data, usually expressed in the form of mathematical equations. By incorporating these equations into the objective function as additional regularization terms, the model can be informed to learn solutions that are not only data-driven but also consistent with established domain-specific knowledge. This approach is often referred to as *physics-informed neural networks* (PINNs). For instance, [39] incorporates the conservation laws of mass and momentum, along with equations describing the pressure-area relationship, to predict blood pressure in arterial networks. Similarly, [40] used conservation laws to achieve energy consistency when emulating climate models with neural networks. By embedding such physical laws into the loss function, the trained neural network's predictions become more physically consistent, even for extrapolated samples, thereby significantly improving the model's generalization capabilities. Moreover, the integration of domain knowledge reduces the reliance on large amounts of high-quality training data. With knowledge-based regularization, the model can be trained effectively with less data, which does not need to be of excellent quality [26]. This is particularly advantageous for applications where acquiring high-quality data is challenging or costly.

1.2 Problem Statement

Motivated by the challenges and approaches presented above, this study aims to extend the capabilities of neural networks for different tasks across heterogenos domains. Thus, we investigate the applicability of different neural networks to gain an understanding of their weaknesses for certain tasks but in particular, how these machine learning models can be customized to better suit the task with which they are faced. This will require a fairly broad set of investigations and analyses but in doing so, we are asking one fundamental question: *Can prior knowledge be incorporated into neural networks to improve their overall predictive accuracy?*

Hypothesis. We hypothesize that integrating prior knowledge into neural networks enhances their overall predictive performance, particularly in complex tasks where there is limited data available.

Complex tasks often involve high-dimensional inputs, dynamic decision-making processes, or relationships among entities, challenges that make simple models insufficient and can require advanced algorithms. Neural networks, which often contain tens of thousands of parameters, are powerful enough to handle a lot of complex tasks. However, when the dataset is small (e.g., fewer than 1,000 or 10,000 samples), they often memorize the limited data (overfitting) rather than generalize it. By incorporating domain knowledge, such as using graph representations to capture relationships among entities or regularizing models with physical constraints, neural networks can more effectively learn meaningful patterns while preserving physically consistent predictions.

This is particularly valuable for machine learning problems where data collection is expensive or time-consuming, limiting the available samples. In such cases, leveraging prior knowledge helps mitigate overfitting, reduces the need for massive datasets, and combines the power of data-driven models with the transparency and interpretability of physics-based models. To test this hypothesis and the ways it could be explored, we pose a series of research questions that will guide the direction of this study.

Research Question 1. *How effectively can generic neural networks be deployed as machine learning solutions in areas such as health and sports?*

In order to address this research question, we explore the capabilities of conventional neural network architectures in various application areas. We focus on problems that are new and challenging, such as exosome classification in health and oxygen uptake estimation in sports, where data is complex and limited. By applying neural networks to these areas, we aim to identify how well they perform and what limitations are exposed.

Research Question 2. *Can graph neural networks be deployed to exploit the structural information inherent in graph-based data?*

Graph models offer a powerful way to represent complex systems where entities are interconnected, capturing relationships that traditional flat data representations often overlook. In domains such as transportation and environmental science, the interactions between components may play critical roles. For example, in a bike-sharing system, the flow of bikes between stations forms a network that reflects user mobility patterns and demand, which cannot be fully understood by analyzing stations independently. Traditional neural networks struggle to capture these relationships because they are not designed to exploit graph-structured data. GNNs have emerged as a promising solution to this challenge, as they are specifically designed to process and learn from graph-based data by considering both the features of the nodes and the topology of the graph. By exploring this second research question, the goal is to enhance neural networks to enabling the modeling of graph data, ultimately leading to better predictive performance.

Research Question 3. *Can neural networks be extended to accurately represent and predict the behavior of dynamical systems governed by a system of ordinary differential equations?*

Purely data-driven neural networks often struggle to capture the underlying physical laws that govern the data, leading to limited generalizability and interpretability [28]. PINNs attempt to bridge this gap by embedding differential equations directly into the learning process. However, training PINNs presents significant challenges, especially when dealing with stiff equations and multi-scale dynamics common in real-world systems [26, 41, 42]. In this third research question, we would like to address the training difficulties of PINNs, improving their capability to solve systems of ordinary differential equations (ODEs). By overcoming these challenges, we can train data-driven models that better adhere to physics-based dynamical systems, particularly in complex, multi-scale scenarios like mosquito population dynamics.

Research Question 4. *Can we determine how neural networks incorporate external factors on dynamic system parameters and validate any solution using real-world observational data?*

While PINNs have shown significant potential in solving inverse problems [26, 43], they often neglect external factors that influence system dynamics. Accurate modeling of biological systems, such as mosquito populations, requires taking into account external influences such as meteorological conditions [44]. Incorporating external forcing factors, such as air humidity and precipitation, is crucial for enhancing model reliability and predictive accuracy [45–47]. Therefore, in this research question, the goal is to extend the PINN framework to learn the effects of external factors on dynamic system parameters directly from the data.

1.3 Thesis Structure

Throughout this dissertation, we present our work that addresses the posed research questions. These contributions are supported by the following list of publications:

1. (PUB1) *Exosomes Classification from Surfaced Enhanced Raman Spectroscopic Data Using a Multilayer Perception*
   Dinh Viet Cuong, John O'Sullivan, Nirod Kumar Sarang, Denise Burtenshaw, Paul Cahill, Tia E Keyes and Mark Roantree (ready for submission)

2. (PUB2) *Estimating Oxygen Uptake in Simulated Team Sports Using Machine Learning Models and Wearable Sensor Data*
   Dermot Sheridan, Arne Jaspers, **Dinh Viet Cuong**, Tim Op De Beéck, Niall M. Moyna, Toon T. de Beukelaar, Mark Roantree
   PLOS ONE, February 2024

3. (PUB3) *Analyzing Shared Bike Usage Through Graph-Based Spatio-Temporal Modeling*
   Dinh Viet Cuong, V. M. Ngo, P. Cappellari and M. Roantree,
   IEEE Open Journal of Intelligent Transportation Systems, vol. 5, pp. 115-131, 2024, doi: 10.1109/OJITS.2024.3350213

4. (PUB4) *Graph-Based Optimisation of Network Expansion in a Dockless Bike*

Sharing System

Mark Roantree, Niamh Murphy, **Dinh Viet Cuong**, Vuong M. Ngo

2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW), pp.48-55, 2024.

5. (PUB5) *Managing Large Dataset Gaps in Urban Air Quality Prediction: DCU-Insight-AQ at MediaEval 2022*

   Dinh Viet Cuong, Phuc H Le-Khac, Adam Stapleton, Elke Eichlemann, Mark Roantree, Alan F Smeaton

   MediaEval'22: Multimedia Evaluation Workshop, January 13–15, 2023, Bergen, Norway and Online, CEUR-WS Proceedings

6. (PUB6) *Adapting Physics-Informed Neural Networks to Improve ODE Optimization in Mosquito Population Dynamics*

   Dinh Viet Cuong, Branislava Lalić, Mina Petrić, Binh Nguyen, Mark Roantree

   PLoS ONE 19(12): e0315762. https://doi.org/10.1371/journal.pone.0315762

7. (PUB7) *Physics-Based Dynamic Models Hybridisation Using Physics-Informed Neural Networks*

   Dinh Viet Cuong*, Branislava Lalic*, Mina Petric, Vladimir Pavlovic, Ana Firanj Sremac, Mark Roantree (Submitted for publication.)

The remainder of the dissertation is structured as follows:

- In Chapter 2, we present a detailed overview of the studies that form this dissertation. This chapter covers introductions to exosome classification and oxygen uptake estimation, graph-based methodologies for analyzing bike-sharing system data, and air quality forecasting. Additionally, we discuss the current state of PINNs for both forward and inverse problems. A brief introduction to the datasets used in the experiments is also included.

- In Chapter 3, we provide a comprehensive literature review across multiple

domains relevant to this work. This includes the application of neural networks for exosome classification and oxygen uptake estimation, graph-based approaches for analyzing bike-sharing system data, graph neural networks for air quality forecasting, and related works of techniques for PINNs for forward and inverse problems.

- Chapter 4 addresses **Research Question 1**, focusing on the capability of neural networks to tackle novel machine learning tasks. We present two studies: exosome classification and oxygen uptake estimation. These works are documented in **PUB1** and **PUB2**, respectively.

- In Chapter 5, we model data using graphs for **Research Question 2**. The first part introduces a systematic framework for constructing and analyzing data using graph representations, published in **PUB3** and supported by **PUB4**. The second part presents a novel graph neural network approach for air quality prediction, supported by **PUB5**.

- Chapter 6 investigates **Research Question 3** which addresses difficulties in training PINNs. Enhanced techniques are proposed to improve convergence and accuracy when solving ODE systems, validated through the Lorenz system and a case study on mosquito population dynamics. These results are published in **PUB6**.

- Chapter 7 examines **Research Question 4**, which seeks to improve PINNs for parameter estimation in dynamical systems. The proposed method incorporates the effects of external forcing on system behavior and is evaluated using mosquito count observations. The proposed method is tested with observation data of mosquito counts. This work is detailed in **PUB7**.

- Finally, Chapter 8 summarizes the contributions of this dissertation and discusses its limitations and potential directions for future work.

Chapter 2

Problems & Datasets

In this chapter, we provide a detailed introduction to the projects explored in this dissertation, along with an overview of the datasets used in each study. We first present the application of neural networks to novel tasks, including exosome classification using surface-enhanced Raman spectroscopy and oxygen uptake estimation from wearable sensor data. Next, we introduce graph-based approaches, beginning with the construction and analysis of bike-sharing system data as graphs and then the development of graph neural networks with attention mechanisms. We then explore physics-informed neural networks, focusing on improving their training methodologies and validating them on mosquito population modeling. Finally, we discuss the application of PINNs to inverse problems in mosquito modeling, incorporating enhancements to account for external forcing factors.

2.1 Neural Networks

2.1.1 Exosome Classification

Exosomes are tiny particles released by all cells and found in all body fluids. They have shown to carry a proteomic profile that reflect the condition of their original cells, which means they can provide a lot of information about a person's health or disease state [48]. Because of their widespread presence in the body fluids and a close

relationship with their cells of origin, exosomes are becoming increasingly important for diagnosing and treating various conditions. For example, they have been used to discriminate between cancerous and non-cancerous cells, such as distinguishing exosomes from pancreatic, lung, and breast cancer cell lines [49–51]. Exosomes have also been used to evaluate disease progression [52].

Raman spectroscopy is a powerful tool for studying exosomes. It provides a detailed molecular fingerprint of the sample. When combined with advanced data analysis, Raman spectroscopy can provide valuable diagnostic output. To enhance the amplitude of the signal, surface enhanced Raman spectroscopy is widely used, particularly in bioanalysis [53, 54]. Using SERS to study exosomes has gained popularity over the past few years, especially when paired with data analysis methods that can effectively discriminate the signals. The initial focus of exosome analysis using SERS has been linear data analysis techniques such as principal component analysis (PCA) for classifying exosomes from various cellular origins. For example, studies by [49] and [50] demonstrated the use of PCA and variants to differentiate exosomes from pancreatic and lung cancer cell lines. However, linear methods often face limitations when the replationship is nonlinear.

An alternative approach, neural network-based models, have shown significant promise. The use of neural networks has been demonstrated to outperform traditional linear models, thanks to their superior capability in extracting and learning non-linear patterns from the data. Exosome SERS-based studies of breast cancer subtypes [51] utilized a MLP model, achieving an impressive classification accuracy of 95.45%. Similarly, deeper neural networks, including one-dimensional convolutional neural networks (1D CNN) and residual networks (ResNet), have shown high performance in classifying cancer-derived exosomes, such as in distinguishing lung cancer from healthy controls [52]. These results indicate that neural networks are highly effective for SERS analysis of exosomes. In this context, we would like to see if neural networks can distinguish exosomes secreted from normal and dysfunctional human aortic endothelial cells. These cells were grown under both normal and high blood sugar

(hyperglycemic) conditions to simulate endothelial dysfunction, which is common in cardiovascular inflammation.

Endothelial dysfunction is an early sign of developing atherosclerosis, especially in people with type 2 diabetes (T2DM). Early detection of this dysfunction is difficult because T2DM often starts without obvious symptoms. Therefore, we need a way to distinguish between healthy blood vessel cells and those that are starting to malfunction due to high blood sugar [55]. Exosomes released from hyperglycemic endothelial cells might help us diagnose these early changes. Recent research has shown that exosomes derived from endothelial cells play a crucial role in the development of atherosclerosis [56]. Thus, analyzing circulating exosomes could serve as an early biomarker for endothelial dysfunction. For this reason, it is crucial to investigate the use of more robust models, like neural networks, to improve the predictive accuracy for detecting this condition.

Dataset

Table 2.1 illustrates the breakdown of data. A total of 2,699 spectra were collected from exosomes from normal media (i.e., healthy), those with hypoglycemia, and those with hyperglycemia. They were recorded from 54 surfaces with each surface yielding 50 spectra. Each spectrum consists of approximately 2,050 absorbance values corresponding to the spectral range of 400 to 1,800 cm^{-1}. An example spectrum is illustrated in Figure 2.1. To reduce any potential negative impact of the imbalance across classes on machine learning experiments, an even distribution across each class is maintained with 18 surfaces or 900 spectra per class.

Table 2.1: **Exosome Dataset.** Each surface corresponds to a single sample, and all spectra from the same surface are acquired under identical conditions. Each surface yields 50 spectra, with each spectrum represented as a vector of approximately 2,050 absorbance values covering the spectral range of 400 to 1,800 cm^{-1}.

class	Total		Modeling set		Test set	
	#surfaces	#spectra	#surfaces	#spectra	#surfaces	#spectra
Healthy	18	899	16	799	2	100
Hypo	18	900	16	800	2	100
Hyper	18	900	16	800	2	100

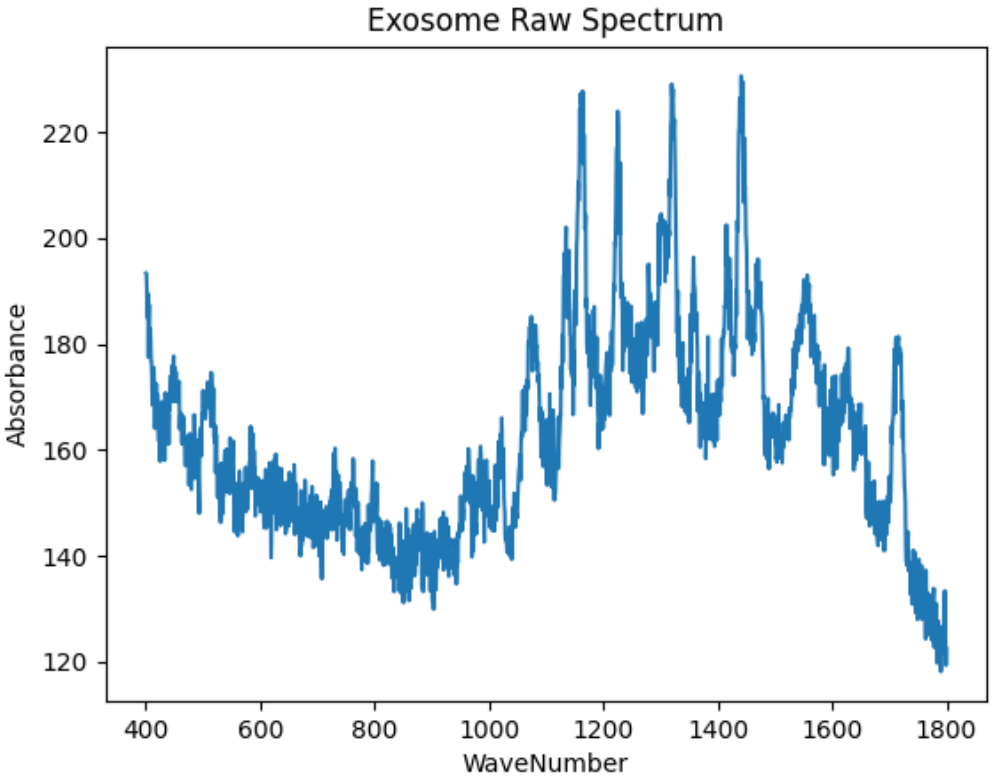


Figure 2.1: An example exosome spectrum.

The dataset was randomly split into two subsets: a modeling set and a final test set. Due to the fact that spectra from the same surface come from the same sample and environment which make them very similar, spectra from the same surface are kept in the same set. The test dataset consists of 6 surfaces, with 2 surfaces or 100 spectra per class while the modeling set has 48 surfaces in total and 16 surfaces or 800 spectra per class. Initially the modeling set is used for an 8-fold cross validation hyper-parameter tuning. Then the set is again used as the training set for the final neural network. At this point, the test dataset is used to determine the final accuracy of the method.

2.1.2 Oxygen Uptake

Oxygen uptake (VO_2), which measures the volume of oxygen consumed by the body per unit of time, is an essential indicator of aerobic endurance [57]. For example, soccer players with higher VO_2 max covering more total distance and intensity

parameters during games, suggesting a direct relationship between high VO_2 max and the ability to sustain greater physical demands [58]. Oxygen uptake is often measured using maximal oxygen uptake tests, which are often performed in controlled laboratory settings. This test involves analyzing the respiratory gases to calculate oxygen uptake during a standardized exercise protocol. However, frequently taking such invasive, standardized tests is often impractical, especially during competitive periods. Therefore, it is important to find easier ways to track VO_2 over time, so coaches and trainers can monitor training and fitness levels throughout the season.

Internal load, estimated oxygen uptake, primarily comes from the forces athletes generate to move. These movements apply forces to the environment, which then react back on the athlete's body, creating external loads. While measuring internal load is often expensive, and time-consuming, external load can be conveniently tracked using lightweight, wearable sensors. Understanding how internal and external loads relate is important because it helps us assess how an athlete is adapting to training, revealing changes in their fitness level or fatigue [59]. However, accurately measuring these movements is a challenge [60]. Among technologies for monitoring loads, like GPS or local positioning systems [61], inertial measurement units (IMUs) are becoming popular for tracking external load. An IMU typically combines 3-dimensional accelerometers, gyroscopes, and magnetometers into one device, which captures high-frequency data, making it ideal for monitoring high-intensity actions [62]. The sensor signals are then processed into indicative metrics of external load, like mean amplitude deviation [63], or accelerometry-based indicators by [64]. Wearable sensors have also been used to estimate oxygen consumption by transforming their data into different formats [63, 65, 66]. In this study, we aim to explore how well IMU devices can predict oxygen uptake. Specifically, we experiment with different placements of IMU sensors on the body to capture external loads and test different ways of representing the sensor data for machine learning models.

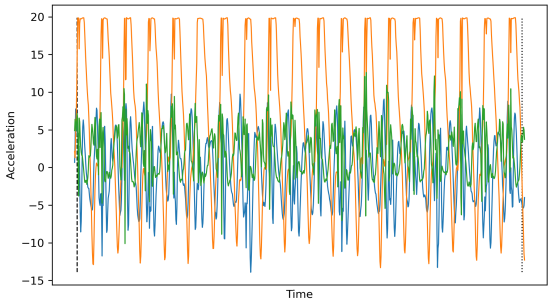
Another challenge is to model the connection between data from wearable sensors (external load) and the athlete's internal response [60]. Classical machine learning

models like random forests and XGBoost have shown promising results when using inputs such as heart rate, breathing frequency, and body motion to estimate VO_2 across different types of exercises, including walking, running, and cycling [65, 67, 68]. Despite that, these classical models often struggle with capturing complex, non-linear relationships, especially when the data is sequential over time. Neural networks, such as LSTM and temporal convolutional networks [69], have been employed to enhance predictive accuracy and capture nonlinearities over time. Recent studies have shown that these models outperform classical methods in estimating VO_2 during diverse activities, leveraging inputs like heart rate, speed, and respiration to provide more precise and personalized predictions [66, 70–72]. In Section 4.2, we compare the performance of different neural network architectures, considering different sensor placements and ways of representing data, to determine which approach works best for estimating VO_2 during outdoor jogging and simulated team sports activities. This comparison will help us understand the best method for using wearable sensors to monitor oxygen consumption during high-intensity sports activities.

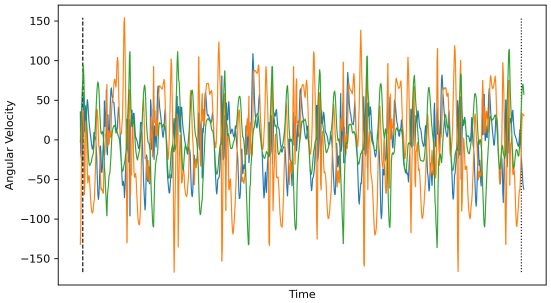
Data Collection

Data collection involved a group of 5 male team players where each participant completed two separate sessions. The first session took place in a controlled laboratory environment and included a resting phase, a sub-maximal exercise phase, and a maximal graded exercise test on a treadmill. The second session was conducted on a synthetic outdoor pitch and included a steady-state jogging trial and a simulated team sports circuit, designed to mimic team sports activities. Throughout both sessions, participants were equipped with multiple wearable sensors: a Cosmed K5 metabolic gas analyser for breath-by-breath VO_2 measurements, a Zephyr BioHarness for heart rate (HR) and breathing rate (BR) monitoring, and multiple Shimmer 3 inertial measurement units placed on the torso, right tibia, and left wrist to capture high-resolution motion data at 250 Hz. Figure 2.2 show examples of IMU raw data from the Accelerometer and Gyroscope during Treadmill Running, Outdoor Running, and

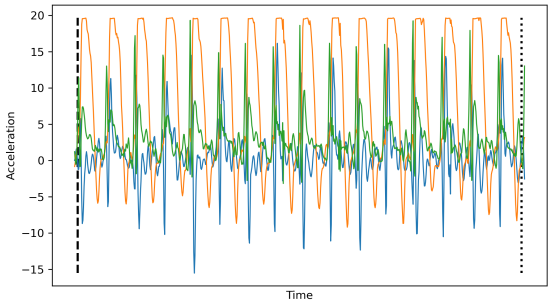
a Simulated Circuit. These sensors were synchronised to ensure accurate time-aligned data across physiological and biomechanical domains.



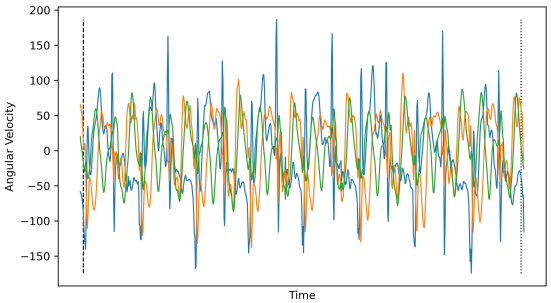
(a) Accelerometer signals during Treadmill Running



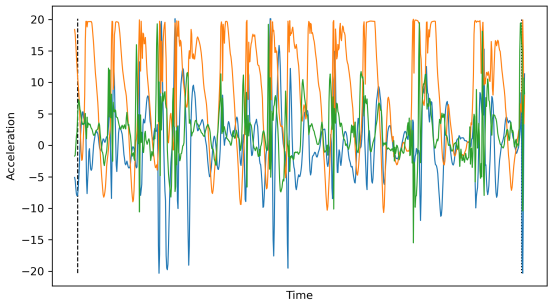
(b) Gyroscope signals during Treadmill Running



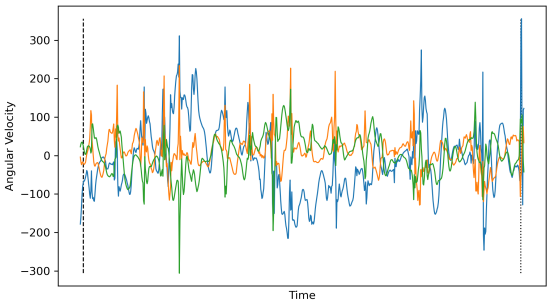
(c) Accelerometer signals during Outdoor Running



(d) Gyroscopec signals during Outdoor Running



(e) Accelerometer signals during Simulated Circuit



(f) Gyroscope signals during Simulated Circuit

Figure 2.2: Examples of IMU raw data during 3 different activities: Treadmill Running, Outdoor Running, and a Simulated Circuit. The blue line is the x-axis, the orange line is the y-axis and the green line is the z-axis.

Table 2.2 summarizes the dataset. The raw data consists of high-resolution sensor data collected from multiple wearable devices. The dataset includes breath-by-breath VO_2 measurements (in $\text{mL} \cdot \text{kg}^{-1} \cdot \text{min}^{-1}$), HR (beats per minute), and BR (breaths per minute) at a sampling rate of 1 Hz. The IMU sensors capture six-axis motion data,

including 3D accelerometer (x, y, z) and 3D gyroscope (angular velocity in x, y, z), at 250 Hz. These raw IMU signals are downsampled to 125 Hz to facilitate data alignment with physiological parameters. The dataset also incorporates contextual variables, such as activity labels (rest, treadmill running, outdoor running, and simulated circuit) and individual participant characteristics (height, weight, age, VO_2max).

Table 2.2: **Summary of the oxygen uptake dataset.**

Description	**Value**
Number of Subjects	5
Number of Columns	40
Sensor Columns	Oxygen uptake, Accelerometer, Gyroscope, Heart rate, Breathing rate
Contextual Columns	Height, Weight, Age, VO_2max, Activity labels
Number of Rows	3,952,923
Breath Count	12,195
Total Duration of Data	527 minutes

2.2 Graph Modeling

2.2.1 Graph Data Representation: Bike Sharing System

Modeling data as networks has gained considerable attention in the last two decades. Network science provides powerful tools that yield insightful results in many different areas, including social science ([73, 74]), transportation ([75, 76]), climate ([77, 78]), biology ([79]), and brain activity ([80, 81]). A network, or graph, is composed of nodes (vertices) and edges (links) that connect pairs of nodes. A common approach to constructing networks involves treating spatial entities of interest as nodes, such as cities, countries, grid cells, or transportation stations. Edges are used to represent intrinsic connections, interactions, similarities in temporal behavior, or functional relationships between nodes. A dataset can be transformed into a stationary network for analyzing overall behavior, or into a time-varying network to explore the dynamic nature of the system. Network analysis enables the identification of important

locations, community structures, temporal patterns across locations, and interactions between spatial elements over time.

In recent years, bike-sharing systems have become increasingly popular worldwide. These systems offer a convenient way for travelers to pick up a bike, travel, and drop it off. This approach offers a low-cost, efficient mode of transportation that addresses the first-and-last mile problem, reduces urban traffic congestion, and increases the overall efficiency of urban transportation systems. In addition, cycling is a healthy and environmentally friendly means of travel. However, the rapid expansion of bike-sharing systems also presents several challenging issues. The lack of bikes at certain locations and the surplus at others can lead to a waste of resources and poor user experiences. This raises the problem of effectively and efficiently rebalancing the system. Understanding travel patterns and traffic demand is essential to solve these challenges. Moreover, the distribution of pick-up and drop-off stations plays a critical role in maintaining user satisfaction and system efficiency. Well-placed stations provide great convenience for users, while too many stations can lead to high maintenance costs. Thus, selecting optimal locations for stations is crucial for managing a bike-sharing system effectively.

Traditionally, bike-sharing systems rely on docking stations placed at fixed locations across a city, where customers could collect and return bikes. More recently, dockless bike-sharing services have emerged, allowing bikes to be picked up and dropped off at more flexible locations, referred to as virtual stations. It is the best interest to continuously monitor bike usage and determine the optimal configuration of these virtual stations. Essentially, the problem can be posed as follows: “How similar is the current virtual station network to the optimal network configuration that maximizes bike usage?” Normally, bike-sharing systems make their data available in a tabular format. Additionally, station location and trip data must be accessible at different levels of granularity to provide both a high-level overview of network activity and more fine-grained analysis where needed. Any solution must account for the fact that decisions made at a global level can have negative consequences

for individual stations, while changes made locally to fix an issue may impact the network as a whole. In Section 5.2, we would like to propose a systematic method for applying complex networks in bike-sharing analysis. The framework should meet requirements from bike sharing system that: is a high-level visualization to highlight the busiest and least-used stations and routes, support time-based drill-downs for analyzing sub-networks over flexible intervals, identify station similarities across spatial and temporal dimensions, and conduct a comprehensive analysis of station activity patterns in relation to all other stations to reveal both common and unique travel behaviors.

Dataset

Our original dataset comprised data from 86 bike stations and recorded a total of 52,936 rentals between June 2020 and September 2021, as summarized in Table 2.3. This dataset also includes detailed information on both pick-up and drop-off times and locations. However, many entries in the dataset involve *random* pick-up and drop-off locations, which require data cleaning before its integration into our graph networks.

Table 2.3: Dataset Overview

Measures	**Original Dataset**	**Cleaned Dataset**
Duration of data	June 2020 - Aug 2021	
Station count	86 stations	
Rental count	52,936 trips	36,181 trips
Max trips (per station)	2,775 trips	1,936 trips
Min trips (per station)	97 trips	82 trips

Firstly, we evaluate whether the pick-up and drop-off locations correspond to existing stations. If these locations are within a 1km radius of at least one station, they are reassigned to the nearest station. If they are located more than 1km away from any station, they are excluded from the dataset. Additionally, trips have both pick-up and drop-off locations at the same station, resulting in what we define as a "loop-trip". In this first step, 654 (approximately 1.2%) are reassigned, while 6,650 trips (around 12.5%) are removed.

Secondly, certain trips in the original dataset may have been created due to system errors or user mistakes. These errors could stem from app compatibility, mistakes in app usage, and failures in properly starting or ending a trip. Specifically, very short trips are defined as those lasting less than 10 minutes or covering a distance of less than 100 meters, while very long trips are defined as those exceeding 1 day in duration. The thresholds for these parameters are determined via a pre-experiment analysis on the original dataset. As a result, in the second step, 10,105 trips (approximately 19%) are removed.

The cleaned dataset now contains 36,181 trips across 86 stations in Dublin city, Ireland, as summarized in Table 2.3. On average, there are 2,412 trips monthly across all stations, with each station averaging 28 trips per month. The busiest station recorded 1,936 trips over 15 months,equivalent to 129 trips per month, while the least frequented station had only 82 trips over the same period, averaging 5 trips per month.

2.2.2 Graph Neural Networks: Air Quality

Air quality, a critical aspect of environmental and public health, has gained increasing prominence in recent years due to the rapid growth of urbanization and industrialization [46, 82]. As air pollution continues to impact the health and well-being of millions worldwide [83], the need to monitor and manage air quality has become more urgent than ever. Air quality forecasting, a key component in addressing this global challenge, serves as an indispensable tool for policymakers, researchers, and citizens alike. By predicting future levels of air pollutants, air quality forecasting enables informed decision-making and the implementation of effective strategies to mitigate pollution, safeguard public health, and promote sustainable development.

One of the mainstream methods for air quality forecasting are mainly is data-driven statistical approaches, commonly referred to as machine learning, have proven effective in predicting air quality by relying on extensive historical data to identify recurring patterns. Notable machine learning techniques include linear regression [84],

multi-layer perceptron [85, 86], long short term memory [87, 88], and graph neural networks [89, 90]. However, current machine learning methods still face limitations when it comes to effectively extracting patterns in the intricate time-space relationships that govern air quality. Many existing models tend to focus on isolated spatial or temporal dimensions, which may hinder their ability to fully capture the complex inter-dependencies between time and space in air quality data. Other limitations come from the mathematical model of the machine learning models. For instance, RNNs typically employed for time series data, are known for their slow training and prediction times due to their recurrent nature. RNNs also struggle to learn long-term dependencies. In other models, graph neural networks can inherently capture spatial relationships, they may not be effectively model the temporal dynamics of spatial-temporal data, as the adjacency matrix is often predetermined or fixed along the temporal axis.

To address these limitations, in Section 5.3, we introduce the attention mechanisms into both spatial and temporal dimensions. Attention mechanisms enable the model to identify which the most influential factors and allocate more focus on them. These mechanisms have proven to be highly effective in enhancing the performance of deep learning models across various domains, including natural language processing, computer vision, signal processing, etc. By applying attention to both temporal and spatial dimensions, information at a specific location is not only processed individually but is also informed by data from other locations, the same happens for information at a given timestep which is processed as a result of surrounding timesteps or even those further away. Therefore employing temporal and spatial attention layers concurrently enables the model to learn spatial-temporal information simultaneously.

Dataset

Table 2.4 provides an overview of the dataset used in our experiments. The dataset was collected from 10 monitoring stations located in Hanoi, Vietnam, over a period

of more than 18 months, from January 1, 2020, to October 31, 2021. The locations of these stations are shown in Figure 2.3. The dataset includes pollutant concentration levels, specifically PM2.5 values, as well as auxiliary features such as temperature and humidity. As an example, Figure 2.4 illustrates the PM2.5, humidity, and temperature measurements recorded at a station throughout the data period. The data was collected at intervals of a few minutes and subsequently pre-processed into hourly timeframes. In total, the dataset comprises 115,600 records across the 10 stations, covering approximately 90% of the entire period.

Table 2.4: **Summary of the air quality dataset.**

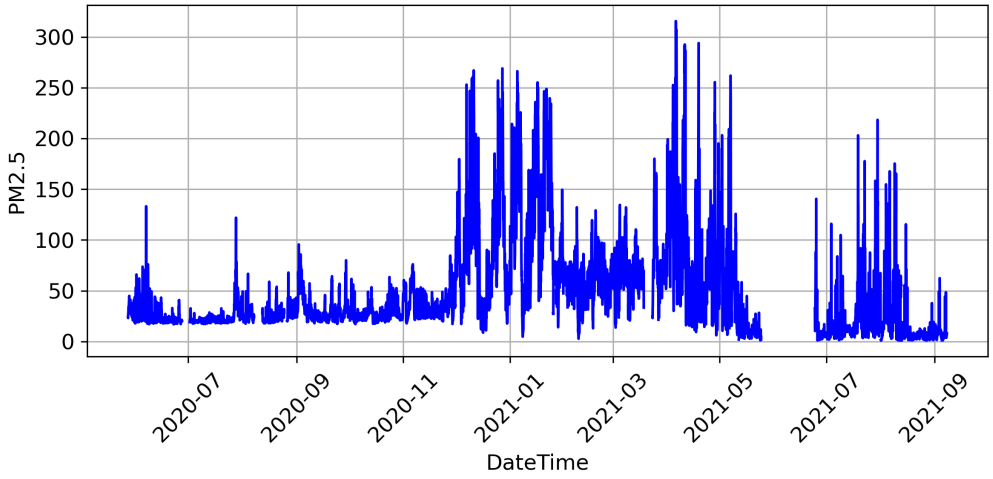
Description	**Value**
Locations	10 stations in Hanoi, Vietnam
Date Range	January 1, 2020 – October 31, 2021
Columns	PM2.5, Humidity, Temperature (Hourly Data)
Total Rows	115,600 rows

Figure 2.3: The 10 considered air quality monitoring stations in Hanoi, Vietnam

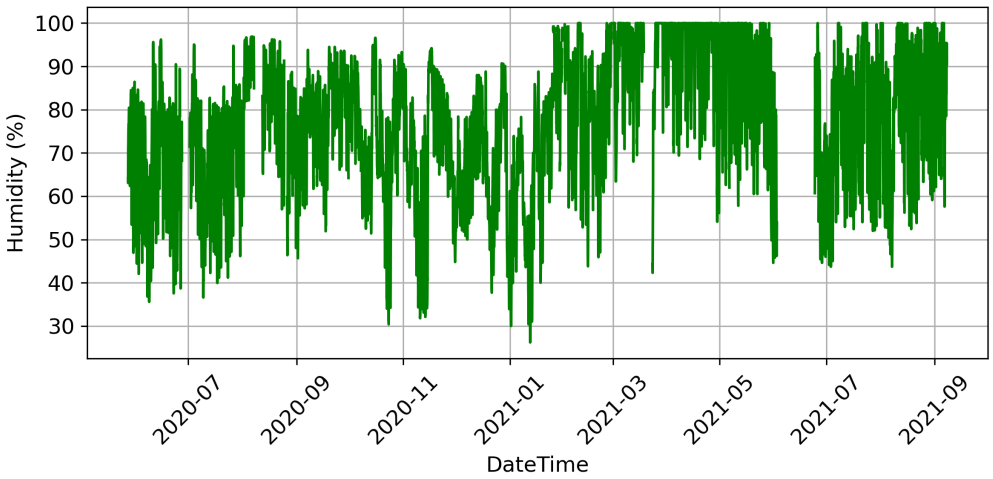
2.3 Physics-Informed Neural Networks

2.3.1 Physics Informed Neural Networks

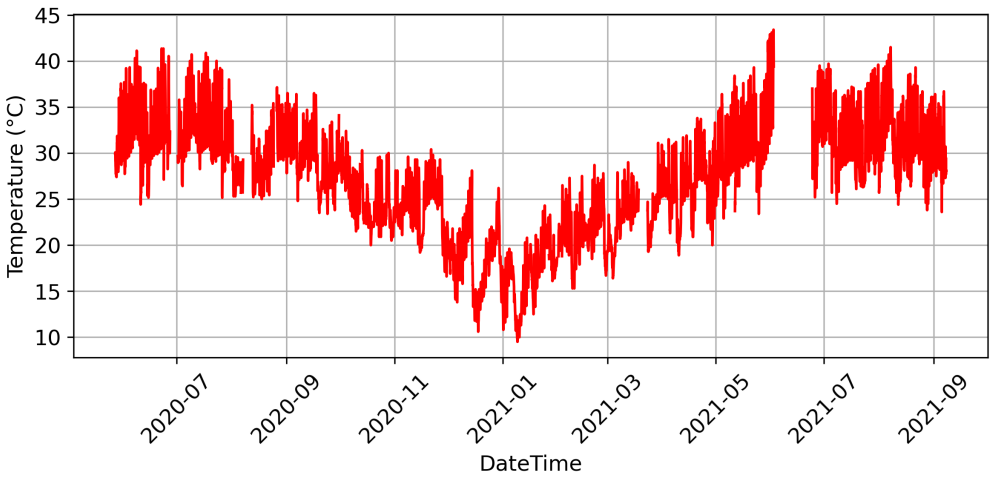
Recent advancements in computational capabilities and the vast increase in data availability have made data-driven approaches a leading strategy in both research and applications. Typically, these approaches involve training neural networks to



(a) PM2.5



(b) Humidity



(c) Temperature

Figure 2.4: Examples of air quality data at a station.

minimize discrepancies between model predictions and observed data. However, this purely data-driven approach has some limitations, such as poor interpretability [91], limited generalization on unseen data [28], and a need for large amounts of training data [26].

Physics-informed neural networks [43] present an alternative for cases where the data is governed by physical laws, often represented as differential equations. In this approach, physical laws are embedded directly into the learning process by adding extra loss functions, minimized along with the traditional data-fitting objective. This approach attempts to both fit observational data and approximate the underlying governing equations. As a result, PINNs are capable of respecting physical laws, enhancing generalizability, and revealing latent patterns from data. This framework can also solve forward problems (predicting system states) and inverse problems (identifying system parameters) simultaneously. Applications of PINNs span a wide range of fields, as surveyed in [26, 92–96].

Despite their potential, PINNs still face challenges in handling multi-scale and stiff solutions [26, 41, 42]. Several advancements have been made to enhance the original framework of PINNs introduced in [43]. These include new neural network architectures [41, 42, 97, 98], adaptive activation functions [99, 100], and improvements in multi-task training by adaptively adjusting loss weights [42, 101, 102]. Researchers have also experimented with the distribution of collocation points [103–106], sequential learning approaches for preserving causality [107–110], and domain decomposition methods to improve convergence and accuracy by training PINNs on subdomains [111–113].

In Chapter 6, we aim to further extend the capabilities of PINNs to tackle extremely complex and multi-scale dynamical systems, using mosquito population modeling as a case study [44]. We propose a set of improvements aimed at overcoming existing challenges in PINN training, which include systematic normalization procedures, advanced loss weighting strategies for diverse scales of the governing equations, more refined training phases for better initialization and convergence, and

simplified domain decomposition techniques. By enhancing these methods, we aim to make PINNs applicable to more complex, real-world dynamical systems. The proposed method will be validated on the popular Lorenz system and applied to a mosquito population model, an ODE system chosen for its practical relevance and multi-scale dynamics across mosquito life stages within a large domain.

2.3.2 Inverse Problems: External Forcing

Physics-based dynamic models (PBDMs) are simplified representations of intricate dynamical systems, capturing essential processes through a defined set of *parameters* and variables. These models have extensive applications across research and technology, ranging from predicting air temperature [114] to the spread of COVID-19 [115] and cancer cell development [116]. "Physics-based modeling is powerful and effective because it gives us a predictive window into the future based on understanding" [117]. This is possible because PBDM is designed to focus specifically on a particular class of physical systems or processes, creating a generalized representation applicable within that scope. However, the accuracy of PBDMs is highly dependent on their parameterization, which is often suboptimal, resulting in uncertainties in their predictions. Refining these parameters is an inverse problem, where the objective is to estimate model parameters based on observed data. This is a key challenge in improving the performance of PBDMs. Solving inverse problems is essential for enhancing both the predictive accuracy.

PINNs offer a potentially powerful tool for solving inverse problems, especially when dealing with limited data. PINNs have been applied to a diverse range of inverse problems, including parameter estimation in nano-optics and metamaterials [118], subsurface flow modeling [119], structural engineering [120, 121], biological systems [96, 122], civil structures [120], etc. However, existing implementations of PINNs do not incorporate external factors that may influence system dynamics.

Accurate identification of both internal and external factors that influence the state of a system over time is essential for ensuring the reliability and accuracy of

model predictions. Data-driven determination of external forcing factors based on historical data allows for the identification of new, or the better parameterization of identified forcing factors and system responses. The identification of these forcing factors and their modeling is an active research domain that lies at the intersection of physical modeling and machine learning, and it extends into the areas of attribution and causality analysis [45–47].

In Chapter 7, we build upon the PINN framework introduced in Chapter 6 where only idealized annual variations in daily temperature is considered. However, based on our understanding of mosquito population dynamics, we hypothesize that additional meteorological factors such as air humidity and precipitation play a significant role as external factors that influence the biological system. To incorporate a complete set of measured meteorological data, we introduce a novel approach using parameter networks that accept inputs from these meteorological conditions. Moreover, we propose employing a Multi-branch Fourier-feature Multi-Layer Perceptron for the parameter networks, which we expect to enhance generalization capabilities. Furthermore, we implement a modified absolute activation function that enforces parameter non-negativity. The proposed framework is validated against real-world observational data and benchmarked against traditional empirical formulas, demonstrating its potential to improve upon existing models.

2.3.3 Mosquito Population Modeling

Arboviruses, transmitted by mosquitoes, can spread rapidly and cause major epidemics of diseases like malaria, dengue, Zika, Chikungunya, and West Nile Virus [123–127]. Many approaches exist to model mosquito population dynamics and seasonal variations to help predict disease risk, broadly categorized into mathematical and statistical models. Mathematical models use laboratory and field data to parameterize life history traits such as the development and mortality rates of mosquito life stages [44, 128–131]. Conversely, statistical models employ correlative and machine learning techniques to link vector abundance with various abiotic factors [132–

138]. Statistical models generally need long-term time-series data from mosquito surveillance, which is labor-intensive and prone to multiple sources of bias.

In Chapters 6 and 7, we explore the feasibility of using PINNs trained on an ODE model of mosquito population dynamics to bridge the gap between traditional mathematical modeling and modern data-driven approaches. Our goal is to retain the physical and biological constraints that govern these systems, while also taking advantage of the power of data science and machine learning techniques.

Dataset

Table 2.5 summarizes the data used in our experiments in Chapter 7, which comprises two distinct datasets collected from Petrovaradin, Serbia. The training dataset spans from February 2016 to December 2017, covering 700 days, and includes 680 daily records of mosquito trap counts. The test dataset, used for validation purposes, was collected over a longer period from January 2000 to December 2007, encompassing 2,921 days. It consists of 179 weekly records gathered only during active mosquito periods, a broader time scope with less frequent measurements. Both datasets focus on counting the number of blood-seeking adult mosquitoes ($A_{b1} + A_{b2}$).

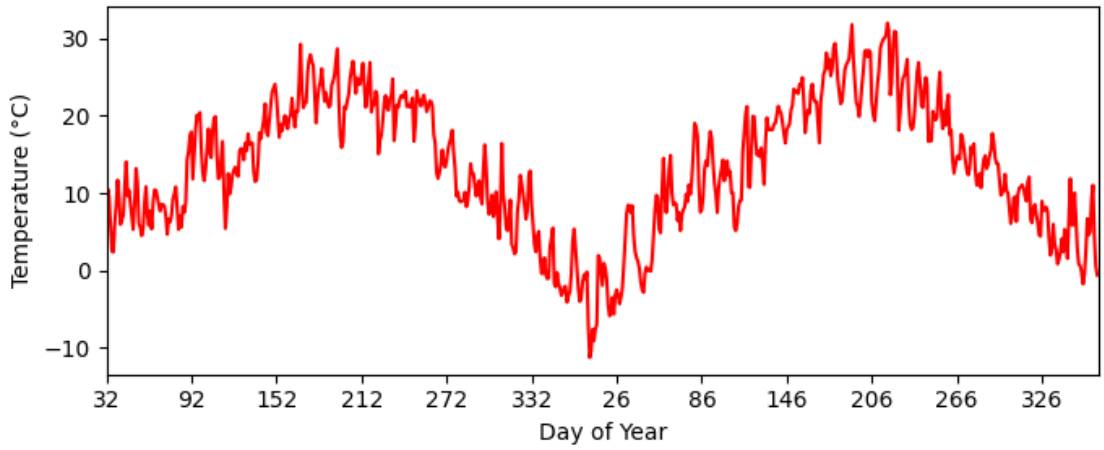
Table 2.5: Mosquito Data for Experiments

	Training Data	Test Data
Location	Petrovaradin, Serbia	
Period	Feb 2016 - Dec 2017	Jan 2000 - Dec 2007
#Days	700	2921
Data Collection	Daily	Weekly
#Records	680	179

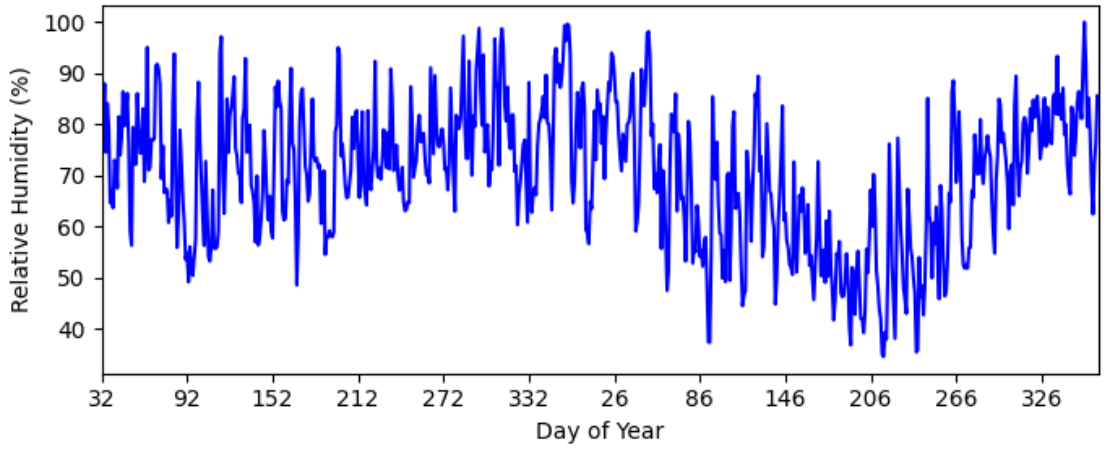
We anticipate that not only air temperature, but also air humidity and precipitation play active role of “forces” affecting our biological system. As external factors, environmental factors considered include daily measurements of air temperature, relative humidity, and precipitation. Due to the absence of on-site meteorological data during the study period, a linear regression model was developed using 2016/2017 measurements from Petrovaradin and the Rimski Sancevi climate station. This

model enables the estimation of meteorological values for the Petrovaradin site when only Rimski Sancevi data are available.

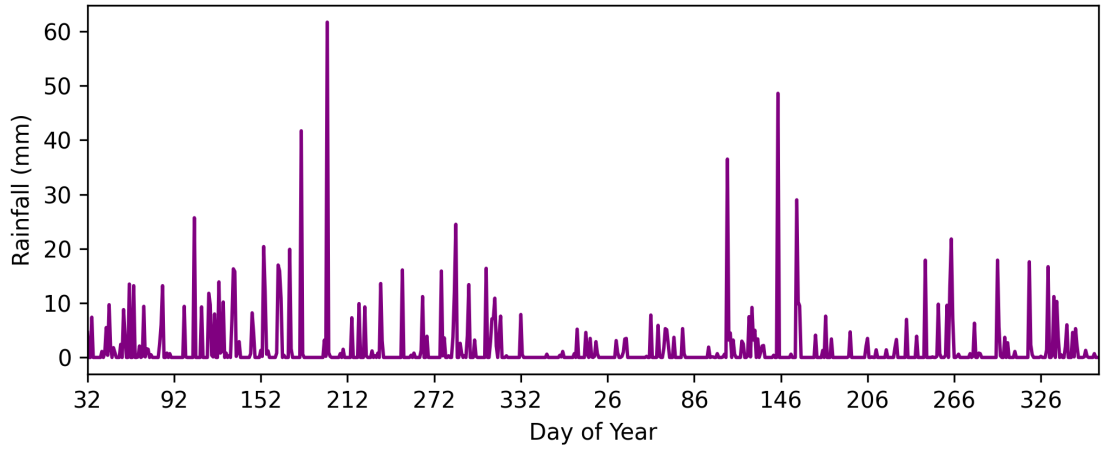
As an example, Figures 2.5 plot the meteorological measurements over the two-year training period. The air temperature has an average of 18.5 °C with a standard deviation of 10, ranging from -8.9 °C to 40.3 °C. It shows a clear periodic pattern resembling a sinusoidal wave, peaking during mid-year. In contrast, the relative humidity averages 70.3% with a standard deviation of 13.6%, ranging from 34.5% to 100%. It exhibits less pronounced annual variation compared to temperature, with a larger dip during the summer of the second year. Precipitation averaged 1.8 mm with a standard deviation of 5.2 ranging from a minimum of 0.0 mm (indicating no rainfall) to a maximum of 61.7 mm. Precipitation peaks occur in clusters of days, likely corresponding to seasonal climatic events.



(a) Temperature



(b) Relative Humidity



(c) Rainfall

Figure 2.5: Meteorological measurements of the training period.

Chapter 3

Literature Review

In this chapter, we present a comprehensive literature review around neural network and graph network deployment in real-world applications and also around more customized forms of neural networks. Figure 3.1 outlines the chapter's structure. In terms of real world applications, the review focuses on related work for the domains selected for this research: disease detection, oxygen uptake in athletes, air quality detection and mobile networks. Finally, we present a literature review on physics-informed neural networks, focusing on their training techniques and approaches to managing the inverse problem associated with these types of neural networks.

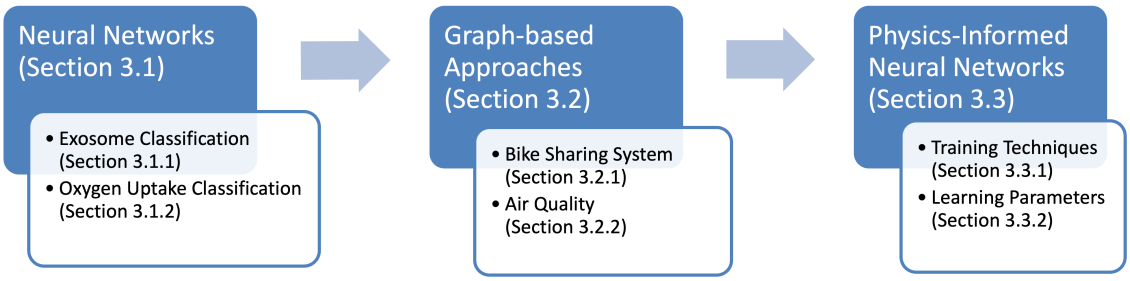


Figure 3.1: Literature Review Chapter Structure

3.1 Neural Networks

3.1.1 Disease Prediction Using Exosomes

Discrimination of cancer cell-derived exosomes has been a primary focus in exosome SERS studies. Carmicheal et al. [49] reported the SERS of exosomes from cell culture medium, demonstrating that principal component differential function analysis (PC-DFA) enabled effective classification of exosomes derived from both a healthy cell line and two pancreatic cancer cell lines. Their model achieved a cross-validation with 90.6% sensitivity and 97.1% specificity. Similarly, researchers in [50] conducted several studies highlighting excellent discrimination between cancerous and non-cancerous exosomes, using principal component analysis (PCA) to identify tumor-derived fingerprints on gold nanoparticle substrates. The study demonstrated that exosomes from two lung cancer cell lines could be effectively distinguished from those originating from normal alveolar lung cells. Furthermore, the study [50] also investigated specific surface protein compositions of exosomes from non-small cell lung cancer cells, revealing strong correlations with several protein markers, which suggested these proteins could serve as exosomal surface markers for cancer diagnosis. PCA analysis was once again employed to differentiate these markers.

Recently, neural network-based methods have emerged as a promising approach to enhance the analysis of SERS data. Neural networks have been applied in several SERS exosome studies and appear to offer superior analytical power for evaluation and discrimination of complex Raman signals and particularly for systems like exosome SERS where there may be significant variation in SERS signature. In examples to date, exosome SERS-based neural network analysis have been collected from mixtures of exosomes [51, 52, 139]. Specifically, the authors in [139] illustrated the potential of machine learning approaches in differentiating signals from exosomes derived from distinct cell lines. Methodology by [51] employed neural networks to distinguish SERS signals from exosomes derived from four breast cancer subtypes, using 8265 SERS spectra to train a multi-layer perceptron (MLP) for this four-class

classification task. Their model achieved an accuracy of 95.45%, outperforming several other models, including principal component analysis with linear discriminant analysis (PCA-LDA), partial least squares discriminant analysis (PLS-DA), support vector machines (SVM), and one-dimensional convolutional neural networks (1D CNN). Additionally, PCA analysis of the MLP's second last layer was shown to be able to evaluate surgical outcomes of breast cancer subtypes. In the study [52], a deep neural network approach was used to classify exosomes from normal and lung cancer cell lines. A one-dimensional residual convolutional neural network (similar to ResNet [140]) was trained on 2150 SERS signals derived from 20 healthy controls and 43 lung cancer patients, achieving an accuracy of 95% with an area under the curve (AUC) of 0.912. The values from the final layer were subsequently used to perform PCA, which revealed that the similarity among exosome groups was proportional to cancer progression.

Summary. Two primary approaches are used in exosome SERS signal classification: one approach employs the first few principal components, typically two, from PCA to cluster the signals, while the other approach inputs the SERS signal vectors directly into machine learning models, particularly neural networks. While performances vary across studies, neural networks, including MLPs and CNNs, have consistently shown superior results. These studies explore the capabilities of exosome SERS signals in the context of cancer diagnostics.

We distinguish our research by taking these approaches further. Our investigation will also focus on whether neural networks could discriminate exosomes secreted from normal and dysfunctional human aortic endothelial cells (HuAECs) where cells were grown under normal and hyperglycemic conditions to promote endothelial dysfunction. This remains an open research topic for both biologists and chemists.

3.1.2 Oxygen Uptake Estimation

Many studies have focused on estimating oxygen uptake (VO_2). Traditional methods often rely on manually defined mathematical formulas that relate VO_2 to predictor

variables, with model parameters often holding physical or physiological meaning [141–143]. While these models are more interpretable, they are limited by the number of variables they can handle and the complexity of mathematically capturing their relationships. Machine learning models have emerged as a powerful alternative for estimating oxygen uptake, capable of complex nonlinear relationships [144].

Several studies have leveraged wearable sensors and machine learning to estimate VO_2 during daily activities. Beltrame et al. [65] utilized data from wearable sensors, such as heart rate, hip acceleration, ventilation, and breathing frequency, captured by a smart shirt. They employed a random forest model to predict VO_2 dynamics. Wang et al. [67] estimated VO_2 across various activities, including stationary postures (lying, sitting, standing), walking, treadmill running, and recovery phases. Using data from a medical-grade wearable vest transformed into features related to heart rate, respiratory rate, lung ventilation, and exercise intensity, they applied XGBoost, linear regression, and random forest models. The XGBoost model achieved the highest accuracy, reducing mean absolute error by 54.7% compared to heart rate-only models. In unsupervised daily activities, the work [145] used wearable sensors monitoring heart rate, breathing rate, minute ventilation, total hip acceleration and physiological inputs (sex, age, weight, height, and various vital signs) to predict VO_2 max via support vector regression. The model showed a high correlation with traditional cardiopulmonary exercise testing.

Focusing on treadmill walking, the study [68] used easily accessible inputs like treadmill speed, grade, body mass, sex, exercise/recovery time, and heart rate. They implemented a multilayer perceptron to optimize VO_2 dynamic predictions. Amelard et al. [70] estimated VO_2 during cycle ergometer exercise using work rate data and data provided by a smart shirt, including heart rate, breathing frequency, and minute ventilation. Their results indicated that temporal convolutional networks and long short-term memory networks outperformed random forest models across various intensities. Further research by the same team demonstrated that Temporal Convolutional Networks (TCNs) accurately predict slower VO_2 kinetics

with increasing exercise intensity [71]. Davidson et al. [66] employed an inertial navigation system combined with GPS and a heart rate monitor to estimate VO_2 during walking and running. They trained an LSTM model using inputs like speed, speed change, cadence, vertical oscillation, and heart rate. In cycling activities, Zignoli et al. [72] trained personalized recurrent neural networks using heart rate, mechanical power, cadence, and respiratory frequency. Their neural network approach showed superior predictive power compared to traditional mathematical models for VO_2 kinetics.

Vähä-Ypyä et al. [63] compared features derived from raw three-dimensional acceleration signals during maximal track or treadmill tests. They found that the mean amplitude deviation (MAD) metric performed well for walking, while other metrics showed less consistent accuracy during running. Research by [146] highlighted that incorporating gyroscope data with accelerometer readings can enhance fatigue detection in runners.

Summary. Advancements in machine learning have significantly improved VO_2 estimation by using data from wearable sensors and advanced neural networks. These studies demonstrate enhanced accuracy over traditional methods by capturing complex physiological dynamics across diverse activities, from daily living tasks to structured exercises. Despite this work, there remains a gap in systematically exploring how different combinations of inertial measurement unit (IMU) sensors, data representations, and neural network architectures impact VO_2 estimation accuracy. This research will address this gap by experimenting with various feature sets (IMU sensor combinations) and evaluating different neural network models.

3.2 Network Models

3.2.1 Transport Networks: Bike Sharing

Transport Network Construction

Complex network analysis techniques have been widely applied to study bike-sharing systems across the globe. In these studies, bike-sharing data are represented through network structures and by analyzing these networks, researchers can reveal fundamental characteristics of bike-sharing among the population. The construction of networks varies based on the objectives of each analysis and network modeling varies depending on the purpose of the analysis. Most commonly, traffic flows are analyzed by viewing spatial locations as nodes, with trips forming edges between each trip's origin and destination [147–153].

Most of the systems studied are dock-based, meaning users must rent and return bikes at designated stations. Therefore, these stations are typically considered as *nodes* in network models. However, *dockless* systems, which allow users to pick up and drop off bikes anywhere, introduce challenges in data representation and modeling, requiring novel approaches. For example, authors in [147] divided the study area into a grid of squares, treating each square as a node, while Yang et al. [148] modeled physical road segments as nodes. Some studies employ an alternative approach by grouping locations and representing each group as a single point in the network. For instance, the study [149] classified bike stations based on their surrounding environment, and researchers in [150] utilized clustering techniques to group stations.

Other methods for constructing complex networks have also been developed to analyze bike-sharing systems. Batista et al. [151] constructed a network where each node represents a specific region within which vehicles travel at the identical average speed; nodes are connected if their corresponding regions are adjacent. This network facilitates the study of relationships between factors such as average travel distance or travel time and the levels of exhaust emissions along bike paths. The authors in [152] trained a graph convolutional neural network to capture the

correlations among stations and predict hourly demand at the station level. The correlation matrix learned during this training process serves as the adjacency matrix for constructing a network, offering insights into the spatial relationships between stations. Additionally, Ghandeharioun et al. [153] developed road networks based on pairwise edge correlations to estimate route travel times.

Although complex network analysis has been widely applied in studying bike-sharing systems, the literature lacks a systematic framework for network construction and analysis, especially with regard to optimizing networks to enhance their analytical power. Furthermore, the potential of correlation-based networks to reveal spatiotemporal patterns remains largely unexplored within the context of bike-sharing systems Therefore, this study aims to develop a formal framework for applying complex network methodologies and to promote the use of correlation-based networks in bike-sharing data analysis.

Network Dynamics

Beyond the static properties of networks, the evaluation of dynamics of bike-sharing systems is also essential. Thus, constructed networks need to be aggregated over different time intervals and periodicities. The choice of these intervals typically depends on the analytical objectives or guidance from domain experts. For instance, authors in [148] assessed the impact of a newly introduced metro line on travel flows within a bike-sharing system in Nanchang, China, by comparing network structures from five days before and after the metro's introduction. Similarly, Jianmin et al. [154] examined changes in bike-sharing systems in response to the outbreak and recovery phases of the Covid-19 pandemic by analyzing networks projected onto the pandemic waves. Studies by [155] and [156] also successfully identified distinct patterns in bike usage across weekdays and weekends, as well as different times of day, through network analysis.

In these studies, time interval selection is predetermined based on specific research interests, which may not always be the ideal choice. Without domain knowledge,

identifying patterns and anomalies across the time dimension and selecting relevant periods for analysis can be challenging. Analyzing every individual time step would be both time-consuming and inefficient. To address this, we propose a clustering-based approach to group similar time steps, allowing for the analysis of these aggregated representations instead.

Network Analysis

In terms of network analysis of bike-sharing networks, similar methodologies are often used, including network metrics, community detection, and the use of visualization tools supported by domain knowledge. Global metrics provide insights into the overall structure of the network, while local metrics reveal the roles or properties of individual nodes within it. Commonly used metrics include the number of nodes, number of edges, as well as degree and strength, which indicate the level of activity and connectivity at a given location [147, 157].

Austwick et al. [156] identified consistent patterns in strength and edge weight distributions across networks constructed from various bike-sharing systems. Meanwhile, the studies [148, 154] recommended incorporating a diverse range of network properties to capture multiple aspects of network structure. These properties include connectivity metrics (such as degree and node flux), spatial distribution (like the clustering coefficient), and interaction metrics (accessibility), as well as indicators of network stability (network connectivity), efficiency (network efficiency), and equity (Gini coefficient). Additionally, other centrality measures, such as betweenness and PageRank, used by [148], have proven valuable in this context.

Community analysis, a prominent approach in network research, plays an essential role in understanding the structure of networks. Community detection algorithms partition the network into distinct communities, where nodes have stronger connections within communities than between them. The Louvain algorithm [158] is among the most widely used methods for this purpose. However, Shi et al. [150] observed that different algorithms yield different community structures depending

on the measurement criteria. Although network analysis techniques are now well-established, they are typically applied to networks where edges represent trips. Our goal is to extend these approaches, including visualization, metrics such as strength, closeness, betweenness, local clustering coefficients, and community detection, to more complex, correlation-based networks.

Summary. While numerous studies have applied network-based analyses to bike-sharing systems, several key limitations remain in current research: (1) existing studies lack more systematic methodologies for applying complex networks in bike-sharing analysis; (2) network construction is frequently overlooked meaning this process lacks the optimization necessary for more effective network analysis; (3) when analyzing system dynamics, the choice of time periods mainly relies on domain knowledge, such as grouping hourly data by similar patterns or identifying evolving periods for stations; and (4) correlation networks hold potential for uncovering both spatial and temporal patterns, yet their application in network analysis remains under explored. Our approach will address each of the existing limitations.

3.2.2 Graph Neural Networks: Air Quality

Statistical methods, particularly neural networks, have become the most popular approach to the air pollutant concentration forecasting problem [159]. Beyond simple models like linear regression [160], support vector machines [161, 162], and MLP [85, 86], recent neural network architectures are designed to exploit both the time-series nature of the inputs and the spatial information of the prediction locations and their surroundings. Typically, recurrent neural networks such as long short-term memory (LSTM) [163] and gated recurrent units (GRU) [164] are used for time-series data, while graph neural networks (GNNs) [38] are used to model spatial relations among prediction locations. Convolutional neural networks [165], popular in image and signal processing, are also suitable for time-series data and can model spatial aspects in air quality problems. Moreover, attention mechanisms have recently proven to be powerful tools for modeling sequence inputs [18, 166], addressing scaling and

long-term dependency issues in recurrent neural networks. Attention is also suitable for more complex data structures like graphs and networks [167]. In this section, we investigate some typical examples of applying these technologies to air quality problems, covering recurrent neural networks, graph neural networks, and attention mechanisms.

Recurrent Neural Networks

Recurrent neural networks are traditional neural networks for working with time-series data, and air quality time-series are no exception [87, 88, 168–172]. Authors in [170] introduce a novel encoder-decoder model with an improved LSTM and apply it to air pollutant predictions. They modify the LSTM cells by adding new gates to enable the model to learn better long-term features and temporal correlations among different data features. The new architecture outperforms traditional LSTM and GRU encoder-decoder models in air quality prediction across 10 cities in China. The study [172] stacks time-delayed historical data from all monitoring stations as input into an LSTM model, allowing both spatial information and temporal correlations to be learned simultaneously within the LSTM cells. Researchers in [168] use a model combining LSTM and fully connected neural networks to predict PM2.5 values over the next 48 hours. The LSTM component models local changes in PM2.5 concentrations along with meteorological and weather features, while the fully connected module combines outputs from the LSTM to model spatial correlations among different locations within a city. Similarly, Wang et al. [169] apply a hybrid sequence-to-sequence architecture to predict ground-level ozone concentrations in Beijing, China, capable of handling both spatial and temporal data.

Summary. The above uses of recurrent neural networks are mostly for feature extraction along the temporal dimension of the data. Spatial features are considered either through fully connected layers or by stacking all data into a tensor. These attempts at incorporating spatial structure into models remain overly simplistic and should be further improved. Furthermore, one disadvantage of recurrent networks is

their computational speed, as each time step requires outputs from the previous step, leading to slow training and prediction for long sequences. Addressing this drawback is necessary to scale to longer historical data and their boost applicability to real world applications.

Graph Neural Networks

Graph neural networks (GNNs) are popular for modeling spatial structures and are often combined with recurrent neural networks to learn spatiotemporal dependencies in air quality forecasting [89, 90, 173–175]. Authors in [90] incorporate graph convolutional layers and LSTM into a single neural network to model and forecast future hourly PM2.5 concentrations. They construct a network structure between air quality stations based on distances, allowing the model to extract spatial features from each location and its neighbors. The extracted spatial signals are input into an LSTM layer to model time-dependent patterns. The model, tested with data from the North China Plain, shows superiority over traditional neural networks like MLP and LSTM. Gao et al. [173] integrate graph structure into LSTM cell to form a graph-based LSTM (GLSTM), where the cell considers information from other locations based on a trainable adjacency matrix, enabling simultaneous learning of spatial and temporal features. The trainable adjacency matrix allows the model to learn spatial correlations automatically, showing positive results over previous models [90] using data from Gansu Province, China. In contrast, Zhou et al. [174] incorporate theory into the training of GNN-LSTM models, guiding the model to learn theory-backed spatial information and improve generalizability. At larger scale, the work [175] applies GNNs hierarchically at station and city levels, with edge weights determined by geographic similarity and dynamically influenced by wind direction, effectively capturing large-scale spatial correlations during training.

In general, GNNs have proven suitable for modeling spatial correlations. However, current works often use a single adjacency matrix to represent spatial relationships, which can be disadvantageous as these relationships can be dynamic and depend on

many factors. More complex mechanisms in GNNs, such as attention, can provide more powerful spatial modeling.

Attention Mechanism

Attention mechanisms have also been applied to enhance recurrent and graph neural network architectures. In air quality problems, attention layers have been used either to enhance LSTM/GRU networks [169, 176–178] or to model spatial interactions between locations [179, 180]. When applied to the temporal dimension, attention is usually used near the prediction stage, especially in the decoder part, where it is most needed to give attention to proper timesteps. Authors in [176] use attention layers to generate inputs to the decoder as weighted outputs of the encoder. [169]'s model applies the attention mechanism right before making predictions. Similarly, researchers in [178] apply attention-enhanced LSTM with features extracted from images to forecast PM2.5 values. Tu et al. [177] improve the mechanism by adding a "time decay factor," making the model give more attention to recent time points and reducing the impact of earlier information. Unlike traditional attention methods, they also utilize hidden states of previous steps in the decoder during the attention process, allowing the model to react to changes even in the predicted future.

Attention mechanisms also improve spatial graph neural networks by adaptively weighting other locations based on the current input. Huang et al.[179] propose a spatial attention operator based on graph attention networks [167] and embed it into recurrent neural networks. They enhance attention by introducing a self-loop normalized adjacency matrix constructed based on geographic distances, providing additional geometric information. The study [180] proposes an attention-based parallel neural network architecture for PM2.5 prediction, where attention is applied to both temporal and spatial dimensions simultaneously, allowing the model to extract spatial and temporal features concurrently before combining them for further processing.

Summary. Attention mechanisms are generally used to support recurrent neural

networks in the temporal dimension and as advanced GNN architectures. In domains such as natural language processing, self-attention has completely replaced recurrent mechanisms. Theoretically, this can also be applied to time-series data in areas such as air quality prediction. Combining this temporal attention with spatial attention will be a consideration for our research. Furthermore, attention mechanisms provide a great tool for *explainable machine learning*, as they estimate the importance of factors contributing to predictions.

3.3 Physics-Informed Neural Networks

Physics-informed neural networks (PINNs) have recently attracted significant interest as a promising approach for addressing problems involving partial differential equations (PDEs), as proposed in the conventional PINN framework by [43]. These frameworks have demonstrated their effectiveness in solving both forward [181] and inverse problems [96, 182] related to simple dynamical systems governed by ordinary differential equations (ODEs).

3.3.1 Training Techniques

While PINNs have proven effective in various scenarios, challenges remain in their application to complex systems, particularly those exhibiting nonlinearities, multi-scale behaviors, or chaotic dynamics [41]. To address these issues, advanced techinques, including loss re-weighting [42, 101, 102, 183], data re-sampling [103–106], and domain decomposition [111, 112] have been proposed.

Normalization

Normalization is a crucial yet often overlooked step in the training of PINNs. Moseley et al. [113] employed a strategy involving the division of input domains and applying individual input normalization beside a unified global output normalization within their model computations. In the work [183], the authors recommended not only nor-

malizing the inputs and outputs of the neural networks but also non-dimensionalizing the differential equations in the objective functions. In the studies by [96, 184] and [122], input and output scaling layers were added that multiply the inputs and outputs by their average magnitudes, which occurs at the model level and thus affects the objective functions and training efficiency.

Loss Re-weighting

PINNs is a multi-task learning framework, incorporating multiple losses for data fidelity and adherence to physical laws. Due to the different scaling and convergence rates of these losses, imbalances can arise, potentially leading the model to converge to incorrect solutions as one objective may disproportionately dominate the training process. A common solution is the re-weighting of losses to achieve more balanced training. Wang et al. [42] demonstrated that one of the primary training challenges in PINNs stems from imbalanced gradients propagated from different losses. They proposed an adaptive approach that adjusts the weights of the losses based on the ratio between the maximum gradient magnitude of the physics loss and the mean gradient magnitude of the data loss with respect to the model parameters. Similarly, authors in [183] used the ratio of the L_2-norm of the gradients, while researchers in [101] balanced the variances of the gradients. In the work [102], they applied the Neural Tangent Kernel to demonstrate the faster convergence of physics law losses compared to initial and boundary condition losses, proposing an algorithm that equalizes their convergence rates by monitoring kernels of the losses.

Collocation Points

Collocation points (residual points), which are where physics constraints are minimized, are traditionally sampled uniformly at random from the input domain. While effective for simple systems, this approach is often not optimal for systems with steep derivatives. To address this, residual-based adaptive refinement (RAR), proposed by [103] iteratively identifies and incorporates new collocation points corresponding

to the highest differential equation residuals. This adaptive strategy directs model training toward the most challenging regions of the domain. Similar strategies have been explored in [104], where collocation points are sampled based on a probability distribution proportional to residuals, and in [105], where importance sampling approximates the distribution using the 2-norm of loss gradients. In the study [106], generative models were used to derive improved collocation point distributions.

Sequential training approaches, such as those in [108], divide the input domain into subdomains and use predictions from earlier subdomains as initial conditions for subsequent ones, while authors in [109] leverages predictions from all prior subdomains to enable a single global approximation network. A progressive learning framework is proposed in [107, 185], wherein collocation points are uniformly sampled from a dynamically expanding subdomain. This strategy respects time causality, enabling the accurate predictions of dynamical systems' evolution. Wang et al. [110] also emphasizes time causality by weighting the residuals to prioritize earlier time points.

Domain Decomposition

For large input domains, PINNs often struggle to converge effectively. Domain decomposition techniques alleviate this challenge by partitioning the domain into smaller subdomains and training separate PINNs for each. Continuity and smoothness across subdomain boundaries are enforced using additional interface losses. In the work [111], two interface conditions are added: one ensuring continuity of solution and the other enforcing conservation laws at the interfaces. For inverse problems, these conditions extend to parameter values at the interfaces. These methods are extended in [112] to accommodate general differential systems by ensuring the continuity of equations themselves. In studies by [113, 186], an implicit approach is adopted using a gating function. For practical applications where high-order smoothness is less critical, we simplify the approach in [112] by enforcing only value continuity at the interfaces. This strategy ensures initial condition consistency and maintains

continuity across subdomains without introducing unnecessary complexity.

Summary. Applying PINNs to complex systems that exhibit non-linearities, multi-scale behaviors, or chaotic dynamics remains a significant challenge. Advanced training techniques such as normalization, loss re-weighting, sampling of collocation points, and domain decomposition have been proposed to improve the training of PINNs. While these approaches have demonstrated a degree of problem solving, further improvements are necessary to make PINNs suited to highly multi-scale and complex dynamical systems, such as mosquito population modeling in [44]. Adjustments for more complex systems involve a *systematic* approach to normalization, enhancing loss weighting strategies to accommodate diverse equation scales and behaviors, incorporating more training phases to improve model initialization and convergence, and simplifying domain decomposition approaches.

3.3.2 Learning Parameters

PINNs have demonstrated a powerful capability for solving inverse problems, with numerous studies exploring their effectiveness in parameter estimation across various domains. An inverse problem, in this context, refers to the process of determining unknown parameters of a system from observed outputs. Traditionally, the approach involves substituting the unknown parameters with trainable variables if these parameters remain constant throughout the system, or replacing them with neural networks that depend on coordinates if the parameters vary with the system state. These trainable variables or neural networks are jointly trained with the neural networks that approximate the state variables [43, 182]. For example, Chen et al. [118] utilize this approach to solve inverse problems in nano-optics and metamaterials, whereas authors in [119] apply it to subsurface flow problems by replacing the unknown constitutive relationships with neural networks that take either time or the system state as inputs, depending on the assumed dependencies of these relationships.

Similarly, researchers in [120] identify parameters in civil structures by applying PINNs to solve inverse problems, incorporating physical constraints into the loss

function to ensure solutions remain physically and mechanically consistent. Nath et al. [187] identify unknown parameters and predict the dynamic behavior of a mean value model of a diesel engine. In addition to the main neural networks, empirical models of other unknowns are replaced with pre-trained neural networks trained beforehand with laboratory data. Furthermore, authors in [121] solve inverse problems in structural engineering by leveraging transfer learning to fine-tune pre-trained models for new training, thereby accelerating convergence and enhancing robustness when handling sparse or noisy data. In the study of [188], PINNs are applied to solve inverse problems in unsaturated groundwater flow, where the problem is reformulated into a double-loop structure: the outer loop optimizes unknown parameters using the global Cross Entropy algorithm [189], while the inner loop employs a conventional gradient-based algorithm to solve the equations.

Work in [190] demonstrates the use of PINNs for parameter estimation in reduced-order models of blood flow in the aorta. They employ a two-phase training procedure: initially training the state neural network with fixed initialized parameters, followed by simultaneous training of both the state and parameter networks. This splitting approach facilitates better adaptation of the network to physical laws before estimating specific parameters. Berardi et al. [191] apply inverse PINNs to transport models in porous materials, incorporating an additional term into the loss function to regularize the parameters toward reference values, thereby improving the accuracy of ill-posed inverse problems. Jagtap et al. [192] investigate solving inverse problems involving in supersonic flows, where they enforce positivity constraints on parameters by applying a maximum function between the output of the neural network and a small positive constant, similar to using a ReLU activation function. Method by [193] applies PINNs to infer the peridynamic kernel for a nonlocal wave equation, leveraging a radial basis function activation layer to enhance kernel shape approximation. Non-negativity and symmetry requirements on the kernel function are enforced by constraining trainable parameters and including symmetry terms in the loss function. Research by [96] integrates ODEs into neural networks to infer system dynamics

and estimate parameters for biological models, enhancing the neural networks with additional feature layers to capture characteristics like periodicity or exponential trends.

In nuclear reactor modeling, authors in [194] combined PINNs with the Theory of Functional Connections to solve stiff ODE systems, such as Point Kinetics Equations. Similarly, Daryakenari et al. [122] integrated PINNs with eXtreme Theory of Functional Connections and symbolic regression to estimate parameters and identify missing physics in systems biology. In this framework, PINNs help integrate partial prior knowledge about ODE systems while discovering unknown functions and estimating parameters. The values derived from the neural networks after training are utilized to derive mathematical formulas via symbolic regression. Similar methodologies are employed by [195, 196], and [197] to solve inverse problems involving the Lorenz system, the Allen-Cahn equation, and reaction-diffusion models related to Alzheimer's disease, respectively.

Summary. The application of PINNs for solving inverse problems across various domains relies on techniques reviewed in Section 3.3.1, including normalization, loss balancing, and domain decomposition, with only minor adaptations. Most studies solve inverse problems by directly learning the mapping from coordinates to parameters, without incorporating external forces that influence the dynamics of the system. However, research should consider the external forces that influence internal system parameters and jointly learn the mapping from external factors to system parameters alongside the system state. This approach would not only enable neural networks to learn the relationships between external factors and the system parameters but also allows the model to be reused under new data, which is not the case with current methods.

3.4 Conclusions

In this chapter, we presented a comprehensive review of the literature on the deployment of neural networks and graph networks in real-world applications, in

addition to the physics-informed neural networks. We examined how machine learning and neural networks have been applied to disease prediction using exosomes and to oxygen uptake estimation. We also explored the construction and analysis of complex networks in bike-sharing systems and investigated neural network architectures for air quality forecasting. Furthermore, we delved into the techniques needed when applying PINNs to complex systems with nonlinearities and multi-scale behaviors, discussing advanced training techniques such as normalization, loss re-weighting, adaptive sampling, and domain decomposition for both forward problems and the learning of dynamical system parameters.

Despite significant advancements, several gaps exist in current research across multiple domains. In exosome SERS signal classification, it remains unexplored whether neural networks can discriminate exosomes from normal and dysfunctional cells. In oxygen uptake estimation, machine learning has improved accuracy using data from wearable sensors and advanced neural networks. However, a systematic exploration of how different combinations of inertial measurement unit sensors, data representations, and neural network architectures affect VO_2 estimation accuracy is lacking. In Chapter 4, we will address both of these issues through: the application of neural networks; and by experimenting with various IMU sensor combinations, exploring data representations, and evaluating different neural network models to identify optimal configurations for VO_2 estimation.

In terms of complex network approach, graph-based analysis has been applied in bike-sharing systems, but a framework for network construction and optimization is lacking, especially correlation-based networks that can reveal spatiotemporal patterns. In Chapter 5, these limitations are addressed by developing a framework for network analysis, developing a clustering-based approach for time-step analysis, and extending network techniques to more correlation-based networks. The same chapter will be used to experiment with attention mechanisms to address the computational inefficiencies of RNNs for long sequences and also to address the relianced by GNNs on static adjacency matrices that fail to capture dynamic spatial relationships.

In Chapters 6 and 7, we will address the limitations of PINNs by introducing a normalization procedure, enhancing loss weighting strategies, incorporating additional training phases, and simplifying domain decomposition approaches. We will also take into account the effect of external factors on internal system parameters, learning the mappings from these factors to the parameters.

Chapter 4

Deployment of Neural Networks in Real-Life Applications

In this chapter, we investigate the suitability of neural networks in two applications from two separate domains. In the first experiment, we investigate the application of neural networks in a classification problem from exosome signals. Exosomes are small extracellular vesicles secreted by all cell types. They have been widely indicated to carry biomarkers within their content that can be used for diagnostic and prognostic insight into disease within the sample. For this part of the research, we developed a three-step preprocessing procedure to process Surface-Enhanced Raman Spectroscopy Spectroscopy (SERS) signals, and then use Multi-Layer Perceptron (MLP) models to distinguish the heterogeneity of exosome signals that secreted from normal and dysfunctional human aortic endothelial cells.

In the second project, we further explore neural network variations in predicting the Oxygen uptake in simulated team sports with wearable sensor signals. In sports, precise evaluation of training status is essential for performance optimization and injury risk reduction. This second study investigates the use of various neural network architectures in estimating individual oxygen uptake (VO_2) from wearable sensor data during outdoor jogging and simulated team sports activities. The architectures examined include MLP, long short-term memory (LSTM) networks and convolutional

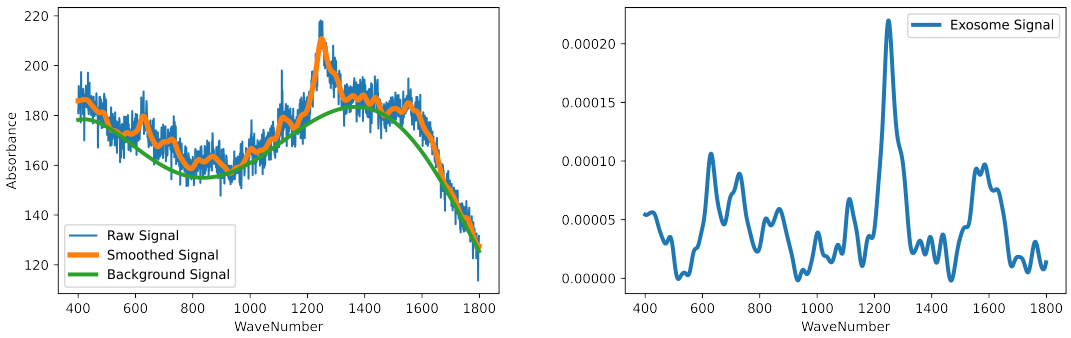
neural networks (CNN). These models are evaluated using both raw features and handcrafted features to assess their predictive performance.

4.1 Exosomes Classification Using Multi-Layer Perceptrons

Exosomes are extracellular vesicles produced by all cells into virtually all body fluids. Exosomes have been shown to have characteristics that reflect their originating cells and the cells disease status [48]. They are rapidly emerging as important diagnostic biomarkers. Raman spectroscopy and SERS offers a potentially powerful approach to extract molecular-level fingerprint signals of the biomaterial profile of exosome samples in diagnostic applications [198]. Interest in the application of SERS as a means of exosome characterization has grown over the past 5 years or so, and is critically enabled by using data analytical methods, capable of discriminating complex aggregate signals that exosome SERS give rise to.

Endothelial dysfunction is a critical early indicator of subclinical atherosclerosis associated with type 2 diabetes mellitus (T2DM) [199]. Due to the asymptomatic nature of early and intermediate T2DM, early diagnosis before complications manifest is a challenge. There is a need to develop methods for distinguishing between healthy vascular endothelial cells and their dysfunctional 'hyperglycaemic' counterparts [55]. Exosomes released from dysfunctional 'hyperglycaemic' endothelial cells offer an opportunity to diagnose early changes in cell functionality. This study aims to investigate the potential of neural networks to differentiate between exosomes secreted by normal and dysfunctional human aortic endothelial cells, cultured under normal, hyperglycemic and hypoglycemic conditions.

The scripts associated with the project is available at `https://github.com/dinhvietcuong1996/exosome-classification`.



(a) Smoothing & Background Estimation (b) Representative Smoothed Exosome Signal.

Figure 4.1: Preprocessing Steps

4.1.1 Methodology

In this section, we present our methodology for exosome spectrum prediction. Figures 4.1a illustrate the pre-processing steps for a spectrum. Initially, the raw input spectrum undergoes smoothing to mitigate random fluctuations, predominantly attributable to electronic instrumentation. Subsequently, we perform background signal subtraction using automated background approximation. The resultant exosome signal is then input into a machine learning model to classify the spectrum as originating from a healthy subject or one afflicted with hyperglycemia/hypoglycemia.

Data Pre-processing

In the pre-processing stage, the aim is to isolate the spectral signal originating solely from exosomes, which is hypothesized to be the key informative factor for distinguishing between normal and diseased spectra. This includes the removal of random noise and background signals.

The initial preprocessing step aims to mitigate the impact of random noise on the spectrum through the application of a simple moving average function. This method involves a sliding window that traverses the spectrum sequence, computing the average for each window position. The resulting average values are then concatenated to produce the final smoothed output. This process is mathematically represented by Equation 4.1, where W denotes the sliding window size, n represents the spectrum

length, and s_i corresponds to the smoothed value of the ith element. To maintain consistent window sizes throughout the spectrum, the function implements symmetric padding.

$$s_i = \frac{1}{W} \sum_{j=-\frac{W-1}{2}}^{\frac{W-1}{2}} x_{i+j}, 0 \leq i < n \tag{4.1}$$

The second preprocessing step approximates the background signal using the ModPoly polynomial method [200]. This iterative process eliminates peaks from the polynomial fit by replacing spectral data points with their corresponding estimated background values where the former exceeds the latter. The background estimation is then refit to the newly formed spectrum. This replacement and fitting cycle continues until the background estimation stabilizes or a predefined iteration limit is reached. Given a polynomial degree D, the algorithm can be described as having 4 steps:

1. Initialize $t = s$, where t serves a temporary holder for the spectrum that does not include Raman peaks in it and as input for the polynomial fitting to approximate the background.

2. Polynomial fitting: Let $p(i) = c_0 + c_1 i + \cdots + c_d i^d, 0 \leq i < n, c_d \in \mathbb{R}, 0 \leq d \leq D$ be a polynomial. Determine the coefficients $c_d, 0 \leq d \leq D$ that minimize the squared error between p and t:

$$E = \sum_{i=0}^{n-1} (p(i) - t_i)^2 = \sum_{i=0}^{n-1} \left(\sum_{d=0}^{D} c_d i^d - t_i \right)^2 . \tag{4.2}$$

   The optimization solution is obtained through standard linear algebra techniques [201]. The resulting polynomial p is considered an approximation of the background.

3. Update $t_i = \min(t_i, p(i)), 0 \leq i < n$, replacing values in t with the background estimation p where $t > p$, thereby eliminating peaks in t for subsequent background estimations.

4. Terminate the algorithm when $t = p$ for all i or when the iteration count

exceeds a predetermined threshold; otherwise, repeat steps 2 to 4.

The final polynomial p represents the estimated background for the given s. The resultant exosome signal x is computed as the difference between the smoothed signal and the estimated background: $x = s - p$.

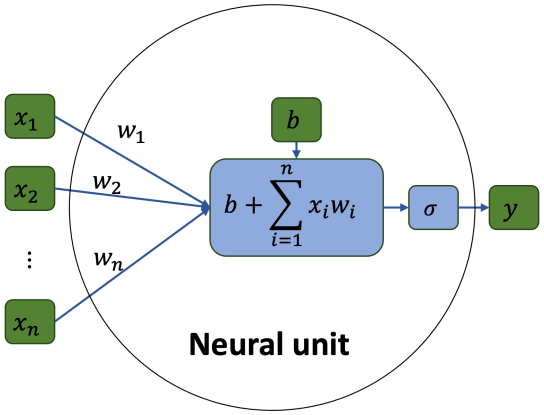
Before inputting into machine learning models, the spectrum must be normalized. This step is necessary due to the differences in the intensity across spectra while collecting the data, thereby facilitating the subsequent training of machine learning models. Normalization is achieved by scaling each spectrum such that its integral equals 1. Figure 4.1b shows an example of a normalized, background-removed spectrum.

Neural Network Architecture

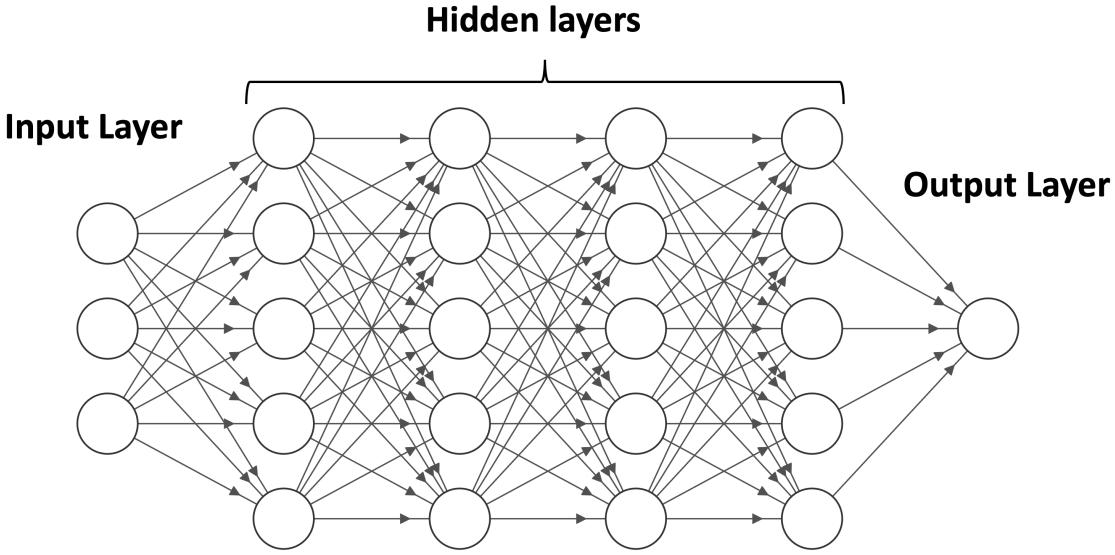
For the predictive model, we implement an MLP, which is also called a feed forward fully-connected neural network [202]. The model has been proven to approximate any functions [30] and demonstrated its capabilities in various domains.

Illustrated in Figure 4.2, a neural network is a directed graph where nodes are neural units and the connections between them are the flow of information. Each neuron (neural unit), depicted in Figure 4.2a, acts as a computational unit that processes inputs and generates outputs. Each node's output is a function of the weighted sum of its inputs, followed by a nonlinear activation function. The non-linearity enables MLPs to approximate highly nonlinear functions, enhancing their flexibility. n example of an MLP architecture is shown in Figure 4.2b. The MLP architecture organizes neurons into distinct layers: input layer receives the initial data vector; hidden layers process information from previous layers; and output layer produces the model's final result.

In this task, the MLP model is used to estimate the conditional probability $P(c|x)$, where x represents a pre-processed spectrum vector of size $1 \times n$, and $c \in C$ denotes one of three classes: healthy, hyperglycemia, or hypoglycemia. The input layer in used solely to pass the input vector to the network while the output layer



(a) Computation within a neural unit that processes n inputs $x_i, i = 1, \ldots, n$ to produce an output y.



(b) An MLP architecture comprising an input layer with 3 units, four hidden layers each containing 5 units, and an output layer with a single unit.

Figure 4.2: Multi-Layer Perceptron Architecture

delivers the outputs (exosome classifications) of the MLP, again represented as a single vector. Let l_i denote the number of units in the i-th layer, where $0 \leq i \leq L+1$. Specifically, $l_0 = n$ is the input layer size, while $l_{L+1} = 3$ corresponds to the output

layer size. The neural network is formulated in Equations 4.3.

$$\begin{aligned} a^0 &= x \\ a^i &= \max\left(a_{i=1}W^i + b^i, 0\right), 1 \leq i \leq L \\ z &= a_L W^{L+1} + b^{L+1} \\ P(c_i|x) &= \frac{e^{z_i}}{e^{z_0} + e^{z_1} + e^{z_2}}, i = 0, 1, 2 \end{aligned} \tag{4.3}$$

Here, $W^i \in \mathbb{R}^{l_{i-1} \times l_i}$ represents the real-valued weight matrix for layer i, $b^i \in \mathbb{R}^{l_i}$ denotes the bias vector for layer i, and $c_i, i = 0, 1, 2$ correspond to the three classes. The max function operates element-wise, resulting in $z \in \mathbb{R}^3$. The network parameters W^i and b^i are optimized to minimize the cross-entropy loss between predicted probabilities and target distributions. We employ the Adam optimizer, a variant of stochastic gradient descent, for parameter optimization [203]. To enhance model generalization, we apply L2 regularization with a penalty parameter α.

4.1.2 Experiments

The dataset used in these experiments is presented in Section 2.1.1. In this section, we describe experimental configurations and a discussion around results.

Model Selection

The aim of model selection is to identify the best performing neural network configuration. As the dataset is relatively small, we conduct 8-fold cross validation. For each iteration, one subset serves as the validation set, while the remaining 7 subsets are used as the training data. Therefore, in each turn (or fold), 6 surfaces or 300 spectra (100 per class) are used for validating and the neural network model is trained with the remaining 2,100 spectra (700 per class).

The following hyper-parameters are considered for tuning: the smoothing window size W; the background polynomial degree D; the number of hidden layers L; and the weight decay rate α. The goal is to ensure that the average validation accuracy across

all folds, is as high as possible. Table 4.1 displays all attempted hyper-parameter values whose ranges were predetermined using an initial round of experiments. A grid search over all values, which is 1,260 combinations, was conducted to find the best performing models.

Table 4.1: Hyper-Parameter Settings

Hyper parameters	Values
W	9; 17; 33; 65; 129
D	5; 6; 7; 8; 9; 10; 11
L	0; 1; 2; 3
α	10^{-6}; 10^{-5}; 10^{-4}; 10^{-3}; 10^{-2}; 10^{-1}; 1; 10; 100

Results

We use accuracy, recall, and precision metrics [204] (defined in Appendix A.2) to evaluate a model's performance. Macro-averaging is used across multiple classes.

Table 4.2 presents the top 5 models ranked by mean validation accuracy. The highest performing models appear to perform equally at about 65% with a standard deviation of 11%. Across all experiments, validation accuracy ranged from 50% to 82%, showing substantial variability. This variance can be attributed to the limited size of the dataset. From the experiments, the best performing models suggest that the MLP should have 2 or 3 hidden layers of 100 units.

Table 4.2: Top Performing Models by Average Accuracy

Rank	W	D	L	α	Training Acc.	Validation Acc.
1	33	10	2	10.0	0.991 ± 0.004	0.652 ± 0.100
2	17	6	3	10^{-5}	1.0 ± 0.0	0.651 ± 0.102
3	33	6	2	10^{-4}	1.0 ± 0.0	0.650 ± 0.117
4	9	6	2	10^{-5}	1.0 ± 0.0	0.647 ± 0.110
5	7	10	3	10^{-1}	0.985 ± 0.007	0.646 ± 0.120

Observations and Discussion

We now use the accuracy metrics to construct 2 tables which enable a more detailed analysis of the best performing model.

Table 4.3 presents the validation confusion matrix as the sum of all confusion matrices for the 8 validation sets described in the previous section. Meanwhile, Table 4.4 presents the test confusion matrix resulting from the model that performs best in validation. Our discussion focuses on the Table 4.4 with the test performance on unseen data. The results demonstrate that the most accurate neural network model performs well when classifying normal samples and achieves exceptionally high accuracy with hypoglycemic samples. However, the model encounters difficulty in distinguishing between normal and hyperglycemic samples, which is most likely due to the presence of normal tissue in the hyperglycemic samples, a frequent byproduct of the sample generation process. This observation raises the possibility that the model's predictive performance could be significantly higher if training and testing were limited to hypoglycemic and hyperglycemic samples only, although we cannot be sure without removing normal samples from the test data.

Table 4.3: Validation Confusion Matrix (Sum over folds). The rows represent the true class labels, while the columns correspond to the model's predicted labels. For instance, the entry of 181 in the "Normal" row and the "Hypo" column indicates that the model has misclassified 181 normal samples as hypoglycemic.

	Normal	Hypo	Hyper	Total	Recall
Normal	443	181	175	799	0.551
Hypo	93	611	96	800	0.764
Hyper	158	132	510	800	0.638
Total	694	924	781		
Precision	0.638	0.661	0.653	Accuracy:	0.652

Table 4.4: Test Confusion Matrix (Best Model). The rows represent the true class labels, while the columns correspond to the model's predicted labels. For instance, the entry of 3 in the "Normal" row and the "Hypo" column indicates that the model has misclassified 3 normal samples as hypoglycemic.

	Normal	Hypo	Hyper	Total	Recall
Normal	66	3	31	100	0.660
Hypo	5	81	14	100	0.810
Hyper	40	7	53	100	0.530
Total	111	91	98		
Precision	0.595	0.890	0.541	Accuracy:	0.667

Precision is a measure of exactness of positive predictions. The results for the

Normal class shows a true positive rate of 66% and a false positive rate of 45%, yielding a precision value of 59.5%. a true positive figure of 66% and a false positive figure of 45%, resulting in a precision value of 59.5%. Notably, an overwhelming majority (89%) of false positives stem from misclassifications of Hyperglycemia. This provides further detail on where incorrect Normal predictions occur but perhaps, strengthens the case for the hypoglycemia class being more detectable.

Among the three classes, hypoglycemia predictions are the least frequent, suggesting that while these samples may be more challenging to detect, its predictions are less likely to to be incorrect. In the Hypoglycemia column, the class has a true positive figure of 81 with a false positive figure of just 10, yielding a precision of 89%. The majority (70%) of the false positives are attributed to the other diseased class. While these results are high, it would also indicate that very strong results can be expected if the model are presented exclusively with hypoglycemia and normal samples. For Hyperglycemia, we achieved lowest overall predictive performance. In contrast, hyperglycemia demonstrates the lowest overall predictive performance. This class achieves a true positive value of 53% and a false positive rate of 44%, resulting in a precision of 54%. The correlation with the Normal class is again evident, as 31 (70%) of the incorrect predictions are misclassified for Normal samples.

Recall is a measure of completeness defined as the percentage of positive samples that are correctly classified as positive. For the Normal class, a low overall sensitivity of 66% is observed, primarily attributable to the misclassification of 31% of Normal samples as hyperglycemia Only 3 of the samples are incorrectly classified as hypoglycemia. The hypoglycemia class has more favorable results, with a high overall score of 81%. The main source of error in this class is the misclassification of 14% of hypoglycemic samples as hyperglycemia. Lastly, the sensitivity for hyperglycemia is low at 53%, driven by a significant proportion (40%) of samples being incorrectly classified as Normal.

Although our best model achieves 66.7% accuracy, which is promising as a first demonstration of neural networks for SERS-based exosome classification of healthy

vs. hyperglycemic and hypoglycemic states, it is not yet clinically practical. Other SERS exosome studies report 90–95% accuracy [51, 52, 139], but those typically target different diseases (often cancer). This result thus serves as an early benchmark, and future efforts, expanding and refining the data collection, and exploring advanced model architectures, are expected to improve performance.

Summary. For this experiment, we developed a methodology for predicting exosome spectra to classify samples as originating from healthy subjects or those with hyperglycemia or hypoglycemia. The approach involves pre-processing steps including smoothing the raw spectra, background removal using the ModPoly method, and normalization. A multi-layer perceptron was trained for the classification, achieving an average validation accuracy of approximately 65%, with high precision and recall in detecting hypoglycemia but had difficulties in distinguishing between normal and hyperglycemic samples.

4.2 Predicting Oxygen Uptake in Athletes

In team sports, accurately assessing a player's physical output during games and training is crucial for coaches and sports scientists [205]. Physical adaptations that are typically measured through fitness testing are often impractical during competitive periods [206]. Finding a convenient way to monitor these changes is essential to accurately assessing training loads and tracking athlete fitness [207].

Wearable devices such as Inertial Measurement Units (IMUs) have demonstrated potential in estimating oxygen consumption (VO_2) across various physical activities [63, 64]. However, two challenges [60] still remain: firstly, determining optimal IMU sensor configurations to effectively capture and quantify complex movements, thereby enhancing VO_2 estimation accuracy; and secondly, identifying suitable models for learning relationships between sensor-based measures and VO_2 values.

This second experiment aims to address these challenges by investigating the prediction of individual oxygen uptake during outdoor running and simulated team sports activities. We explore the efficacy of different data representations, including

handcrafted features and raw sensor data, as well as various sensor combinations positioned at different body locations. Furthermore, we conduct a comparative analysis of multiple machine learning models to identify the most suitable algorithms for this task.

The source code for this project is hosted on GitHub at `https://github.com/dinhvietcuong1996/oxygen-uptake-estimation`.

4.2.1 Data Preprocessing

Data collection is described in Section 2.1.2. This section presents the data preprocessing for the experiments.

One of the goals in the experiments is to examine the influence of sensor placement configurations on predictive accuracy. For this experiment, we constructed four datasets, each comprising common baseline variables including the subject's age, height, weight, heart rate, breathing rate, the treadmill speed and the GPS speed, activity categories (one of resting, treadmill running, outdoor running, or simulated team sports activities), These datasets differ in their inclusion of IMU sensor data from various on-body locations:

- Dataset A: IMU Torso
- Dataset B: IMU Torso + IMU Arm
- Dataset C: IMU Torso + IMU Leg
- Dataset D: IMU Torso + IMU Arm + IMU Leg

For each of the four configurations above, we develop two distinct data representations for model inputs: RAW and MAD (Mean Amplitude Deviation) dataset. The earlier representation is basically raw data with minor preprocessing while the later one significantly transform the data into more information-condensed variables. Both representations are organized into breath-based windows, typically spanning 2 to 6 seconds. We use 7 breaths (about 30 seconds) to predict the VO_2 value of

the central breath. The key distinction lies in the sensor data: in MAD, all sensor readings are transformed into mean amplitude deviation values, as described by the formula in Equation 4.4. For the target variable, VO_2 values undergo smoothing using a 31-point moving average window to mitigate interference noise [67]. This preprocessing step reduces interference noise.

$$MAD_{xyz} = \sqrt{\left(\frac{1}{N}\sum_{i=1}^{N}|x_i - \overline{x}|\right)^2 + \left(\frac{1}{N}\sum_{i=1}^{N}|y_i - \overline{y}|\right)^2 + \left(\frac{1}{N}\sum_{i=1}^{N}|z_i - \overline{z}|\right)^2} \tag{4.4}$$

In total, there are four dataset configurations, each of which can be combined with two different input representations. This results in a total of eight combinations that can be inputted into the predictive models.

4.2.2 Neural Networks

In this experiment, we evaluate the effectiveness of various neural network architectures for our specific task, encompassing models from simple linear regression, multi-layer perceptrons (MLPs) to more complex structures such as, long short-term memory networks (LSTMs), and convolutional neural networks (CNNs).

The input to the models, denoted as x, is structured as a matrix with dimensions $[n_b \times n_f]$. The number of breaths n_b is set to 7, while the number of features n_f varies from 18 to 32, depending on the dataset and chosen representations. Let F_W represent a neural network with parameters (weights and biases) W. The model's output is given by $\hat{y} = F_W(x)$. We train these weights W by minimizing the mean squared error between the predicted values $\hat{y}$ and the actual values y, as described in Equation 4.5.

$$\mathcal{L} = \frac{1}{N_{\text{dataset}}} \sum_{(x_i, y_i) \in \text{dataset}} (y_i - F_W(x_i))^2 \tag{4.5}$$

For optimization, we use the Adam algorithm [203], a gradient-based method. To mitigate overfitting, we apply L2 regularization which favors weights with smaller

norms. We continue this section with a more detailed discussion of the neural network architectures used in the experiment.

Linear Regression (LR)

LR is a straightforward yet powerful model commonly employed in various problems [208]. It characterizes the relationship between dependent and independent variables by fitting a linear function to the observed data. In this model, each breath is treated uniformly by flattening the input x to a vector of size $n_b \times n_f$.

$$\hat{y} = x \cdot W + b \tag{4.6}$$

To make predictions, the model makes a linear transformation from $\mathbb{R}^{n_b \times n_f}$ to $\mathbb{R}$ by multiplying the input with a weight matrix of size $\mathbb{R}^{n_b \times n_f} \times 1$. Additionally, a scalar bias term b is added to enhance the model's flexibility. The output is defined in Equation 4.6.

Multi-Layer Perceptron

For this task, the MLP extends the capabilities of linear regression by introducing non-linearity and additional complexity to the model [209]. This is achieved by incorporating multiple layers, where each layer consists of a linear transformation followed by a non-linear activation function. Here, we adjust the model described Section 4.1.1 to accommodate the breath dimension in the input x, the input undergoes a linear transformation that maps it to a common latent space of dimension d_{lat}. Subsequently, the transformed data are flattened into a vector of size $n_b \cdot d_{\text{lat}}$, which serves as the input to the MLP layers.

Specifically, let $d_1, d_2, \ldots, d_L$ represent the sizes of the hidden layers $1, 2, \ldots, L$ in the MLP, where $d_0 = n_b \cdot d_{\text{lat}}$ is the dimension of the input vector to the MLP and $d_L = 1$ is the output size. The parameter matrix W_{lat} for the initial linear transformation is of size $n_f \times d_{\text{lat}}$, and $W^{(l)}$ denotes the parameter matrices of the

MLP layers, with each matrix having dimensions $d_{l-1} \times d_l$. The biases b_{lat} and $b^{(l)}$ correspond to sizes n_b and d_l, respectively. The function ϕ represents an element-wise non-linear activation function employed within the layers. The prediction of the model is defined in Equations 4.7.

$$\begin{aligned} x_0 &= \text{flatten}\left(x \cdot W_{\text{lat}} + b_{\text{lat}}\right) \\ x^{(l)} &= \phi^{(l)}\left(x^{(l-1)} \cdot W^{(l)} + b^{(l)}\right), l = 1, 2, \ldots, L-1 \\ \hat{y} &= x^{(L-1)} \cdot W^{(L)} + b^{(L)} \end{aligned} \tag{4.7}$$

In our experiments, a grid search strategy is implemented to fine-tune the architecture of the MLP. The hyper-parameter settings are detailed in Table 4.5, where the number of layers varies from 1 to 4 and the number of neurons per layer is set to either 32 or 64. The activation function employed is rectified linear unit (ReLU). To reduce overfitting, a weight decay coefficient of 10^{-4} is used, though no dropout is applied.

Table 4.5: MLP Hyper-parameters

Hyper-parameter	Values
Number of layers	$1; 2; 3; 4$
Hidden layer size	$32; 64$
Weight decay	10^{-4}

Long Short-Term Memory (LSTM)

LSTM, a variant of recurrent neural networks, is specifically designed to process sequence data similar to time series [163]. Figure 4.3 illustrates a typical LSTM layer. This architecture uses recurrent mechanisms where a common LSTM cell sequentially processes the input x_t at each step t, producing the output h_t and passing the hidden cell state c_t alongside h_t to the next time step. The core idea of LSTM is its cell state c which carries temporal information across time steps. This state is controlled by several gates within the LSTM cell that either add or remove information based

on current inputs and previous outputs.

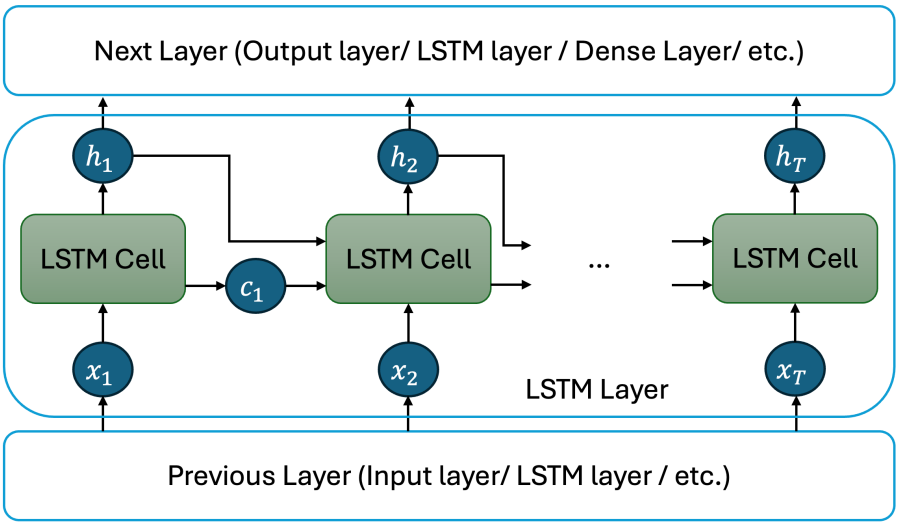


Figure 4.3: **LSTM Layer Architecture**. The LSTM cell is shared across time steps. At each time step t, the cell receives the input x_t from the previous layer and the hidden state h_{t-1} from previous time step. It produces the new hidden state h_t, which is passed to the next layer, as well as the updated cell state c_t. The initial state c_{-1} and output h_{-1} are initialized as zero vectors.

Let d_h be the hidden size of a LSTM layer. Consider $x_1, x_2, \ldots, x_T$ as the inputs at time steps $1, 2, \ldots, T$ respectively, where each input vector is of dimension d_f. The LSTM layer parameters are organized as follows: the input gate has a weight matrix W_i and a bias vector b_i; the forget gate has a weight matrix W_f and a bias vector b_f; the output gate has a weight matrix W_o and a bias vector b_o; and the cell state is updated by a weight matrix W_c and a bias b_c. All of the weight matrices has the size of $d_h \times (d_h + d_f)$ and all of the biases are of size d_h. Let σ denote the sigmoid function; $[;]$ be the concatenation operator and $*$ be element-wise multiplication. At time step t, the input gate i_t, the forget gate f_t, the output gate o_t, new candidate value $\tilde{c}_t$, the cell state c_t and output h_t of the LSTM are defined in Equations 4.8.

$$
\begin{aligned}
i_t &= \sigma\left([x_t; h_{t-1}] \cdot W_i + b_i\right) \\
f_t &= \sigma\left([x_t; h_{t-1}] \cdot W_f + b_f\right) \\
o_t &= \sigma\left([x_t; h_{t-1}] \cdot W_o + b_o\right) \\
\tilde{c}_t &= \sigma\left([x_t; h_{t-1}] \cdot W_c + b_c\right) \\
c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t \\
h_t &= o_t * \tanh\left(c_t\right)
\end{aligned}
\tag{4.8}
$$

In our experiments, the breath dimension is treated as the time dimension, hence fixing the dimension to 7. We employ a bidirectional LSTM architecture (BiLSTM), allowing information flow in both directions across the time dimension. Furthermore, the LSTM layers are stacked such that each LSTM layer's output at a time step serves as the input for the subsequent layer at the same step. The final output from the middle time step of the last LSTM layer is passed to a fully connected layer for generating predictions. The number of layers and layer size are tuned through a grid search with parameters specified in Table 4.6, which explores configurations ranging from one to four layers, each with 32 or 64 units. No dropout is employed; instead, regularization is achieved through a weight decay of 10^{-4} to mitigate overfitting.

Table 4.6: LSTM Hyper-parameters

Hyper-parameter	Values
Number of layers	$1; 2; 3; 4$
Hidden layer size	$32; 64$
Weight decay	10^{-4}

Convolutional Neural Network (CNN)

CNNs are neural network architectures designed to process structured data efficiently [210]. In particular, one-dimensional CNNs (1D-CNNs) are well-suited for sequential data processing [211]. Figure 4.4 depicts the architecture of the one-dimensional CNN layers. Each CNN layer consists of multiple kernels that slide along

the temporal dimension, performing convolutional operations between the kernel and segments of the input to extract local features. The core principle of CNNs is that shared kernels evaluate how well different segments of the input match specific patterns at each time step. This convolutional mechanism enables the network to detect local patterns regardless of their position within the input sequence, thereby enhancing the effectiveness of 1D-CNNs for sequential data.

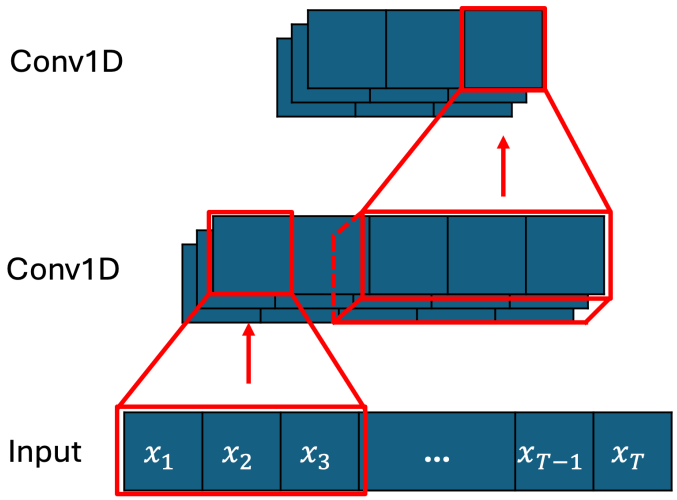


Figure 4.4: An illustration of one-dimensional CNN architecture. In the convolutional layers, three kernels, each of size 3, slide along the time dimension of the input to generate three feature channels. Red boxes and arrows at the bottom left demonstrate a convolution computation involving x_1, x_2 and x_3 that results in a single output.

Consider the input sequence $x = (x_1, x_2, \ldots, x_T)$ to a 1D-CNN layer, x_t corresponds to time step t and have a dimensionality of d_f. Assume the layer has n_k kernels of size d_k, represented by a tensor K of dimensions $n_k \times d_k \times d_f$. Let σ denote the activation function. The output of the 1D-CNN layer is defined as in Equation 4.9, where $y_{t,i}$ is the output at time step t for the i-th kernel. Indices that are less than 0 or exceed the sequence length are defaulted to 0 (commonly referred to as zero padding).

$$y_{t,i} = \sigma\left(\sum_{k=0}^{d_k-1}\sum_{j=1}^{d_f} K_{i,k,j} x_{t+k-\left\lfloor \frac{d_k}{2} \right\rfloor, j}\right) \tag{4.9}$$

In our experiment the CNN architecture treats the *breath* dimension as the *time* dimension. In addition, multiple CNN layers are stacked to enable the network to

learn more abstract temporal patterns. The output at the middle time step of the last CNN layer is fed into a fully connected layer to produce the final predictions. Various CNN configurations are explored via grid search to tune hyperparameters such as the number of layers and the number of kernels. Values for the hyperparameters are listed in Table 4.7. The CNN architectures vary in depth, ranging from 3 to 5 1D-CNN layers, and in width, using either 32 or 64 kernels per layer. A fixed kernel size of 3 is employed across all layers, which allows for the capture of local temporal patterns while the increasing depth enables the learning of broader contextual information. The activation function used is ReLU. Consistent with other models, we employ L2 regularization with a weight decay parameter of 10^{-4}.

Table 4.7: 1D-CNN Hyper-parameters

Hyper-parameter	Values
Number of layers	$3; 4; 5$
Hidden layer size	$32; 64$
Kernel size	3
Weight decay	10^{-4}

4.2.3 Results

Table 4.8 presents the top 15 performing configurations of the machine learning models, ranked by validation RMSE. From the table, it can be seen that the MLP model using the MAD representation and dataset A achieves the best performance on the validation set, with a RMSE of 3.18 and MAE of 2.26. However, this model does not generalize well to the test set, where it records an RMSE of 7.65 and MAE of 5.74. Conversely, the LSTM model with RAW representation and dataset A attains the best results on the test set, achieving an RMSE of 4.98 and MAE of 3.70, while maintaining comparatively low errors on the validation set (RMSE of 4.11 and MAE of 3.30). Overall, all models except for LR demonstrate strong performance on the validation set, but only BiLSTM excels on the test set. This suggests that while MLP, CNN, and BiLSTM are effective during training, BiLSTM may offer better generalizability when applied to unseen data. Data representation also plays an

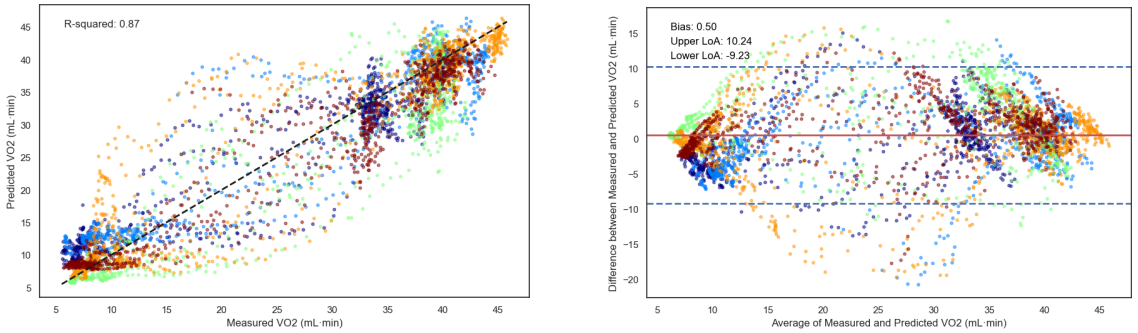
important role. The MAD representation generally leads to better validation results (ranking in the top 1–8 positions) but yields poorer results on the test set (test RMSE greater than 6.6). In contrast, the RAW representation results in slightly lower performance on the validation set (top 9–14, RMSE ranging from 3.97 to 4.20) but provides superior performance on the test set, with RMSE values ranging from 5.0 to 5.7. For instance, the LSTM model using RAW representation and dataset A achieves a test RMSE of 4.98, and the MLP model with datasets A and C achieves test RMSEs of 5.74 and 5.70, respectively. This indicates that the choice of data representation significantly affects a model's ability to generalize. The RAW representation without of handcrafted features allows deep models to extract generalizable features, whereas the MAD representation may lose information compared to RAW, leading to poorer generalization. Regarding sensor configurations, dataset C—which includes additional sensors on the leg—provides the best performance for estimating oxygen consumption when using the RAW representation and models such as LSTM or MLP. This implies that these additional sensors capture patterns that generalize well. The variability in validation performance across datasets may indicate that models are highly overfitting to the validation set; however, overall, dataset C yields slightly better test performance.

Figure 4.5a shows the relationship between predicted VO_2 and measured VO_2 for the LSTM model using RAW data representation and dataset C. The coefficient of determination, R^2, is 0.87, indicating a strong correlation between the predicted and measured VO_2 values. This high R^2 value suggests that the LSTM model explains 87% of the variability in the measured VO_2 data. The predicted values consistently align closely with the measured values, demonstrating good model performance. Figure 4.5b presents a Bland-Altman plot showing the difference between measured and predicted VO_2 values against their average. This plot helps identify any systematic bias and the limits of agreement (LoA) for the predictions. The calculated bias is 0.50 mL/min, indicating a slight overprediction on average. The upper LoA is 10.24 $mL \cdot kg^{-1} \cdot min^{-1}$, and the lower LoA is -9.23 $mL \cdot kg^{-1} \cdot min^{-1}$.

Table 4.8: **Top-15 performance of neural network models.** This table presents the top 10 performing neural network models, ranked by their valid RMSE. The table includes both RMSE and MAE metrics for the validation and test sets. The unit of the metrics is $\mathrm{mL} \cdot \mathrm{kg}^{-1} \cdot \mathrm{min}^{-1}$. The best results, corresponding to the smallest error values, are highlighted in **bold**, while the second-best results are underlined.

Rank	Dataset	Data Repr.	Model	Valid RMSE	Valid MAE	Test RMSE	Test MAE
1	A	MAD	MLP	**3.18**	**2.26**	7.65	5.74
2	D	MAD	MLP	3.25	2.34	6.83	5.30
3	B	MAD	MLP	3.30	2.38	7.00	5.34
4	C	MAD	CNN	3.34	2.55	6.67	4.98
5	B	MAD	CNN	3.41	2.59	6.87	5.12
6	C	MAD	MLP	3.65	2.74	6.64	4.89
7	A	MAD	CNN	3.66	2.74	7.29	5.76
8	D	MAD	CNN	3.67	2.50	7.34	5.46
9	B	RAW	MLP	3.97	3.00	7.75	5.85
10	D	RAW	BiLSTM	3.98	3.07	5.94	4.38
11	C	RAW	BiLSTM	4.11	3.30	**4.98**	**3.70**
12	A	RAW	CNN	4.13	3.23	6.10	4.59
13	A	RAW	MLP	4.13	3.08	5.74	4.33
14	C	RAW	MLP	4.20	3.35	5.70	4.33
15	C	MAD	BiLSTM	4.28	3.49	7.94	5.90

This range reflects the spread of differences between measured and predicted values. Most data points lie within these LoAs, suggesting that the model's predictions are generally accurate.



(a) Linear Correlation plot

(b) Bland-Altman plot

Figure 4.5: (a) Linear correlation plot illustrating the relationship between predicted VO_2 and measured VO_2 using the LSTM model with RAW representation and sensor configuration C, achieving an R^2 value of 0.87. (b) Bland-Altman plot showing the differences between measured and predicted VO_2 values against their averages for all subjects combined.

Figure 4.6 illustrates the residuals (predicted VO_2 minus measured VO_2) across different exercise conditions for the LSTM model using RAW data representation and dataset C. The exercise conditions include baseline, jogging, recovery1, circuit1,

recovery2, circuit2, and recovery3. The box plots reveal that the model generally exhibits a median residual close to zero across the different exercise conditions, indicating minimal prediction bias. However, there is noticeable variability in the residuals, particularly during the recovery phases. The model predicts more consistently during circuit activities than during the baseline and recovery phases.

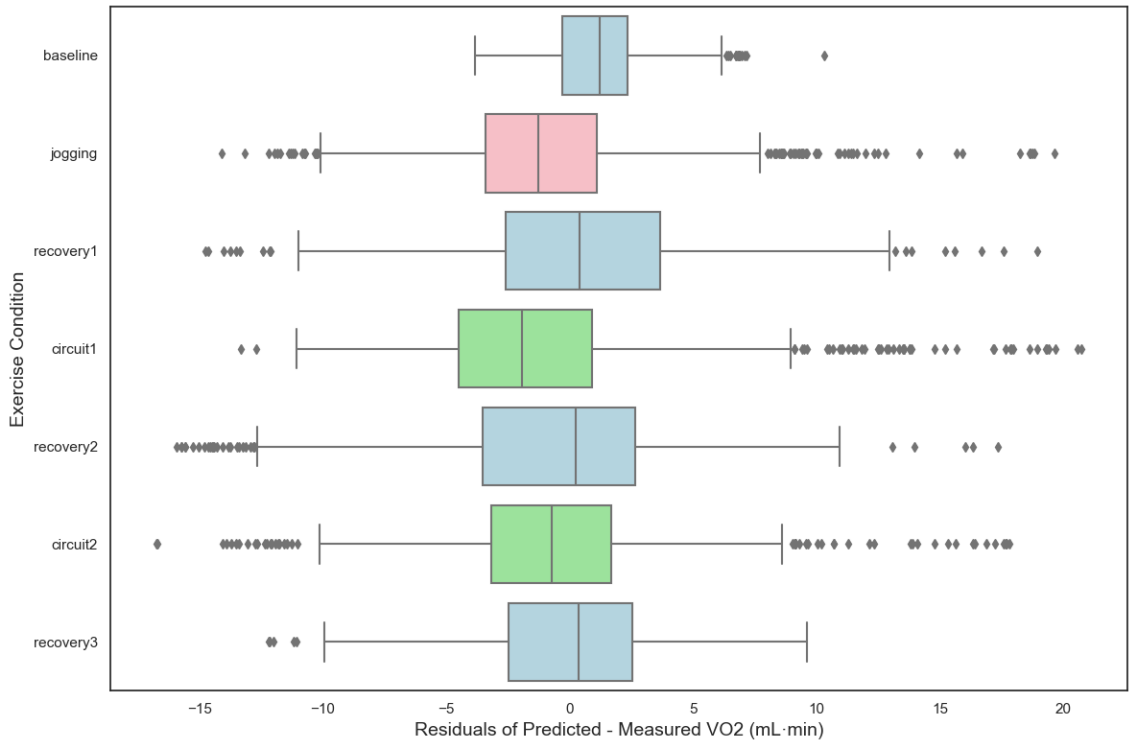
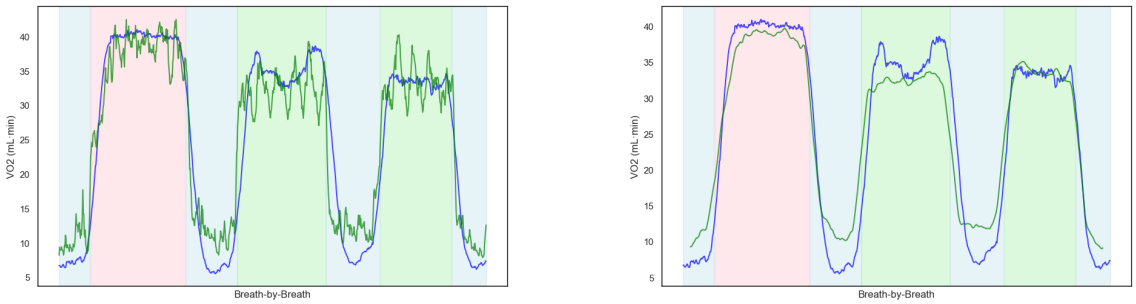


Figure 4.6: Box plots of the residuals (predicted VO_2 minus measured VO_2) across different exercise conditions for the LSTM model using RAW data representation and dataset C. The exercise conditions include baseline, jogging, recovery1, circuit1, recovery2, circuit2, and recovery3.

Figure 4.7 displays a comparison of breath-by-breath measured VO_2 values (blue line) versus predicted VO_2 values (green line) obtained from the LSTM model using RAW representation and dataset C. While the predicted VO_2 values generally mirror the overall trend of the measured data, notable deviations are observed during exercise and recovery phases. The model effectively captures the general pattern of VO_2 fluctuations but exhibits challenges with precise tracking. During high-intensity exercise periods, significant discrepancies between the predicted and measured VO_2 values become apparent. The predicted values often overshoot or undershoot the peaks of the measured data, indicating difficulty in accurately estimating VO_2 levels

during increased physical activity. In the recovery phases, the model's predictions do not closely follow the rapid decreases in VO_2 observed, sometimes lagging behind or leading ahead of the actual changes. These inconsistencies suggest that the model struggles to accurately capture the oxygen kinetics during recovery periods. Finally, the model demonstrates limitations in tracking rapid changes in VO_2, particularly during transitions between exercise and rest.



(a) Unsmoothed predictions

(b) Smoothed predictions

Figure 4.7: Comparison of measured VO_2 values (blue line) and breath-by-breath VO_2 predictions (green line) for Subject 2 using the LSTM model with RAW representation and dataset C. The left plot shows unsmoothed predictions with a MAE of 3.374 $mL \cdot kg^{-1}$, while the right plot displays smoothed predictions with an MAE of 2.902 $mL \cdot kg^{-1}$. The plots include different exercise and recovery phases, shaded as follows: baseline and recovery phases (light blue), jogging (pink), and simulated soccer circuit (light green).

Our pilot study is the first to explore VO_2 estimation specifically during simulated team-sport exercise, making a direct comparison to existing treadmill or cycling research less straightforward; however, within the broader context of studies [66, 68, 70–72], our results achieved are similar. We do encounter challenges in transitions between different intensities, partly because our training dataset did not fully capture the rapid changes observed in the test phase. Even so, the performance achieved with raw IMU data and LSTM architectures highlights the feasibility of wearable-based VO_2 monitoring in team-sports scenarios. Our findings serve as an important starting point in future work on athlete monitoring.

Summary. In this second study, we evaluated a range of neural network models to predict individual oxygen uptake during outdoor running and simulated team sports activities, using four sensor configurations and two data representations. While the

MLP with MAD representation and torso sensor data achieves the lowest validation error, it does not generalize well to the test set. Conversely, the BiLSTM model using raw data from torso and leg sensors demonstrated better test performance. However, the models exhibit limitations in precisely tracking rapid VO_2 changes during high-intensity exercise and recovery phases, highlighting challenges in capturing oxygen kinetics during transitions between activity and rest.

4.3 Conclusions

In this chapter, we examined the capabilities of neural networks in two distinct tasks across different domains: exosome classification and oxygen uptake estimation. In the first project, we developed a methodolgy for classifying healthy, hyperglycemic, and hypoglycemic conditions using SERS spectra from exosomes. The approach involves smoothing the raw input spectrum and subtracting the estimated background signal, leaving the exosome signal, which is then fed into a MLP model for classification. The method yields promising results, achieving an overall accuracy of 66.7%. The primary challenge encountered by the classifier is the strong correlation between hyperglycemic and normal samples. Additionally, the model tends to misclassify hypoglycemic samples in favor of hyperglycemic ones. Despite these challenges, the result suggests that this method is effective in capturing and analyzing exosomes, demonstrating its potential for distinguishing subtle differences in complex and heterogeneous Raman signatures from various cell models.

In the second project, we investigated the use of a range of neural network architectures to estimate individual oxygen uptake during simulated team sports activities using wearable sensor data. As part of this study, we investigated different sensor configurations, including torso placement with additional sensors placed on the arm, leg, or both. While results showed no substantial advantage of deep learning models over the baseline MLP model in terms of predictive performance, it was possible to uncover some interesting results. The best-performing MLP model achieves a MAE of 3.79 $\mathrm{mL} \cdot \mathrm{kg}^{-1} \cdot \mathrm{min}^{-1}$, slightly higher than the LSTM model,

which achieves an MAE of 3.69. While deep learning models such as LSTM and CNN demonstrate strong performance with raw sensor data, MLP models remain competitive, particularly when using MAD data representations. The choice of sensor placement configuration also had a significant impact on performance, with multi-sensor setups, such as torso and leg (Dataset C) or torso and arm (Dataset B), yielding the most accurate predictions.

Through these two tasks, the different forms of neural network demonstrated robust performance across different domains, solving different problems. However, these methods still exhibit certain limitations. First, the model architectures do not fully exploit the information contained within the inputs. In the exosome classification task, the entire exosome signal is input into the machine learning models with minimal pre-processing. This could be one of the reasons why the predictive performance does not reach practical levels. In the oxygen uptake prediction task, two representations are provided to the machine learning models: raw sensor signals and a mildly transformed version. Although the transformation does not improve performance on the test sets, it shows marginally lower errors on the validation sets, which are technically still unseen data. Moreover, while LSTM and 1D-CNN models can leverage the temporal nature of the data, they may not fully capture the complex relational structures inherent in the data. The good performance of the LSTM in the second task further underlines how specialized architectures can be effective in capturing complex dependencies, suggesting that future research may benefit from more specialized models such as GNNs that explicitly leverage data structure. Therefore, in Chapter 5, we will adopt a graph-based approach, where exosome or sensor data are represented as networks whose structure encodes potential interactions or dependencies. We employ graph neural networks because they are specifically designed to learn from graph-structured data, thereby providing a more expressive way to model inter-dependencies and potentially improve performance.

Secondly, neural networks experience significant overfitting. As model complexity increases, the risk of overfitting grows, with only marginal improvements observed

over simpler models. In the oxygen uptake prediction task, the performance of MLPs and more complex architectures is comparable across different data representations, with both approaches ranking among the top-performing models. A plausible explanation is that deeper models may overfit not only the training data but also the validation data. While we used L2 regularization, cross-validation, and early stopping to mitigate the risk of overfitting, this problem and challenge persists. A popular approach to address overfitting is to introduce a regularization term into the objective function, guiding the training process to favor model properties that are more likely to generalize well. This regularization term can often be informed by domain-specific knowledge of the underlying process. In Chapters 6 and 7, we adopt the approach of physics-informed neural networks, which incorporate a regularization term into the objective function to ensure that the trained neural networks adhere to physical laws. This hybrid approach leverages domain knowledge, in the form of differential equations describing the system's dynamics, while maintaining the flexibility of neural networks to learn from the provided data.

Chapter 5

Graph Neural Networks

In Chapter 4, we evaluated the capabilities of neural networks on two separate real world machine learning tasks. While neural networks are regarded as performing strongly across a wider range of applications, achieving robust performance requires the incorporation of more extensive domain knowledge. This may require more careful data analysis and the use of more complex architectures to effectively capture intricate patterns. One approach to making neural networks more powerful and potentially more interpretable is to adopt a graph-based structure for the neural network. The concept here is to merge graph-based modeling with a neural network machine learning function to improve its performance. To develop this approach, we first conduct a sizeable research project using graph analytics and then introduce some of our learnings to develop a graph-based neural network. To continue to align with our research goals, we conduct these research problems on real world problems and datasets. In Section 5.1, we provide a general introduction to graph models and then in Section 5.2, use a series of graph analytics to measure, evaluate and make predictions using a shared bicycle network. We then present our work on building a graph-based neural network in Section 5.3 and evaluate this novel function in air quality prediction.

5.1 Graph Modeling

Modeling data as networks has received considerable attention in the last two decades [212, 213]. A network (or graph) consists of a set of nodes (vertices) and a set of edges (links) connecting pairs of nodes. Nodes typically represent entities of interest, while edges denote specific relationships or interactions between these entities. Analyzing networks constructed from data provides several advantages over traditional tabular representations, such as enhanced data visualization, the ability to identify influential components or critical relationships through network metrics, and the detection of patterns, connectivity, and modular structures through community detection algorithms. Additionally, constructing complex networks facilitates the application of an advanced variant of neural networks known as graph neural networks (GNNs). By leveraging a graph representation, GNNs can effectively regulate the flow of information, enabling each node to aggregate information from itself and its immediate neighbors. This allows for the discovery of potential relational patterns embedded within the graph structure.

In this chapter, we take the approach of complex network analysis and graph neural networks in two separate projects. In the first one, we introduce a network-based methodology for the spatial-temporal analysis of transportation data. The proposed method involves several steps, each providing a distinct perspective on the data. First, exploratory statistical analysis is conducted to summarize and gain a comprehensive understanding of the dataset. Next, transportation networks are constructed to describe how individuals move through the transportation system. The dynamics of these flow networks are then examined by constructing a series of flow networks over different time intervals to capture changes in movement patterns over time. Finally, the spatial-temporal behaviors of stations are investigated by building correlation networks that represent temporal similarities in activity patterns.

In the second project, we propose a novel graph neural network designed to enhance spatio-temporal feature extraction for air quality forecasting. Our methodology integrates an attention mechanism that adaptively assigns weights to different

factors, thereby improving prediction accuracy. By applying attention layers to both temporal and spatial dimensions, the model is able to simultaneously learn critical spatio-temporal features. The proposed model is validated using air quality data collected from 10 monitoring stations in Hanoi, Vietnam, over a period of more than one year. The results show improvements compared to baseline methods as well as the interpretability of the model.

The implementations for these two projects can be found at:

- Bike-sharing system network analysis: `https://github.com/dinhvietcuong1996/networks-bike-sharing-system`;
- Spatio-temporal attention-based air quality forecasting `https://github.com/dinhvietcuong1996/SpaTemAtt-Air`

5.2 Graph Analytics using a Travel Network

In this section, we introduce a dataset which captures a network of bike trips over a 2 year period and a series of analyses that exploit a graph model to gain new insights. Bike sharing systems have been gaining widespread popularity globally, offering significant convenience to travelers. These systems allow users to pick up a bike, travel to their destination, and return the bike at any convenient location. This approach presents a low-cost and efficient mode of urban transportation. Additionally, bike sharing addresses the challenge of first-and-last mile connectivity and alleviates traffic congestion in cities, thereby contributing to increased efficiency of the overall transportation system. Furthermore, cycling is a healthy, environmentally friendly alternative for urban mobility.

5.2.1 Problem

Recently, bike sharing providers have introduced dockless bike sharing services, allowing users to pick up and drop off bikes at more informal locations, often referred to as *virtual stations*. Given the flexibility of virtual stations, providers

are incentivized to monitor bike usage and determine the optimal configuration of these stations. In essence, the challenge lies in assessing "how closely the currently deployed network of virtual stations approximates the configuration that optimizes bike usage". Bike sharing data is typically made available in tabular form, such data is often limited in its ability to model and analyze this type of problem. Moreover, data related to station locations and trips must be analyzed at varying levels of granularity to provide both a comprehensive view of network activity as well as detailed insights where needed. Any proposed solution must take into account that decisions made at a global level may have adverse effects on individual stations, while local adjustments to the network topology could impact the global system negatively. Specifically, a series of research questions can be articulated as follows:

- **Research Question 1.** How can a detailed overview of the most and least active stations and routes be created?
- **Research Question 2.** Is it possible to drill-down using the time dimension to better understand these levels of activity?
- **Research Question 3.** Spatial similarities can be identified as part of RQ1 and temporal similarities can be observed as part of RQ2. Is it possible to combine these dimensions to identify stations that exhibit similar characteristics over specific time intervals?
- **Research Question 4.** A more advanced global analysis involves understanding the relationship between each station and all other stations in terms of activity. Can a station's *pattern* be defined by its activity over time so that two stations are related if their patterns are similar?

5.2.2 Methodology

Preliminaries

In general, there is no single network structure for a particular dataset and researchers generally measure or compare different graph structures according to their needs.

Based on the types of analyses planed, we developed different structures to meet different requirements.

Spatial Graph Network. The Spatial Graph Network (SGN) is the most fundamental form of network. In this network, nodes are linked by undirected edges, where an edge between two nodes is established if at least one trip has occurred between the origin node (source) and the destination node (target). The SGN offers a comprehensive, aggregated perspective on the network's topology and the volume of activities within it. Each edge is further associated by a weight, which quantifies the total number of trips that have taken place along the route connecting the two nodes over the entire period of interest. More formally, the SGN is defined in Def. 5.1.

Definition 5.1 (Spatial Graph Network) *The SGN is a pair* $SGN = \langle S, J \rangle$*, where* S *is the set of nodes (vertices), and* J *denotes the set of edges connecting these nodes. A node (or vertex)* $s \in S$ *is described as a pair* $s = \langle lat, lon \rangle$*, where* lat *and* lon *correspond to the latitude and longitude, respectively, specifying the spatial location of the node. An edge* $j \in J$ *is defined as a triple* $j = \langle o, d, a \rangle$*, where* $o, d \in S$ *are the origin and destination nodes of the edge, and* a *represents the activity between these two vertices.*

Temporal Graph Network. The aggregated nature of the SGN can hide important details and the dynamic evolution of networks over time. To address this limitation and allow for the study of network evolution, it is essential to explicitly incorporate time as a dimension within the graph. Segmenting data, especially graph or unstructured data, enables more granular analysis and improves efficiency in terms of query response times [214].

For the above reason, we introduce the temporal graph network (TGN). A TGN is defined as an ordered sequence of graphs, where each graph represents the state of the network at a specific moment in time. In other words, each graph in the TGN provides a snapshot of the SGN during a time window, over the course of the period of interest. The TGN facilitates a more detailed analysis of the network by allowing for comparisons between different projections of the graph, thereby supporting the

examination of the network's evolution over time. Formally, a TGN is defined in Definition 5.2.

Definition 5.2 (Temporal Graph Network) *A TGN is a set* $TGN = \{\mathcal{SGN}_1, \mathcal{SGN}_2, \ldots, \mathcal{SGN}_T\}$*, where* SGN_i *represents a time-bounded SGN, capturing the state of the network during a specific time interval. Each* SGN_i *is a pair* $SGN_i = \langle S, K \rangle$*, where a node* $s \in S$ *is described by the pair* $s = \langle lat, lon \rangle$*, corresponding to its spatial coordinates. An edge* $k \in K$ *is defined as the tuple* $k = \langle o, d, t, \delta, a \rangle$*, representing the volume of activity between two nodes during a specific time interval. Here,* $o, d \in S$ *are the origin and destination nodes,* t *is the starting time point of the observation period,* δ *is the duration, and* a *represents the volume of activity between nodes* o *and* d *during the observation period from* t *over the interval* δ*.*

Spatio-Temporal Graph Network. The Spatio-Temporal Graph Network (STGN) has fundamentally different characteristics compared to previous graph representations. In the STGN, the edges model the *similarity* between two nodes (vertices) based on the temporal *patterns* in their activities. This allows the data to be analyzed as timeseries patterns [215], facilitating the identification of trends, seasonal or cyclical components, irregularities, and potentially, the diversity within the data [216]. The STGN is defined formally in Def. 5.3.

Definition 5.3 (Spatio-Temporal Graph Network) *A STGN is a tuple* $STGN = \langle S, H, T_s, T_e \rangle$*, where* S *represents the set of nodes (vertices), and* H *denotes the set of edges connecting these nodes.* T_s *marks the starting point of the network observation, while* T_e *represents the time at which the observation ends. Each node (vertex)* $s \in S$ *is a pair* $s = \langle lat, lon \rangle$*, where* lat *and* lon *represent the latitude and longitude of the spatial location of the node. An edge* $h \in H$ *is a triple* $h = \langle s_1, s_2, r \rangle$*, where* s_1 *and* $s_2 \in S$ *are two nodes, and* r *denotes the similarity value between them. This similarity is calculated based on the temporal patterns of their activities observed over the series of time periods.*

Network Constructions

Bike-sharing systems, as time-evolving networks, can be characterized by three core features: space, time, and activity volume. Space and time can be considered dimensions, whereas activity is a measure whose value depends on how one looks at the dimensions. In this context, we model the bike-sharing system as a graph network that analyzes activity (trips) with respect to one or more dimensions. Specifically, we construct three types of networks, based on the generic network types defined in Section 5.2.2. The first network type focuses solely on the spatial dimension. The second network, while still incorporating spatial properties, focuses on the temporal dimension for the purpose of analysis. The third network integrates both spatial and temporal dimensions. Each network type offers greater analytical power and complexity than the preceding one. However, simpler networks are easier to construct, and as the cost of building more complex networks increases, it is important to understand the strengths and limitations of each network type. In the following sections, we will discuss the construction of these networks, the analyses each allows, and their applications.

Spatial Bike Graph Network. We construct a Spatial Bike Graph Network (SBiGN) based on the SGN framework to capture the spatial characteristics of bike-sharing usage, particularly station connections and the volume of trips between stations. This representation is invaluable for analyzing the *traffic flow* within the transportation system. In a straightforward sense, the nodes of the SBiGN represent bike stations, while the edges correspond to trips (journeys) made between these stations. Each edge is associated with a weight, reflecting the total number of trips occurring along the route between the two connected stations over the entire period of interest. A formal definition of the SBiGN is provided in Def. 5.4.

Definition 5.4 (Spatial Bike Graph Network) *The SBiGN is a pair* $SGN = \langle S, J \rangle$*, following the structure outlined in Def. 5.1, where each node corresponds to a bike station and each edge represents journeys (trips) between stations. Additionally, the activity a between two stations can be interpreted as the cardinality, or the total*

number, of trips made along that route.

We refine the SBiGN by removing statistically insignificant data, which introduce noise and reduce the effectiveness of the analysis. First, we identify and remove *weak* edges, defined as edges with relatively low weight values compared to the overall network aggregation. This is a common step in network optimization [217], where large numbers of insignificant data points can slow down the analysis without providing substantial new insights. The threshold for edge removal is chosen to ensure the network remains strongly connected, thereby preserving the well-defined nature of network algorithms such as closeness and betweenness, while also reducing the network's size. Hence, this optimization improves both the quality and efficiency of the analysis. A binary search version of the threshold algorithm is outlined in Algorithm 5.1. The network's strongly connected property is continually verified to ensure that the network's connectivity remains with the threshold. The `is_strongly_connected` function ensures that all nodes in the network are reachable from any given starting node, as described in [218].

Algorithm 5.1 find_threshold(edges): Binary Search to find the largest *threshold* for strong connectivity from edge.

Require: $edges \leftarrow$ A list of all edges in the network, each with a weight
Ensure: *threshold*: Largest threshold value for strong connectivity, $NULL$ if there is none.

$edgeValues \leftarrow$ unique weights extracted from edges sorted in ascending order
$low \leftarrow 0$
$high \leftarrow \text{length}(edgeValues) - 1$
$threshold \leftarrow NULL$
while $low \leq high$ **do**
    $mid \leftarrow \left\lfloor \frac{high+low}{2} \right\rfloor$
    $network \leftarrow$ build_network$(edges, edgeValues[mid])$
                                ▷ Removes edges with weight $< edgeValues[mid]$
    if is_strongly_connected$(network)$ **then**
        $threshold \leftarrow edgeValues[mid]$
        $low \leftarrow mid + 1$
    else
        $high \leftarrow mid - 1$
    end if
end while
return *threshold*

Secondly, we remove *looping* edges, which are defined as edges where the source and destination nodes are the same (i.e., bike trips that originate and terminate at the same station). These edges contribute little to the overall analysis and may introduce complications for certain graph algorithms. However, loops could be a meaningful aspect of the transportation system's behavior, and thus, they should be examined within a separate network.

For analytical purposes, one key advantage of the spatial graph is its ability to be overlaid onto a geographical map of the bike-sharing network's deployment area. In line with **Requirement 1**, Figure 5.1 illustrates the SBiGN using data collected over a 15-month period from June 2020 to August 2021, enabling the clear identification of the busiest stations and the spatial distribution of high-activity stations. In this visualization, node size is proportional to the total number of trips either originating from or ending at the station. The ten most active stations are highlighted in red, while other stations are shown in blue. Only the top 10% of edges (most frequently used routes) are displayed in red and the remaining routes in blue.

Temporal Bike Graph Network. From a practical standpoint, we construct a Temporal Bike Graph Network (TBiGN) by generating a sequence of time-bounded SBiGNs, as defined in Def. 5.5. The TBiGN follows a specified temporal order and a predetermined time scale, with each SBiGN_i capturing network activity within a specific time interval.

These intervals can have any duration and may or may not overlap. This flexibility is essential for modeling how temporally adjacent networks reflect the system's evolution over time. The choice of intervals influences the degree of similarity between successive networks. Intuitively, shorter intervals result in networks that are less similar but allow for a more granular representation of system dynamics, aligning with **Requirement 2** outlined in the introduction. In contrast, longer intervals may produce more similar networks due to the aggregation of data over extended periods. However, this comes at the expense of overlooking certain aspects of system evolution. Longer intervals are nonetheless necessary for capturing broader temporal

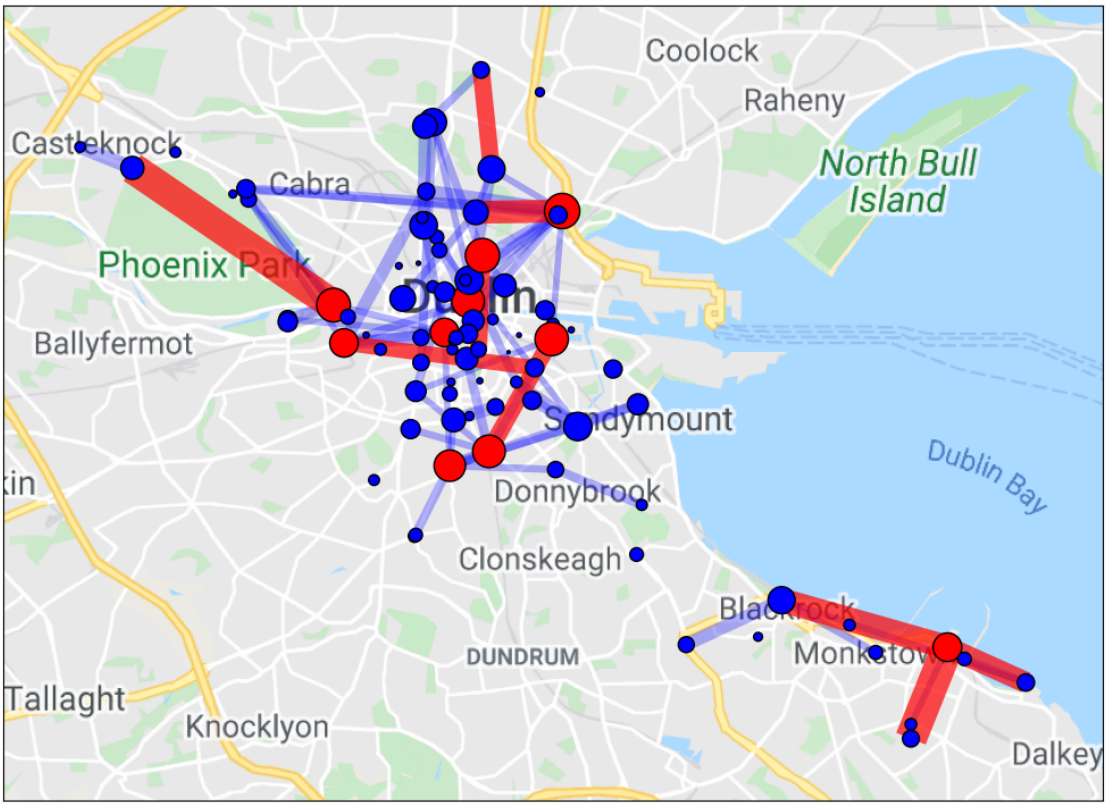


Figure 5.1: **Geographic Map overlaid with SBiGN**. Each circle represents a bike station and is sized according to its trip volume, with the 10 busiest stations (by trip volume) shown in red and all others in blue. Lines represent routes between stations and are also sized by trip volume; the 10 most frequently used routes are highlighted in red, and the remaining routes are shown in blue.

properties such as seasonality and stationarity [216].

Definition 5.5 (Temporal Bike Graph Network) *A TBiGN is a set* $TBiGN = \{SBiGN_1, SBiGN_2, \ldots, SBiGN_T\}$*, where* $SBiGN_i$ *is a time-bounded SBiGN and is described as a pair* $SGN_i = \langle S, K \rangle$*. This corresponds to Def. 5.2, where each node and edge in* SGN_i *corresponds to a bike station and a route, respectively, within the associated* $SBiGN_i$*.*

For bike-sharing graph networks, time intervals are typically defined over hours, days, or weeks to facilitate the analysis of periodic and seasonal patterns. In addressing **Requirement 3**, these networks are subsequently clustered into groups, with the centroid of each cluster representing the entire group for analysis. This approach simplifies the analysis of a large set of graphs, allowing the focus to be placed on a smaller subset of centroid graphs.

Spatio-Temporal Bike Graph Network. The Spatio-Temporal Bike Graph Network (STBiGN) is designed to use STGNs as a fundamental building block and is defined in Def. 5.6. This type of network is used to address **Requirement 4** where a different perspective on station-by-station correlation can be explored.

The construction of STBiGN involves three steps:

1. In the first step, a time series is generated for each station based on a specified timescale. Typically, the analysis is performed over contiguous, non-overlapping intervals such as hourly, daily, weekly, monthly, or yearly periods. This allows for the exploration of periodic patterns in the transportation system, such as morning and evening rush hours. Once the timescale is known, each station is represented by a time series reflecting the number of trips within each time-step window, enabling the study of station activity over the defined timescale.

2. In the second step, a similarity score is computed for *every* pair of stations using the Pearson correlation coefficient, as defined in Equation 5.1.

$$S(x,y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})}}. \tag{5.1}$$

   Here, (x_i) and (y_i) are two time series of length n is the number of time steps in which activity is measured. The mean values $\overline{x}$ and $\overline{y}$ are the averages of x and y, respectively. The Pearson correlation coefficient, which measures the linear relationship between the two variables, is in the range $-1 \leq r \leq 1$. As most network algorithms operate on non-negative edge weights, the coefficient is normalized to a range of 0 to 1 using the transformation $S(x,y) = (S(x,y) + 1)/2$.

3. In the final step, STBiGN is constructed based on the correlation matrix of the Pearson coefficients. The matrix of all normalized similarity scores forms the adjacency matrix of the network. In this network, each node represents a station, and an edge *always* exists between every pair of stations, with the edge weight corresponding to the computed similarity score. To improve the

usability of network metrics such as closeness, betweenness, and local clustering coefficients—which perform poorly on fully connected or weighted networks, statistically insignificant (weak) edges are removed. During this edge-trimming step, the strongly connected property of the network is maintained.

Definition 5.6 (Spatio-Temporal Bike Graph Network) *A STBiGN is a tuple* $STGN = \langle S, H, T_s, T_e \rangle$*, in accordance with Def. 5.3, where a node and an edge in* $STGN_i$ *refer to a bike station and a trip in* $STBiGN_i$*, respectively.*

Graph Metrics

The primary techniques for analyzing the structure of both SGN/TGN and SBiGN/TBiGN are centered around centrality metrics. These metrics provide quantitative measures of a node's importance within the network [219, 220]. When analyzing centrality metrics, particular attention should be given to nodes with exceptionally high scores, as these nodes may either represent outliers or play crucial roles in the networks.

Graph Metrics for SBiGN and TBiGN

- **Strength**. Node strength is defined as the sum of the weights of the edges connecting a node to its immediate neighbors [221]. In the context of SBiGN/TBiGN, the edge weight represents the volume of activity (i.e., the number of trips). Therefore, the strength of a station is the total number of trips occurring to or from that station.

- **Degree.** A node's degree is the number of edges connecting it to adjacent nodes, which is the number of nodes the node is directly connected to [222]. In the SBiGN/TBiGN, the degree of a station represents the number of other stations with which it shares at least one trip.

- **Closeness.** Closeness centrality measures how close, on average, a node is to all other nodes in the network [223, 224]. The distance between nodes is defined as the length of the shortest path, or the minimum number of hops (routes) between them. In SBiGN/TBiGN, a station's closeness score quantifies

how centrally located it is in relation to all other stations, reflecting its role in overall movement across the network.

- **Betweenness.** Betweenness centrality indicates how often a node lies on the shortest paths between other node pairs [224, 225]. A high betweenness score suggests that the station plays a key role in network connectivity. In SBiGN/TBiGN, stations with high betweenness scores often serve as "bridges" or critical waypoints that connect different parts of the network.

- **Local Clustering Coefficient.** The local clustering coefficient measures the likelihood that a node's neighbors are also connected to one another [226]. In the context of SBiGN/TBiGN, a high local clustering coefficient suggests that travelers departing from a station are likely to move between only a few closely interconnected stations.

- **Communities.** Communities refer to groups of nodes that are more densely connected with each other than with nodes outside the group [158, 227]. In the SBiGN/TBiGN, a community represents a cluster of stations where a higher proportion of trips occur within the community than between stations outside of it.

Graph Metrics for STBiGN. The same graph metrics are employed as in the previous graphs. But due to the difference in nature of STBiGN, the interpretation of these metrics in such correlation-based networks [228–230] differs based on the network's construction and definitions:

- **Strength.** Node strength, defined as the sum of scaled correlations connected to the node, reflects the *degree* of similarity between the station and other stations in terms of their activity patterns.

- **Degree.** In a STGN, the degree is the number of stations that exhibit similar behavior to the station under observation.

- **Closeness.** In the context of a STGN, closeness is defined as the minimum number of significant similarities that form a path between two stations. The greater the distance between two stations, the less similar their temporal activity patterns.

- **Betweenness.** In this case, betweenness centrality measures a station's role in connecting groups of similar stations. Stations with high betweenness scores typically show similarity to multiple distinct groups.

- **Local clustering coefficient.** A high local clustering coefficient in a STGN indicates that a station's neighboring stations are also likely to be similar to one another. Stations with high coefficients form groups with more or less the same activity patterns.

- **Communities.** Hence connections are built based on the similarity of stations' activity time series Members of the same community share common temporal patterns in their activity.

5.2.3 Experiments

The dataset used for the experiments is described in Section 2.2.1.

Materials and Tools

The programming environment uses Python, with several key libraries for experiments, including Scipy [231], Matplotlib [232], and NetworkX [233]. Pandas [234] is utilized for data processing, analysis, and graph construction while Matplotlib is used for data visualization. Graph visualizations are produced using Google Maps API [235] in combination with NetworkX. The Neo4j graph database [236] is used to manage and store all graphs. Additionally Neo4j's Graph Data Science Library [237], with built-in centrality metrics and community detection algorithms, is used for all network metrics.

Network Constructions

SBiGN. As described in Def. 5.4, stations are modeled as nodes in the SBiGN, and are linked by undirected, weighted edges if at least one trip occurred between them in the 15-month period. The weight of each edge is the total number of trips, counting both directions. To enhance the analysis, loop edges and weak links are removed from the SBiGN.

To identify and remove weak connections, we progressively lower the edge weight threshold until the network achieves strong connectivity. In the experiment, the final threshold is set at 11 trips by Algorithm 5.1. The resulting network density, calculated as the ratio of actual edges to potential edges, is approximately 19%. Following the removal of 9,544 loop trips and 7,920 trips associated with weak links, the final SBiGN consists of 18,717 trips connected by 686 edges, with the number of nodes unchanged at 86.

TBiGNs. TBiGNs are used to analyze movement patterns with specific time intervals. In this experiment, daily and monthly graph projections are constructed to enable detailed and broader analyses, respectively.

For daily analysis, 7 TBiGNs are generated, representing each day of the week, using data spanning 66 weeks within the 15-month period from June 2020 to August 2021. These 7 TBiGNs are clustered into two groups: "weekday," comprising 5 TBiGNs for weekdays, and "weekend," comprising 2 TBiGNs for the weekend days. Similar to the SBiGN, loop trips and weak links were removed from the TBiGNs. In this case, weak links are defined as edges with fewer than 5 trips. On average, approximately 1,364 loop trips are removed per day, accounting for 26.4% of the data, along with 551 trips of weak edges, representing 10.6%.

For broader monthly analysis, monthly TBiGNs are used to examine variations in network activity across a monthly timeframe. A rolling window of *4 weeks* size, starting at week 1, slides one week at a time. This results in 63 networks covering the 66-week period June 2020 to August 2021. Nodes represent stations, and edges are weighted by the number of trips occurring within each 4-week window. In addition

to removing loop trips, the network is further optimized by eliminating edges with fewer than 3 trips, using a weekly threshold of 0.75 trips per week. On average, around 384 trips are removed from each TBiGN, totaling of 1,628 trips (23.6%).

STBiGN. Spatio-Temporal Graphs enable the analysis of temporal patterns occurring between spatial locations. We build three levels of granularity for analyzing the similarity between stations across temporal dimensions: hourly, where the 24 hourly intervals are averaged across the entire dataset; daily, where each day of the week is averaged, and monthly, where a single monthly value is calculated for each station over the 15 months of the study.

A typical investigation for this type of network examines the similarity of the growth of stations over time. Nodes are stations and an edge connects two stations based on the degree of similarity in their trips volumes over the analyzed time period. The edge weight is the Pearson correlation coefficient between two stations' timeseries. Similarity is considered significant only when the correlation coefficient exceeds the threshold value of $T = 0.533$. This threshold ensures strong network connectivity while minimizing the number of edges. The network's density, which indicates how well connected the graph is, is 0.098. This is calculated as $\#edges/\#possibleEdges$. In STBiGNs, the higher the value the greater the similarity between any pair of stations. In contrast, lower density values can indicate stations with more distinctive travel patterns.

5.2.4 Analysis and Discussion

In this section, we validate our research using the networks constructed in Section 5.2.3 against the set of requirements outlined in Section 5.2.1. There are 6 parts that reflect the different types of analyses and different levels of granularity. However, we only present in details two parts, the monthly temporal networks and daily spatio-temporal networks. Other discussions, including spatial bike graph, daily temporal graph, monthly spatio-temporal graph and hourly spatio-temporal graph, are similar in terms of techniques and are presented in the Appendix B.

4.2.2.1 Spatial Bike Graph Networks (SBiGN)

Basic graph analytics are applied to analyze bike movement across the entire network. Table 5.1 lists stations in descending order according to their **strength** values. Notably, the station at *Fairview Avenue Lower* has the highest strength at 1,478, closely followed by centrally located stations such as *Mountjoy Square South* (1,385), *Criminal Courts of Justice* (1,350), *Grand Canal Docks* (1,322) and *O'Connell Street* (1,272). The decline in station strength follows a slightly negative-exponential trend, near linearity. Small stations tend to have very few trips over the observation period, often fewer than several dozen. Interestingly, many of these small stations are situated near larger, higher-traffic stations.

Table 5.1: Node Strength in SBiGN

Station	**Strength**
Fairview Avenue Lower	1,478
Mountjoy Square South northside opposite No. 40-45	1,385
Outside Criminal Courts of Justice	1,350
Grand Canal Docks, outside Fresh	1,322
Beside Penneys O'Connell Street	1,272
...	...
Benson Street	43
Start of St. Stephen's Green terrace.	43
Pearse Street, outside Subway	26
Phibsborough Rd, outside Broadstone Hall Studen...	23
Merrion Square North opposite Oscar Wilde House	11

Table 5.2 presents the edge weights. The most frequently traveled routes are predominantly associated with commuting trips or follow the coastal line between Blackrock and Monkstown. The edge weights exhibit an exponential decrease, with the most popular routes having nearly twice the weight of the third most popular route. Unsurprisingly, the least traveled routes have negligible weights. While some stations rank highly in both node strength and edge weight metrics, there are notable differences in the patterns of route and station popularity. Station strength displays a more gradual decline, whereas route popularity decreases more sharply.

Figures 5.2 to 5.4 highlight key geographical areas, with the size of a station's node representing its relative importance. In Figure 5.2, **degree** of a station is closely

Table 5.2: Edge Weight in SBiGN

Source	**Target**	Weight
Dun Laoghaire Dart	Honeypark Neptune Way	247
Criminal Courts of Justice	Phoenix Park Gate	243
Blackrock Main St.	Dun Laoghaire Dart	190
Drumcondra Road Upper	Fairview Avenue Lower	179
Dun Laoghaire Dart	Sandycove Beach	165
...	...	...
End of Adelaide Road	Talbot Hotel Stillorgan	11
Richmond Row	Dun Laoghaire Dart	11
Sandymout Village	Dun Laoghaire Dart	11
Grand Canal Docks	Seapoint	11
Ballsbridge	Monkstown	11

correlated with its **strength**, except for the *Monkstown* station, which High-degree nodes, depicted in red, are typically located in residential areas, acting as hubs for people traveling to and from these regions.

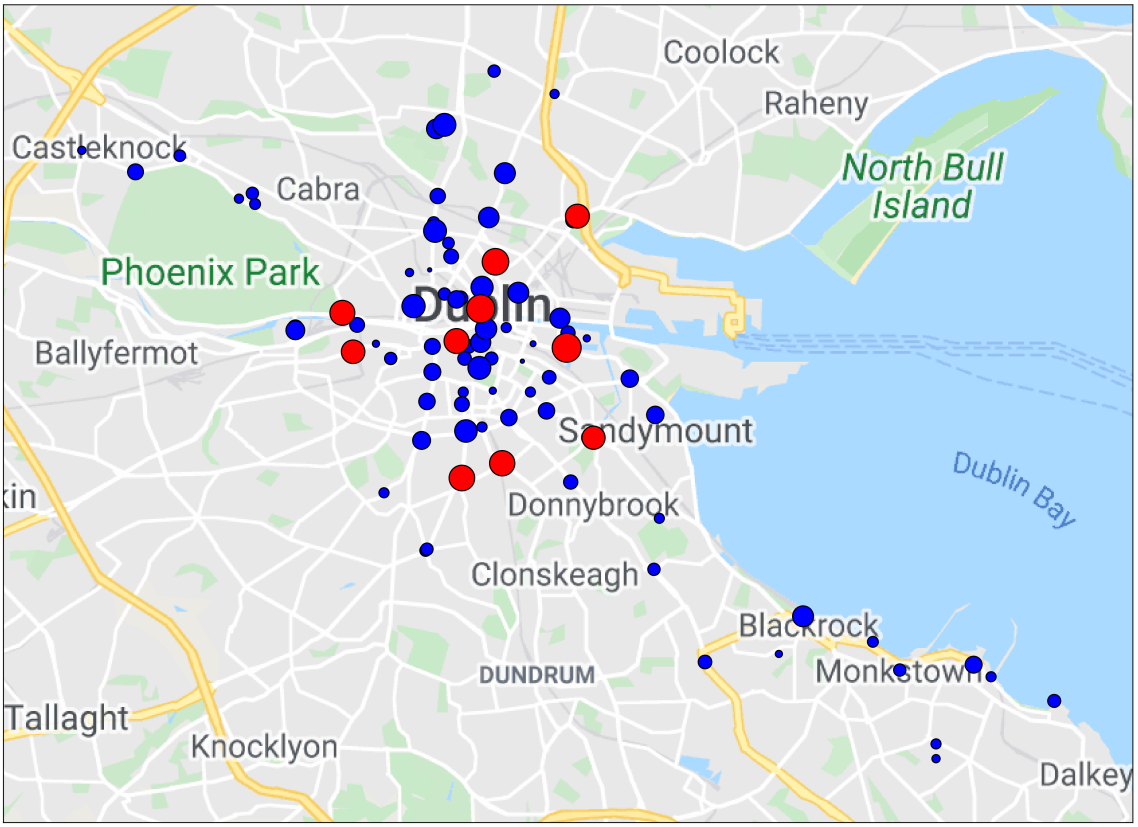


Figure 5.2: **Graph Analytics for Spatial Bike Graph Networks: Degree**. Each circle represents a bike station and is sized by its degree, with the 10 highest-degree stations shown in red and all others in blue.

The **closeness** metric offers further insights into the significance of stations

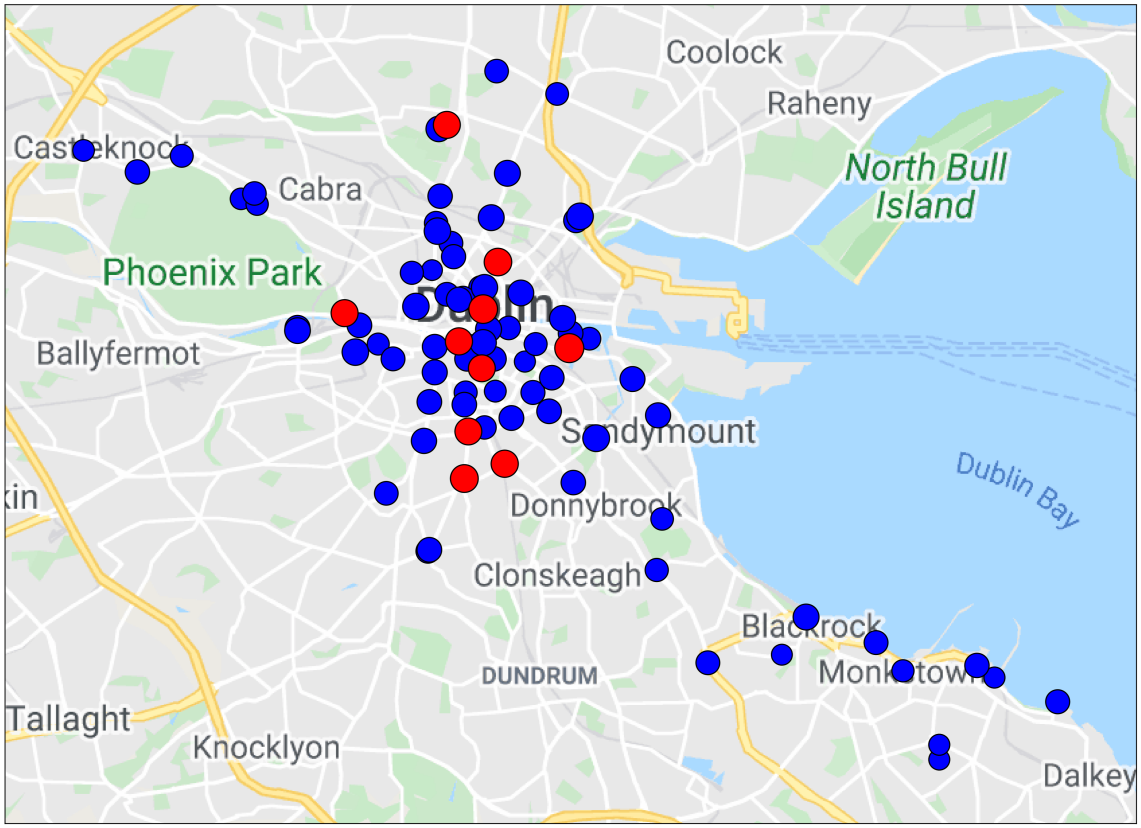


Figure 5.3: **Graph Analytics for Spatial Bike Graph Networks: Closeness**. Each circle represents a bike station and is sized by its closeness centrality, with the 10 highest-closeness stations shown in red and all others in blue.

within the network. Stations with high closeness scores (depicted in red in Figure 5.3) are well-connected to numerous other stations, whereas stations with lower scores tend to be more isolated. The top 10 stations in terms of closeness are located in the city center, underscoring their extensive connectivity with stations across Dublin. The network's average closeness score is 0.59, indicating that, on average, a station is approximately 1.7 trips away from other stations. As closeness values exhibit minimal variation across the stations, this suggests that there is no clear standout station that serves as a central hub for reaching many other stations via existing trips. As a result, the removal or closure of any single station would not significantly disrupt movement across the entire network.

In Figure 5.4, stations with high **betweenness** scores are highlighted in red. These stations are primarily located along the river in the city center or in the Blackrock area. If an isolated, high-betweenness station like Blackrock is to close, it

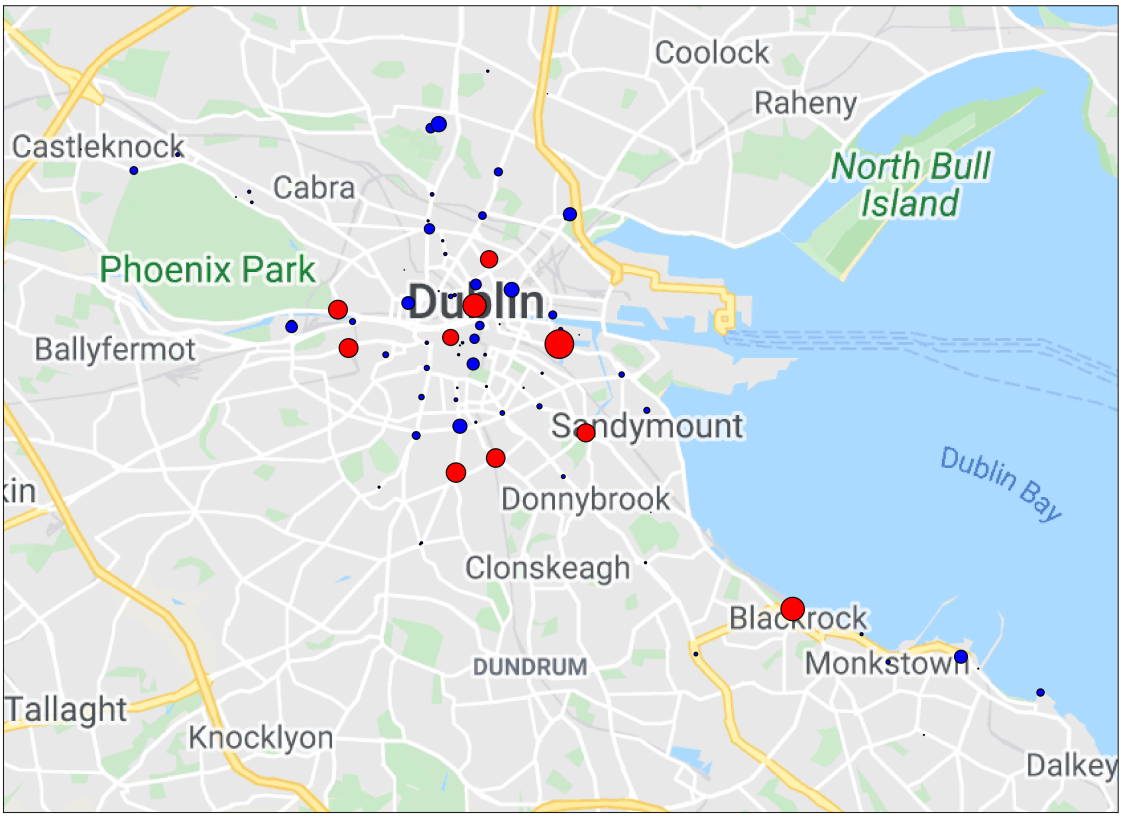


Figure 5.4: **Graph Analytics for Spatial Bike Graph Networks: Betweenness**. Each circle represents a bike station and is sized by its betweenness centrality, with the 10 highest-betweenness stations shown in red and all others in blue.

could potentially divide the network into separate segments and limiting bike rentals and movement within their respective networks.

The final analysis of the SBiGN focuses on identifying communities based solely on spatial data, as depicted in Figure 5.5, where nodes of the same color represent stations belonging to the same community. Four communities are identified: the north-east (purple), the north-west (bright aqua), the south-city group (pale chartreuse) and the south-suburban (Blackrock and Monkstown) area (red). It is interesting that three of these communities are located close to the city center, with two on the north side and one on the south side. This indicates a more widespread use of the bike-sharing system in the northern part of the city, potentially indicating that more customers utilize the bikes for work or reflecting underlying socioeconomic factors. The fourth community, located in the Blackrock-Monkstown area, is relatively isolated from the other three, being far from Dublin's city center. Trips within this community are

typically along the coastline or from nearby residential areas to the coast, with limited interaction between this community and the others. This finding is particularly interesting as it reveals the presence of isolated communities with minimal overlap, suggesting that bikes remain "trapped" within the network.

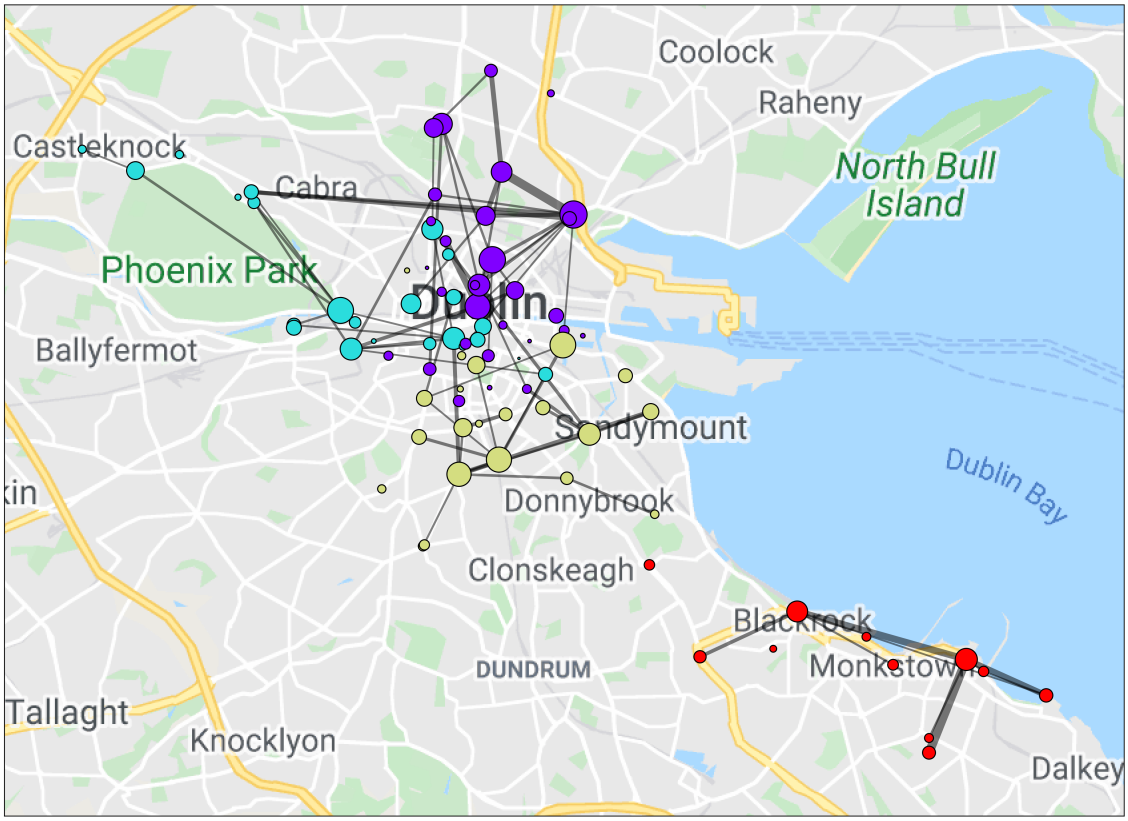


Figure 5.5: **SBiGN Community Detection**. Each circle represents a bike station and is sized according to its trip volume. The node colors indicate different communities, with four communities labeled: purple, bright aqua, pale chartreuse, and red. Lines represent routes between stations and are also sized by trip volume.

SBiGNs: Summary In SBiGNs, we addressed **Requirement 1** by providing a high-level visualization. The information contained in Tables 5.1 and 5.2, along with Figures 5.2 to 5.5, provides additional insights and context to the network visualization. These include key metrics such as **strength**, **degree**, **closeness**, and **betweenness**, as well as the identification of station **communities** and the **weights** of routes over the 15-month period.

4.2.2.2 Temporal Bike Graph Networks (Monthly TBiGN)

We employed agglomerative hierarchical clustering using Euclidean distance, as described in [238], to generate a dendrogram illustrating network changes over time. Figure 5.6 displays the results of hierarchical clustering on a series of monthly basic networks, with the x-axis representing the start date of each 4-week period. The analysis reveals three distinct clusters: the first spans the period from June 2, 2020, to November 15, 2020; the second extends from November 2, 2020, to July 4, 2021; and the third covers the timeframe from June 28, 2021, to the end of August 2021. These clusters closely correlate with two major phases of COVID-19: lockdowns (restrictions, lockdowns) and reopenings (easing). This alignment illustrates how significant events, such as the pandemic, influenced network dynamics over time in the monthly temporal graphs.

Figures 5.7 to 5.9 depict the temporal evolution of the three distinct TBiGN clusters, providing insights into the differences among these clusters over time.

- **Restrictions Cluster.** The first cluster, representing the period from June to mid-November 2020 (Figure 5.7), shows that network activity was largely concentrated at a limited number of key stations and routes, primarily involving *Phoenix Park* and *Fairview Park* to the city center. Notably, although activity is focused near the city, the biggest stations and routes are distributed across several central areas rather than being concentrated in a single location.

- **Lockdown Cluster.** During the second cluster, which corresponds to the stricter lockdown period from November 2020 to early July 2021 (Figure 5.8), overall activity levels decline significantly. The size of the top stations and routes also diminishes, with the top 10 most active stations shifting away from the western part of Dublin.

- **Easing Cluster.** In the third cluster, covering July and August 2021 (Figure 5.9), activity levels almost return to pre-lockdown levels. During this period, the most active stations are located within the city center, with major

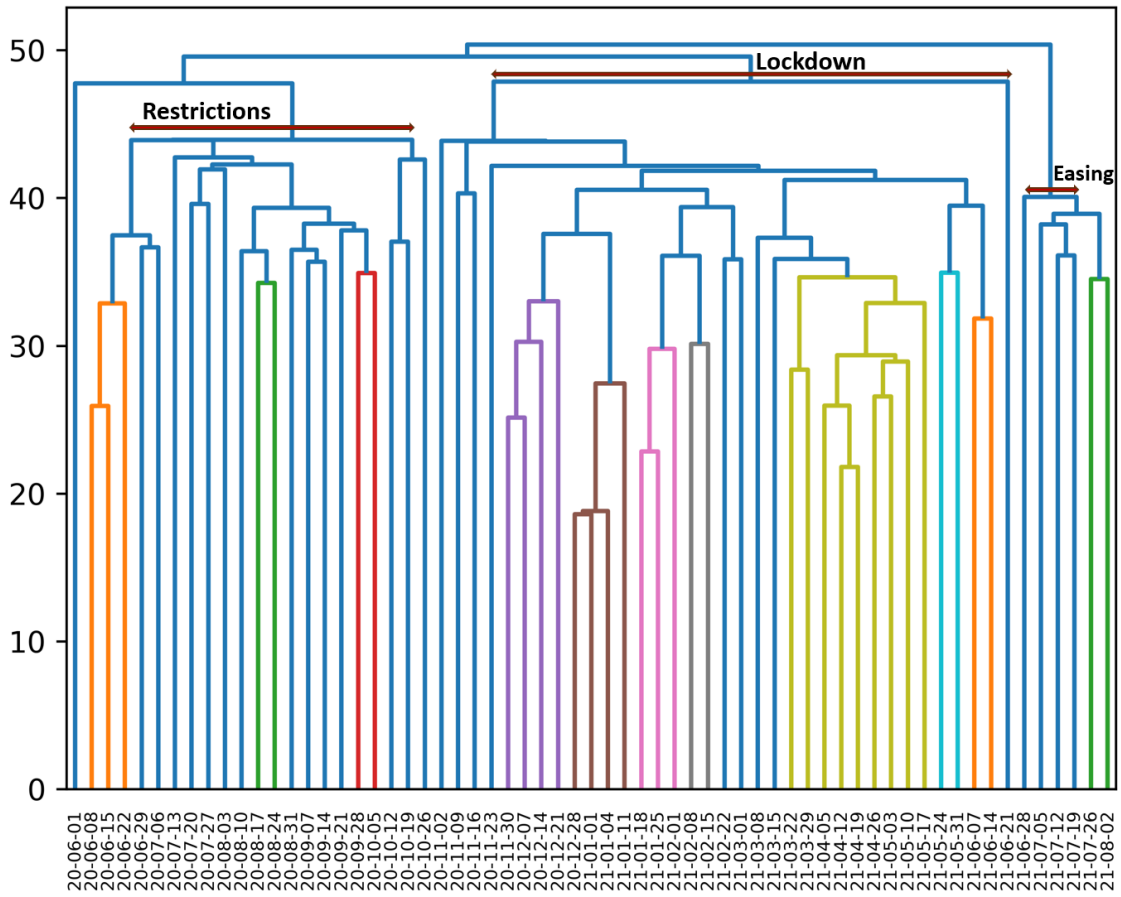


Figure 5.6: **Rolling Window Monthly Clustering**. This dendrogram shows how monthly bike-sharing networks (each represented on the x-axis by its start date) cluster based on their similarity. The y-axis indicates the distance between these monthly networks, with lower values signifying higher similarity. Horizontal lines connect two clusters at their respective distance. Three primary clusters are detected: Restrictions, Lockdown and Easing.

routes connecting residential areas to the central urban zone.

Furthermore, this type of graph addressed **Requirement 3** by clustering the monthly TBiGNs over time. Figures 5.7 to 5.9 provide information regarding the **strength** of stations and the **weight** of edges within the clusters, which correspond to the periods of COVID-19 restrictions, lockdown, and subsequent easing phases.

TBiGNs: Summary In TBiGNs, we addressed **Requirement 2** by presenting daily TBiGNs categorized and normalized into a weekday TBiGN and a weekend TBiGN. The data found in Tables B.1 and B.2, as well as Figures B.1, B.2, B.3 and B.4 provides insights into various aspects, including **strength** and **communities** of stations, **weight** of edges, and activity networks (illustrating **strength** and **weight**).

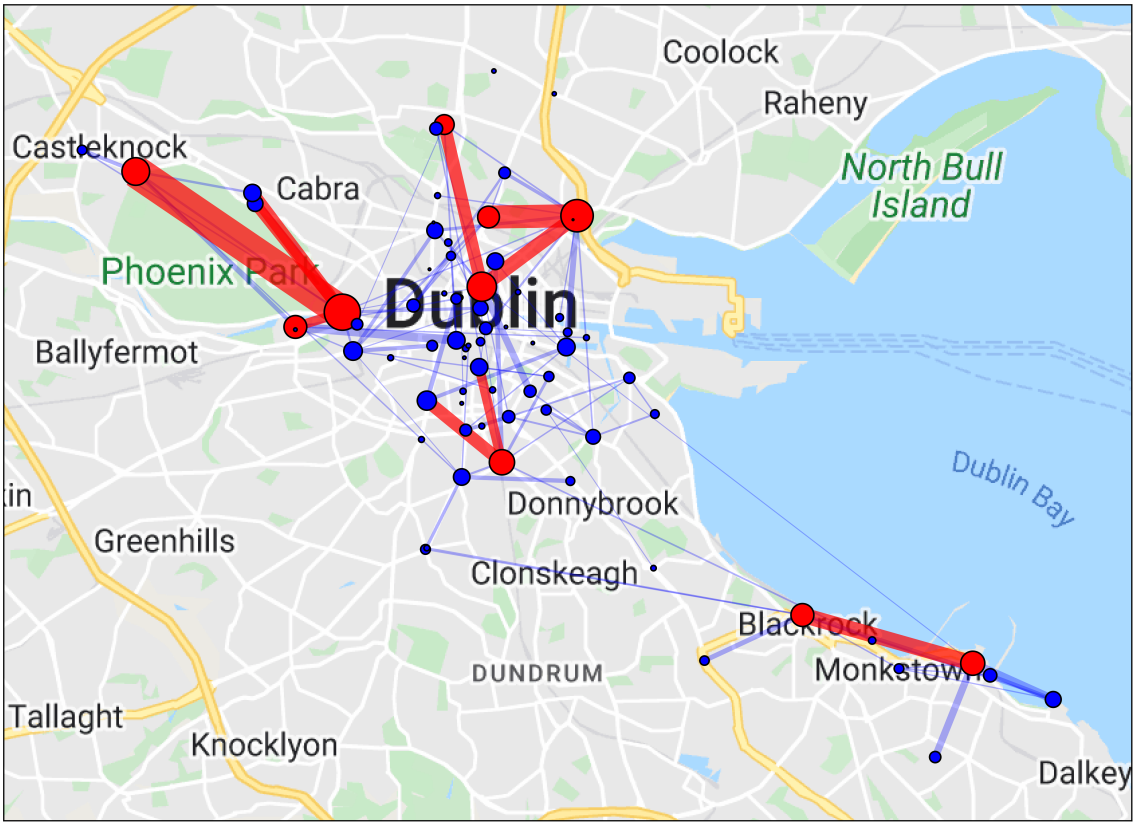


Figure 5.7: **Geographical Plot of By Month Clusters. (Representation Networks of the Restriction Cluster)**. Each circle represents a bike station and is sized according to its trip volume, with the 10 busiest stations (by trip volume) shown in red and all others in blue. Lines represent routes between stations and are also sized by trip volume; the 10 most frequently used routes are highlighted in red, and the remaining routes are shown in blue.

4.2.2.3 Spatio-Temporal Bike Graph Networks (Daily STBiGN)

Dimensional modeling and analysis [239] often follows a hierarchical approach, particularly when analyzing data over time. Higher-level abstractions can reveal periods that need more detailed investigation. A daily correlation network offers a day-by-day analysis over a specified period, where activities are aggregated by day. In this network, two stations are connected when their activities exhibit similarity based on daily comparisons. The timeseries for each station is represented as a vector of length 7, with each entry proportional to the total number of trips occurring on a specific day of the week. An edge between two stations is kept only if the Pearson correlation coefficient between their respective timeseries exceeds a threshold T. The aim is to select a low threshold while maintaining a strongly connected network.

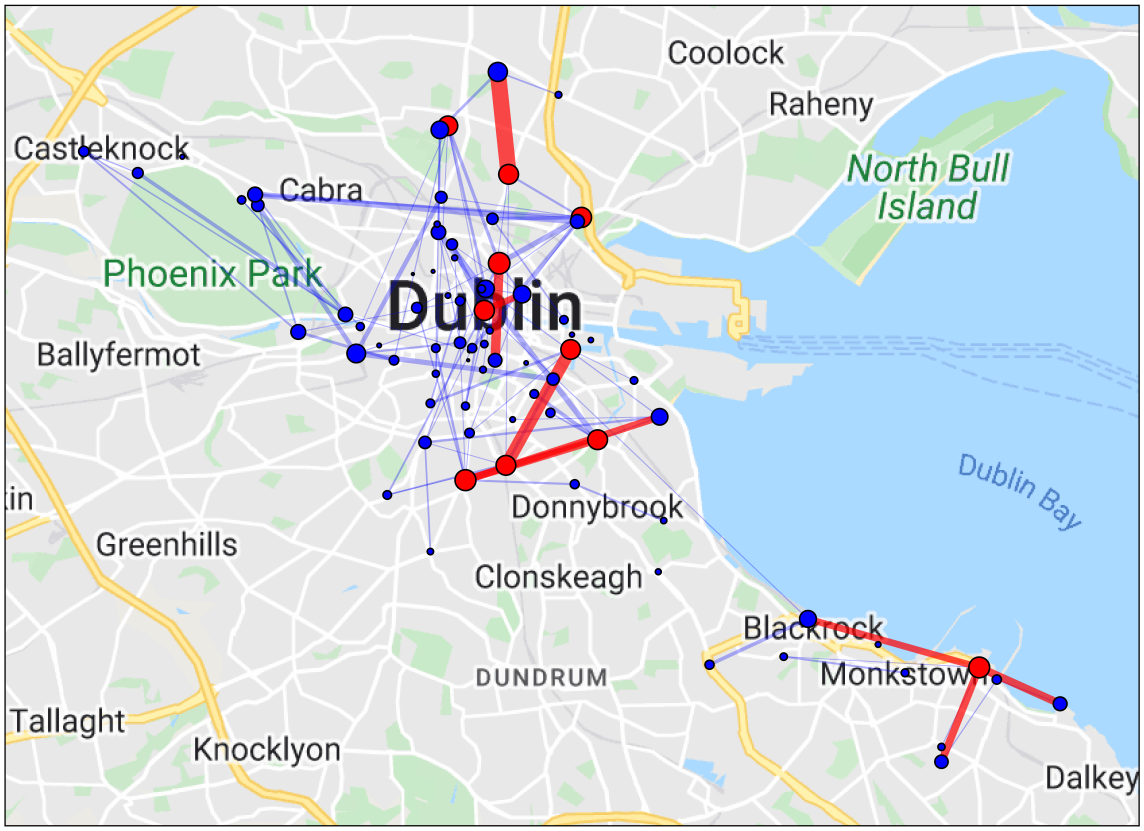


Figure 5.8: **Geographical Plot of By Month Clusters. (Representation Networks of the Lockdown Cluster)**. Each circle represents a bike station and is sized according to its trip volume, with the 10 busiest stations (by trip volume) shown in red and all others in blue. Lines represent routes between stations and are also sized by trip volume; the 10 most frequently used routes are highlighted in red, and the remaining routes are shown in blue.

For the experiments, we set $T = 0.609$, resulting in a network density $D = 0.238$. Figure 5.10 depicts the daily correlation network for the entire dataset. Only the top 5 percent of edges are visualized, with node colors representing the detected clusters. The top 10 most correlated edges are highlighted in red, while all other edges are shown in blue.

In the day-of-week correlation network (Figure 5.10), three major communities and one smaller, three-member community are identified. The cardinalities of the communities are as follows: the purple community comprises 3 stations, the bright aqua community includes 31 stations, the pale chartreuse community contains 34 stations, and the red community consists of 18 stations. The spatial distribution of these communities shows some influence from their geographic locations. The

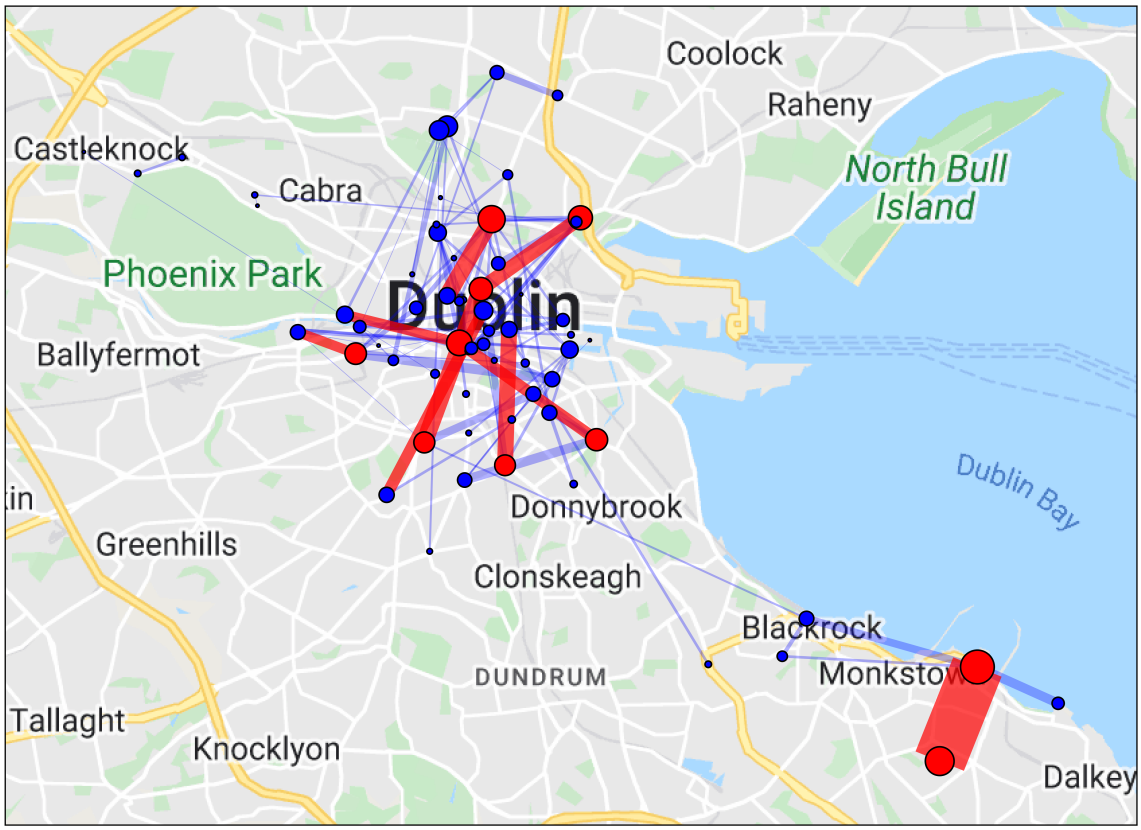


Figure 5.9: **Geographical Plot of By Month Clusters. (Representation Networks of the Easing Cluster)**. Each circle represents a bike station and is sized according to its trip volume, with the 10 busiest stations (by trip volume) shown in red and all others in blue. Lines represent routes between stations and are also sized by trip volume; the 10 most frequently used routes are highlighted in red, and the remaining routes are shown in blue.

largest community (pale chartreuse) is mainly in the south of Dublin, including the entire Blackrock-Monkstown area but also includes some stations in the Phoenix Park. The bright aqua community is primarily around the River Liffey, while the red nodes are distributed across the central and northern areas of Dublin. The pale chartreuse community demonstrates strong connectivity, with many stations showing high strength, including Castleknock, which ranks as the highest strength station. This identifies that the stations in this community exhibit highly similar activity patterns. In contrast, the bright aqua community consists of stations with noticeably lower strength and lacks strong connections. Meanwhile, although the red community has fewer members, its stations are still strongly connected.

Figure 5.11 displays the daily time series for each community, with the values

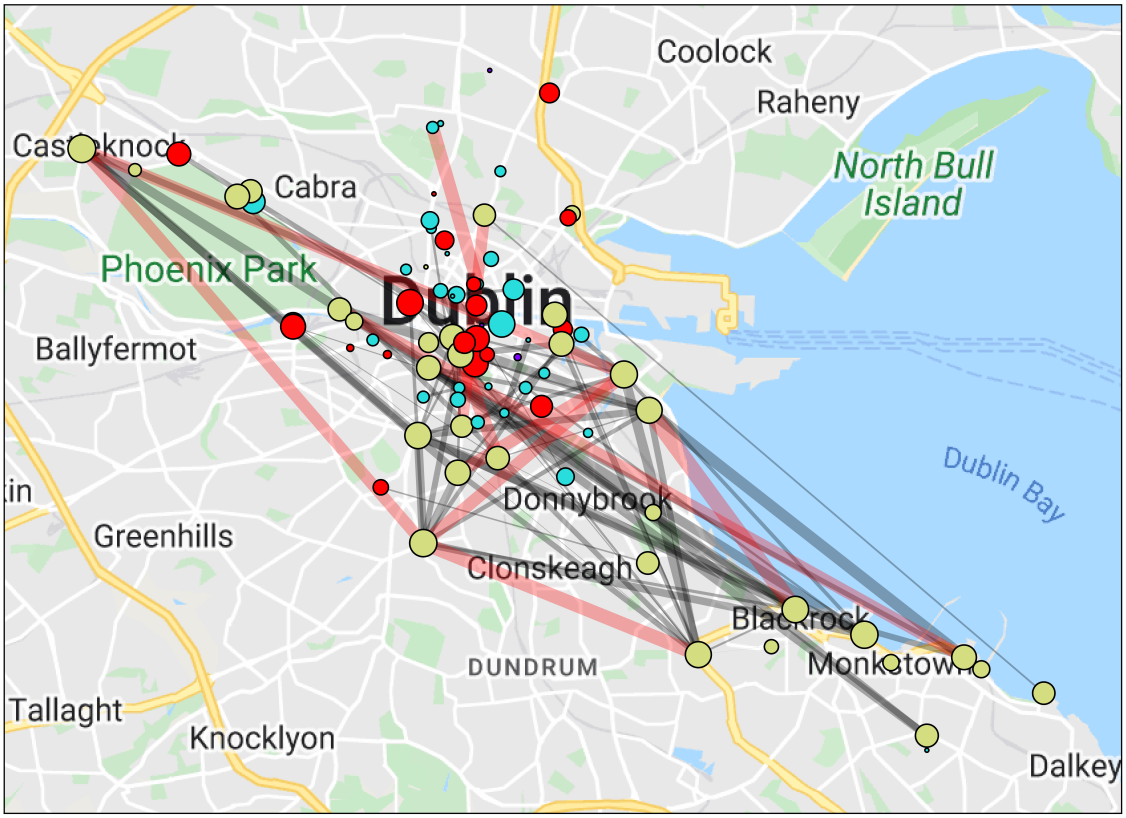


Figure 5.10: **Daily Correlation Network.** Each circle represents a bike station and is sized by its strength. The node colors indicate different communities, with four communities labeled: purple (small), bright aqua, pale chartreuse, and red. Lines represent routes between stations and are sized by their correlation score.

averaged. Generally, the Moby Move service tends to experience higher activity levels on weekends, particularly on Saturdays. Through analysis using simpler network models, it is identified that the pale chartreuse stations can be classified as *weekend* stations. These stations show very low activity during weekdays, but trip numbers increase sharply starting on Fridays, reaching their peak on Sundays. Similarly, the red stations exhibit reduced activity on Mondays and Tuesdays, with a steady level of activity throughout the remainder of the week, peaking on Saturdays. In contrast, the bright aqua stations operate as *weekday* stations, where the service is predominantly used on weekdays.

Table 5.3 follows the same format as Table B.3, but focuses on identifying the strongest stations based on daily correlations. The variance in strengths and average correlation coefficients is notably higher, ranging from 0.31 to 0.74. This greater variability indicates that more distinct patterns should emerge, leading to

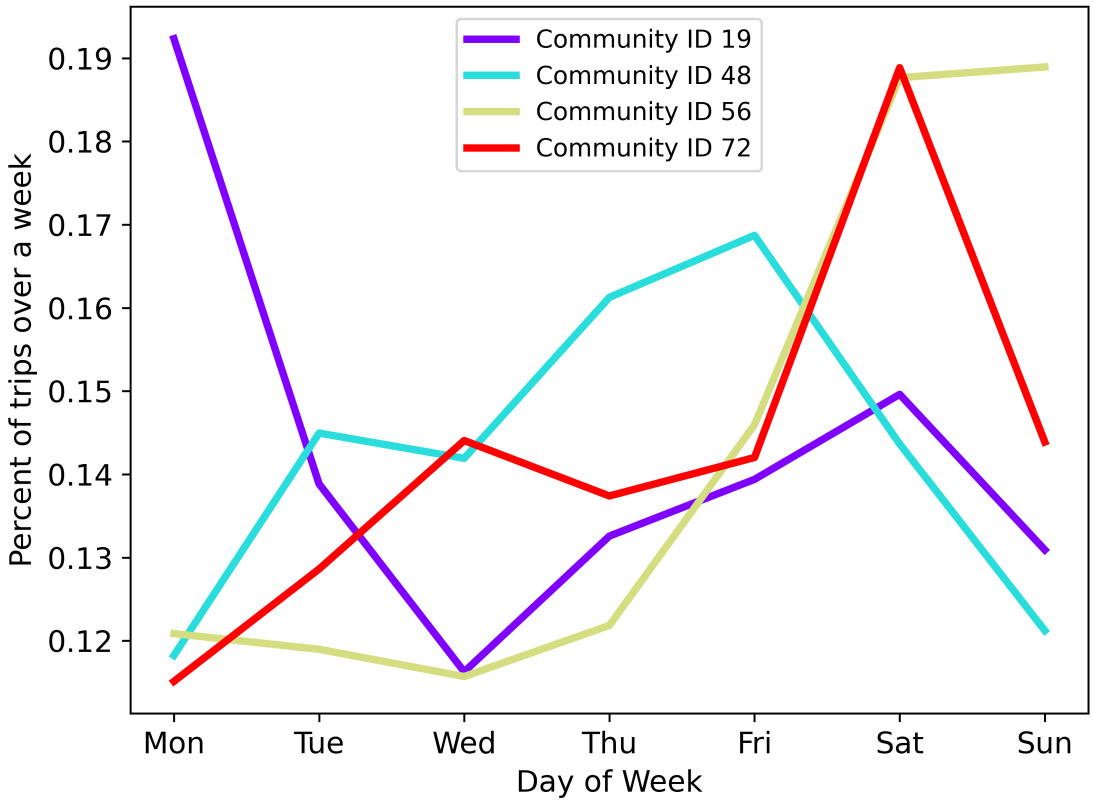


Figure 5.11: Timeseries Communities in Daily STBiGNs

the formation of larger communities within the network. This also suggests the characteristics of these communities are more distinguishable.

Table 5.3: Node (station) Strength in Daily STBiGNs

Station	**Strength**	**Avg. Cor.**	**#Pos**	**#Neu**	**#Neg**
Castleknock	63.1	0.742	62	18	5
Luke Street	62.8	0.739	63	17	5
Irishtown Rd	62.5	0.735	62	16	7
...					
Westmoreland street	34.4	0.405	6	43	36
Blessington Street	30.4	0.358	7	29	49
Warehouse	26.5	0.312	12	23	50

STBiGNs: Summary In the STBiGNs analysis, we addressed **Requirement 4** by providing correlation networks that highlight the **strength**, **weight**, and **communities**, as well as timeseries **communities** and **station strength**. These were analyzed at three temporal levels: Monthly (Figures B.5 and B.6, and Table B.3), Daily (Figures 5.10 and 5.11, and Table 5.3), Hourly (Figures B.7 and B.8, and

Table B.4). These outputs provide insights into the relationships between each station and all other stations, as well as the relationships between different communities, based on timeseries comparisons.

5.2.5 Conclusion

In this section, we introduce a graph-based modeling framework designed to analyze the dynamics of a bike sharing system. The goal is to have high-level visualizations of network activity levels; facilitate the examination of spatial regions over specified time intervals and identify stations that exhibit similar connectivity based on spatio-temporal metrics. Three types of networks are constructed: Spatial Bike Graph Networks, Temporal Bike Graph Networks, and Spatio-Temporal Bike Graph Networks. Each type provides unique insights, ranging from high-level activity patterns to granular temporal trends and correlations between station activities. The analysis revealed critical aspects such as centrality metrics, community structures, and temporal activity trends.

The Spatial Graph Networks, the simplest to construct, are effective at identifying high-traffic stations. Metrics like closeness and betweenness provide insights into popular station-to-station trips and critical "bridge" stations whose removal could segment the network, indicating where new stations may be needed. Temporal Graph Networks incorporate temporal granularity to capture both long-term trends (e.g., seasonal patterns, effects of COVID-19) and specific events. Finally, Spatio-Temporal Graph Networks allow comparison of activity patterns across stations, identifying similar behavior despite geographic differences, unique patterns, and anomalies within network communities.

The findings show that graph-based representations can capture spatio-temporal relationships and complex patterns in the data. Consequently, integrating graph structures into neural network models is promising, as these structures can guide neural networks to autonomously capture spatio-temporal information. In the next section, we take this approach by applying a novel GNN architecture to the problem

of air quality forecasting. This model leverages attention mechanisms to adaptively learn the connections between stations, thereby extracting relevant spatio-temporal features.

5.3 Air Quality Forecasting

In Section 5.2, we modeled spatio-temporal data, such as that from a bike-sharing system, using a graph-based representation. In this section, however, we aim to leverage the advantages of graph representations within a machine learning context, specifically through graph neural networks (GNNs). These models enable the automatic learning of structural features and relationships among entities within the graph.

In particular, we introduce a novel graph neural network designed to enhance spatio-temporal feature extraction for forecasting air quality. Our approach incorporates an attention mechanism that dynamically assigns different weights to various factors. By applying attention layers to both the temporal and spatial dimensions, the model is capable of simultaneously learning spatio-temporal features that affect the prediction outcome. Again, maintaining our goal of real world problems and associated data, we validate our proposed model using air quality data collected from 10 monitoring stations in Hanoi, Vietnam, spanning a period of over one year.

5.3.1 Attention Mechanisms

The core concept of our proposed neural network architecture is the incorporation of attention mechanism layers into both the temporal and spatial dimensions, stacking them in a manner that enables the model to capture spatio-temporal patterns from the input data. The attention mechanisms employed in this architecture are inspired by the self-attention approach in natural language processing [18], as well as graph attention networks [167]. Specifically, at each timestep or location, features are transformed into three distinct vector representations: query, key, and value vectors.

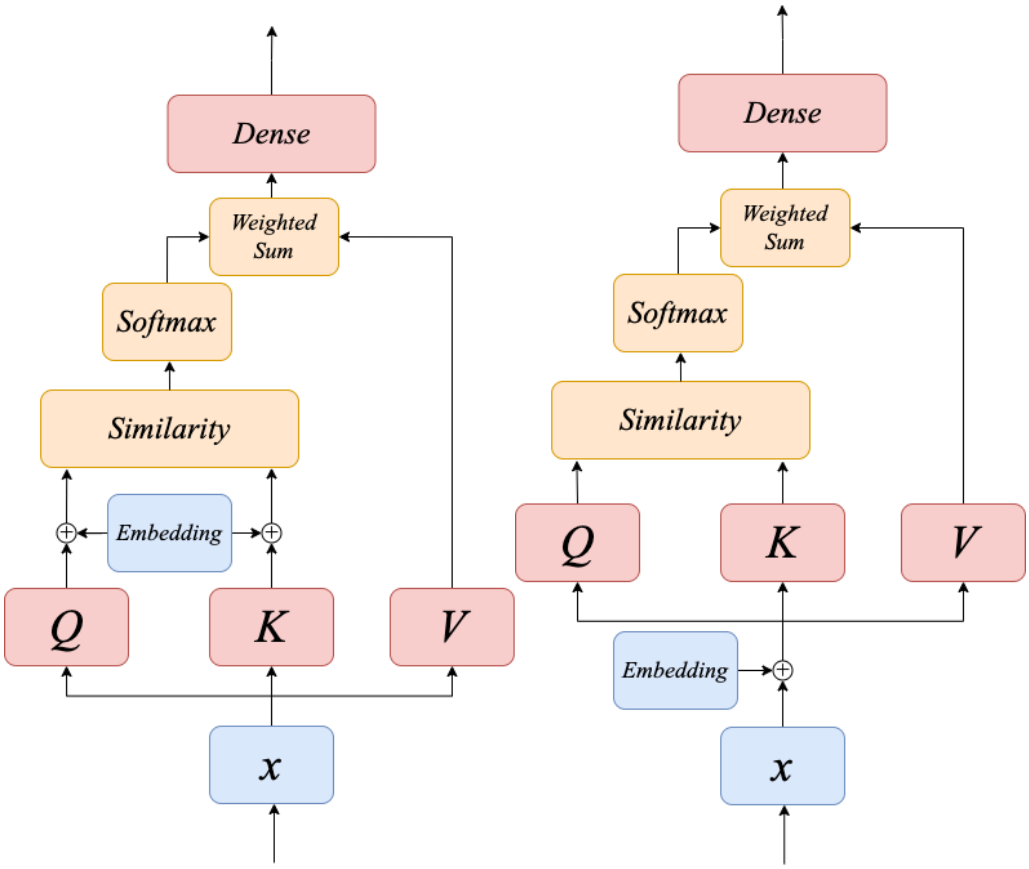
Queries and keys, along with separately trained temporal and spatial representations for each timestep or location, are used to compute the weights between any pair of timesteps or locations. The output of the layer is a weighted average of the value vectors, where the weights are determined by the calculated pairwise weights. This adaptive weighting allows the model to dynamically respond to the current state of the input, while the separate temporal and spatial representations capture the average correlations within the underlying process. The temporal and spatial layers are stacked in a structure similar to the approach used in [240], where each spatio-temporal block consists of two temporal modules with a spatial module positioned between them. In this way, the model integrates both temporal and spatial information by alternating between temporal and spatial layers.

Formally, the problem and our proposed solution are defined as follows. Let $x_0 \in \mathbb{R}^{S \times T \times F_0}$ represent the input data, where S is the number of locations, T is the number of historical timesteps, and F_0 denotes the size of the feature space. Let $y \in \mathbb{R}^{S \times 24}$ represent the air pollutant concentrations in the next 24 hours at each monitoring station, and let f represent the relationship between historical data and future air quality. Our objective is to build a model, denoted $\hat{f}$, that estimates air quality, $\hat{y} = \hat{f}(x)$, based on the historical data x. The architecture of the model comprises a series of temporal attention layers ($TemAtt$) and spatial attention layers ($SpaAtt$). Figure 5.12 shows the architecture of the layer. A detailed description of the entire architecture will be provided in the rest of the section.

Temporal Attention Layer

Assume $x \in \mathbb{R}^{T \times F}$ represents the input at a given location. The temporal attention layer processes this input and generates temporally enriched features a. This layer consists of two components: a self-attention sub-layer, which captures temporal patterns, and a feed-forward sub-layer, which further refines these temporal features.

In the self-attention sub-layer, the layer computes weights $w_{t,u}$ (Formulas 5.2) that determine the significance of timestep u to timestep t. These weights are



(a) Temporal Attention Layer (b) Spatial Attention

Figure 5.12: Temporal and Spatial Attention Layers

derived by applying a softmax function to the dot product between two nonlinear transformations of the input: the query vector Q and the key vector K. Temporal embeddings are incorporated into the query and key transformations to differentiate between timesteps.

$$\begin{aligned} Q_t &= W_Q(x_t + b_Q) + E_t \\ K_u &= W_K(x_u + b_K) + E_u \\ w_{t,u} &= \text{softmax}(Q_t \cdot K_u) \end{aligned} \tag{5.2}$$

In the Equations 5.2, W_Q and W_K are matrices of size $A \times F$, b_Q and b_K are bias vectors of size A, and E_t and E_u are temporal embeddings.

The output is then calculated as a weighted sum of another input transformation,

V, in Equations 5.3, Where $W_V \in \mathbb{R}^{F\times F}$ and $b_V \in \mathbb{R}^F$.

$$V_u = \text{ReLU}(W_V x_u + b_V), \forall u \tag{5.3}$$
$$V'_t = \sum_u w_{t,u} V_u$$

The processed temporal features are then passed through a feed-forward layer applied independently to each timestep t, described in Equation 5.4, where $W_{FF} \in \mathbb{R}^{F\times F}$ and $b_{FF} \in \mathbb{R}^F$ are the feed-forward layer parameters.

$$a_t = \text{ReLU}(W_{FF} V'_t + b_{FF}) \tag{5.4}$$

This architecture enables the layer to extract temporal features adaptively, based on the inputs at each location. It calculates the importance of different timesteps, taking into account both the input at the location and the relative time lags, as captured by the temporal embeddings. It is important to note that the above equations apply to a single location; hence, the computed weights vary across locations, allowing the model to consider spatial information embedded within the input data.

Spatial Attention Layer

Similar to the temporal attention layer, the attention mechanism is applied along the spatial dimension in the same manner, treating monitoring stations analogously to timesteps. Thus, when extracting features at a specific location, the spatial attention layer assesses the importance of other locations, including the current location itself. 0 Let $x \in \mathbb{R}^{S\times F}$ represent the input to the layer at a single timestep, where S denotes the number of locations and F represents the feature space. The output a_s for location s is computed as in Equations 5.5. Here, W_Q, W_K, W_V, W_{FF} and b_Q, b_K, b_V, b_{FF} are the model parameters of appropriate sizes, and E_s is the spatial embedding for location s.

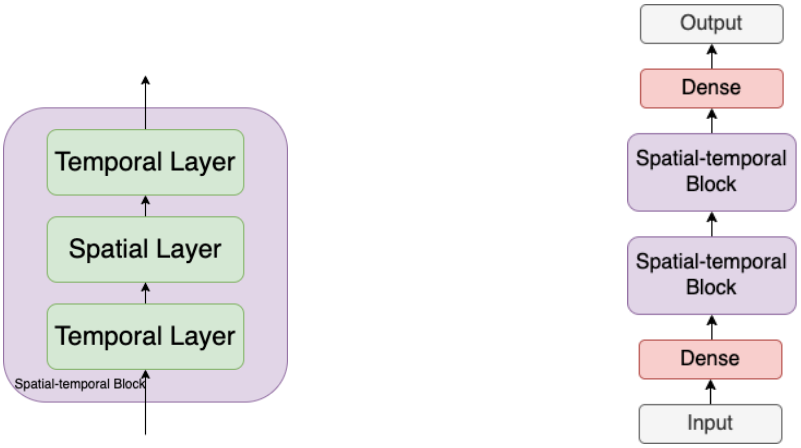
$$
\begin{aligned}
x's &= x_s + E_s \\
Q_s &= W_Q(x'_s + b_Q) \\
K_u &= W_K(x'_u + b_K) \\
V_s &= \mathrm{ReLU}(W_V x'_s + b_V), \forall u \\
w_{s,u} &= \mathrm{softmax}(Q_s \cdot K_u) \\
V'_s &= \sum_u w_{s,u} V_u \\
a_s &= \mathrm{ReLU}(W_{FF} V'_s + b_{FF})
\end{aligned}
\tag{5.5}
$$

It is important to note that these equations pertain to a single timestep. The attention weights can vary across different timesteps, allowing the model to incorporate temporal information embedded in the input.

Network Architecture

To facilitate spatio-temporal feature extraction, we structure the temporal and spatial layers into a spatio-temporal block. Following a similar approach to [240], we design the block with two temporal layers at the beginning and end, and one spatial layer in between, as depicted in Figure 5.13a. The rationale behind this design is that historical data at a given location provides a more reliable basis for predicting future behavior at that location. Thus, processing along the temporal dimension takes precedence over the spatial dimension. The block begins with a temporal attention layer to extract time-aware features before integrating information from related locations via the spatial attention layer. Subsequently, the block reconfirms these features by processing them again through a temporal layer. The outputs of this block incorporate features from both spatial and temporal dimensions. The block can be formalized as follows in equation The outputs produced by the block contain features from both spatial and temporal dimensions. The block can be formulated as Equation 5.6, where x_l is the feature vector generated by the previous layer, and

x_{l+1} is the output of the block.



(a) Spatial-temporal Attention Block (b) Entire Architecture

Figure 5.13: Proposed attention-based spatial-temporal neural network architecture

$$x_{l+1} = SPBlock(x_l) = TemAtt(SpaAtt(TemAtt(x_l))) \tag{5.6}$$

As in many traditional deep learning architectures, we stack multiple spatio-temporal blocks to enable the model to learn increasingly deep features. The complete architecture of our proposed attention-based spatio-temporal neural network is illustrated in Figure 5.13b. First, the raw inputs are transformed into a new vector space using a dense layer to facilitate feature processing in subsequent layers. The transformed inputs are then passed through a series of spatio-temporal blocks to generate spatio-temporal features. The final predictions are made using a fully connected layer that processes the outputs of the spatio-temporal blocks. Formally, let x_0 denote the raw input, and let L represent the number of spatio-temporal blocks. The air quality prediction is given by Formulas 5.7, where W_{Input}, W_{Output}, b_{Input},

and b_{Output} are the model parameters, and the activation function is set to ReLU.

$$\begin{aligned} x_1 &= \text{ReLU}(W_{Input}x_0 + b_{Input}) \\ x_{l+1} &= SPBlock(x_l), l = 1 \dots L \\ \hat{y} &= W_{Output}x_{L+1} + b_{Output} \end{aligned} \tag{5.7}$$

After conducting extensive experiments, we determined that the optimal number of spatio-temporal blocks L is 2. Using more layers increases the risk of overfitting due to the growing number of trainable parameters, while fewer layers result in shallow spatio-temporal patterns and underfitting. The optimal attention size A for the temporal and spatial layers lies between 32 and 64, while the hidden size should be in the range of 128 to 256. It is important that the attention size remains smaller than the feature size to prevent the model from overreacting to minor changes in the input, while the feature size should be sufficiently large to capture more information. Of course choosing these hyperparameters that are too small or too large can lead to underfitting or overfitting during model training. We set the number of timesteps T to 7 days, enabling the model to learn weekly patterns, as longer historical data did not improve performance and increased computation time.

Given that air pollutant concentration forecasting is a regression task requiring a real-valued output, we train our model using the least mean squared error (MSE) loss function. The model parameters are optimized via backpropagation using the Adam optimizer [203], with a learning rate of 10^{-4}. Early stopping is employed when the validation loss ceases to decrease.

5.3.2 Experiments

In this section, we begin by providing a detailed description of the dataset used to evaluate our proposed method. This is followed by a discussion on the preprocessing steps, including the generation and splitting of the training samples. Then we outline the experimental setup and present the results using appropriate evaluation metrics.

Data Preprocessing

The dataset used for the experiments is described in Section 2.2.2.

To prepare the data for neural network training, we utilize a sliding window approach with a window size of $T + 24$, where T represents the number of historical timesteps, and 24 corresponds to the number of hours to be predicted into the future. The window slides over time, generating one training sample for each window. With $T = 7$ days, this process yields 12,940 data samples. For missing values, we apply short-term interpolation by considering data up to 3 days before and after the missing values when performing linear interpolation, adjusted for weekly periodicity. For larger gaps, we either fill the missing data with seasonal mean values or apply masking during training and prediction, provided the models support masking.

To train and evaluate the models, we divide the dataset into three subsets: the training set, used for model training; the validation set, employed to select optimal hyperparameters; and the test set, used to evaluate model performance. Specifically, we use the first 12 months of data, accounting for approximately 66% of the dataset, for training. The subsequent 3 months (roughly 16%) are used for validation, and the final 3 months are reserved for testing. This train-validation-test split reflects real-world applications, where air quality prediction models are trained and validated on recent data and then deployed for predictions over a few months before being retrained.

Experiments

For comparison, we implement several baseline models ranging from traditional machine learning approaches to more complex architectures commonly used in timeseries-related tasks. These models include XGBoost [241], linear regression (LR), multi-layer perceptrons (MLP), support vector machines (SVM), and long short-term memory networks (LSTM) [163]. In addition, we re-implement established models specifically designed for air quality prediction, such as GNN-LSTM [90] and GLSTM [173]. All models are re-implemented using Pytorch in Python, with the

exception of XGBoost, for which we utilize its own Python package. Regarding our proposed approach, we train three different architectures: temporal attention models (TemAtt), spatial attention models (SpaAtt), and the proposed spatio-temporal attention models (SpaTemAtt). All models are trained on the same training dataset, with hyperparameter tuning and model selection based on performance on the validation set. For each model type, we conduct extensive training and hyperparameter optimization to ensure optimal performance.

We evaluate the performance of our methods using standard metrics commonly applied in regression tasks: root mean squared error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). Detailed descriptions of these metrics are provided in Appendix A.1.

Results

Table 5.4 show the results of the experiment. From the table, it can be seen that the performances are quite similar across different models, with only slight variations. The Temporal Attention (TemAtten) model achieves the lowest RMSE on the validation set (28.06), suggesting its effectiveness in capturing temporal patterns during training. However, on the test set, the Spatial-Temporal Attention (Spa-TemAtten) model yields the best RMSE (29.01), marginally outperforming the GLSTM model (29.06). Interestingly, the simple Linear Regression (LR) model achieved the lowest MAE on the validation set (14.75) and the second-best MAE on the test set (18.46), closely behind the LSTM-GNN model, which recorded the lowest test set MAE (18.42). Comparing the LSTM and Temporal Attention (TemAtten) models, we observe that TemAtten outperformed LSTM on the validation set in terms of RMSE (28.06 vs. 30.16), indicating that the attention mechanism enhances the model's ability to learn temporal features during validation. Notably, the Spa-TemAtten, LSTM-GNN, and GLSTM models exhibited strong performance on the test set, demonstrating their effectiveness in simultaneously capturing both spatial and temporal dimensions. Despite the expectation that advanced models incorporating

attention mechanisms and graph structures would significantly outperform simpler models, the improvements observed are marginal. This finding suggests that, in scenarios with limited available training data, such as this forecasting task, the advantages of more complex architectures may be diminished.

Table 5.4: **Air quality 24-hour forecasting performance**. Average metrics computed over a 24-hour period with hourly predictions. Lower metric values indicate better performance. The best result in each column is highlighted in **bold**, while the second best is underlined.

Model	RMSE-Valid	MAE-Valid	RMSE-Test	MAE-Test
LR	30.14	**14.75**	29.24	18.46
MLP	29.14	15.68	30.15	20.23
LSTM	30.16	15.31	29.57	18.93
LSTM-GNN	30.19	15.31	29.52	**18.42**
GLSTM	30.11	15.21	29.06	18.48
TemAtten	**28.06**	15.60	31.05	21.65
SpaAtten	30.01	15.70	29.47	18.98
SpaTemAtten	28.55	14.80	**29.01**	18.57

Furthermore, examining the optimal configurations of the TemAtten, SpaAtten, and SpaTemAtten models reveals patterns regarding the temporal and spatial dependencies captured by the models. For the TemAtten model, the best-performing time windows are [24, 48, 168] hours, while for the SpaTemAtten model, they are [24, 48, 24, 48]. This indicates that, although long-term dependencies (e.g., 168 hours, or one week) along the time dimension are important for accurate predictions, the model still prioritizes short-term information (24 and 48 hours) before incorporating longer-term patterns. This sequential approach suggests that the model first captures local short-term dynamics before extending its focus to more distant time intervals, thereby improving its temporal feature extraction. Similarly, in the case of spatial attention models, the optimal distance thresholds for neighboring stations are [2, 4] kilometers for SpaAtten, and [2, 4, 2, 4] kilometers for SpaTemAtten. This implies that the model focuses initially on nearby stations (within 2 kilometers) before expanding its attention to those at greater distances (up to 4 kilometers).

5.3.3 Conclusion

In this project, we have proposed an attention-based deep learning architecture specifically tailored to handle spatio-temporal data, and we have validated its effectiveness in the context of air quality forecasting. The methodology incorporates attention mechanisms in both the temporal and spatial dimensions, enabling the model to effectively learn spatio-temporal features. The model processes the input through a series of spatio-temporal blocks, each consisting of alternating temporal and spatial attention layers, starting with the temporal layer.

In these attention layers, nodes represent either timesteps or locations, while the directed edge weights reflect the relative importance of one node to another. The output at each node is calculated as a weighted average of the features from its neighboring nodes, including itself. We also introduce a spatio-temporal mask, which limits the attention to nodes within a certain radius. This radius expands as the model progresses through deeper layers, allowing the model to focus on nearby, more relevant factors first—those that are likely the primary drivers of forecasting—before considering influences on a larger scale.

The proposed architecture not only demonstrates improved predictive performance but also enhances model interpretability. It addresses several shortcomings of existing approaches, such as the slow computational speed and weak long-term dependency modeling of LSTM networks, as well as the static nature of traditional graph neural networks. However, the results also indicate that the performance improvement is marginal compared to simpler models like linear regression. This suggests that the proposed architecture may offer limited benefits when the available training data is small.

5.4 Conclusions

In this chapter, we explored the use of graph-based representations in two distinct projects. In the first project, we demonstrated the effectiveness of graph structures

for analyzing bike-sharing data. This representation provides a more intuitive and insightful way to analyze data compared to traditional tabular formats, offering unique perspectives on the underlying relationships.

In the second project, we extended the concept of spatio-temporal graphs by integrating it into a newly proposed neural network architecture. In this architecture, attention layers treat locations or time steps as nodes in a graph, and adaptively calculate the weights between them based on the input data and inherent characteristics of the stations or time lags. This approach allows for the automatic extraction of spatio-temporal features, adapting to the specific properties of the input.

While the results are promising, the use of graph-based neural networks for problems with sparse data does not lead to significant improvements. The model is prone to overfitting, particularly when working with small or noisy time series datasets. This is due to the relatively large architecture compared to conventional models. Stronger regularization techniques are essential to mitigate overfitting and ensure proper generalization. In physics, it is often assumed that data follows certain governing equations, which can serve as a powerful form of regularization. In subsequent chapters, we will explore this approach, aiming to enhance model generalization by incorporating such domain-specific assumptions.

Chapter 6

Physics Informed Neural Networks

In the previous two chapters, we explored the capabilities of neural networks across different domains. In Chapter 4, challenges from two domain areas, medical and human performance, were examined using fairly traditional neural networks with varying degrees of performance. However, pure neural network-based approaches, with the complexity of the models, exhibit limitations such as poor interpretability [91], poor generalization [28], and the necessity for substantial training data [26]. In Chapter 5, we then examined graph based machine learning and a graph-neural network combination, which again provided interesting results and outcomes but it is still possible that the full power of neural networks is not being reached. For this reason, we adopt a different approach in this chapter which is physics-informed neural networks (PINN), a more integrated methodology of neural networks and dynamical models which can be used in domains such as epidemiology [242], entomology [243], fluid dynamics, material science [26], to name but a few.

The chapter is structured as follows: in Section 6.1, we present our hypotheses for addressing the current limitations of PINNs; Section 6.2 describes our PINN framework as applied to a system of ordinary differential equations (ODEs), incorporating our proposed enhancements; in Section 6.3, we conduct an ablation study using the Lorenz system to assess the contributions of each component in our framework; Section 6.4 explores the broader impact and potential applications of our approach through a model of mosquito population dynamics; and finally, we provide concluding

remarks in Section 6.5.

6.1 Introduction

The goal of this chapter is to investigate the integration of domain knowledge, specifically in the form of physics equations, into data-driven methodologies, with the chosen application area of mosquito population modeling. This ODE system serves as an useful test case for the PINN approach. It was chosen as validation for not only for its practical application in real-world but also for its characteristics, including multi-scale behavior and an exceptionally large input domain, which pose significant challenges for the PINN framework.

By incorporating the equations of dynamical systems, this approach potentially reduces data requirements while preserving the robustness of deep learning. However, current Physics-Informed Neural Network (PINN) frameworks have not yet reached sufficient maturity for real-world ODE systems, particularly those exhibiting extreme multi-scale behavior such as mosquito population dynamics [26, 41, 42]. For instance, [42] demonstrated that PINN convergence can be hindered by gradient imbalances arising from multiple loss terms. Similarly, [111] showed that training PINNs over large domains with complex solutions is challenging, requiring the domain to be broken down. Given such difficulties, we pose the following research question: *Can neural networks be extended to accurately represent and predict the behavior of dynamical systems governed by a system of ordinary differential equations?*

This chapter addresses one of our fundamental research questions through several research tasks aimed at enhancing convergence and accuracy when simultaneously training with data and physics losses. These tasks involve improvements specifically tailored for ODE systems. Firstly, we propose that normalizing individual ODE equations, in addition to the traditional normalization of neural network inputs and outputs, can significantly improve PINN training efficiency. Secondly, we propose that balancing the gradients back-propagated from the losses to the training weights is crucial for achieving a convergence of the optimization. Furthermore, for scenarios

with limited training data (e.g., only initial conditions), gradually expanding the training time domain may facilitate the system's adherence to the inherent time causality of ODE systems. Lastly, training separate models for distinct input domains could reduce optimization complexity, thereby enhancing convergence and accuracy. Based on these assumptions, a PINN framework was developed with several improvements for forward and inverse problems for ODE systems with a case study application in modeling the dynamics of mosquito populations. Here, forward problems involve predicting system behavior given known parameters, whereas inverse problems focus on estimating unknown parameters from observed data. To evaluate our approach, we conduct experiments using **simulated data** from the Lorenz system and our mosquito population case study.

The implementation of our PINN framework, including experiments, is available on GitHub: `https://github.com/dinhvietcuong1996/pinn-mosquito`.

6.2 PINN Framework Development

This section begins with a formulation of the problem over a number of steps, followed by a description of additional techniques which further enhance the performance of the framework.

6.2.1 PINN Structure

Let $u(t)$ denote a V-dimensional vector representing the state of a dynamical system at any given time $t \in [0, T]$. This dynamical system u is governed by a set of F ODEs as expressed in Equation 6.1.

$$\frac{du}{dt} = f^{(i)}\left(t, u, \theta\right), \quad i = 1, 2, \ldots, F. \tag{6.1}$$

In Equation 6.1, the system's behavior is influenced by a set of parameters θ, which might or might not be known in advance. The functions $f^{(i)}$ are pre-

defined and characterize the system's dynamics. Given a set of observations $\mathcal{D}_u = \{(t_1, u_1), (t_2, u_2), \ldots, (t_j, u_j), \ldots\}$ at various time points, our objective is to determine a solution u and potentially θ that simultaneously satisfies these observations while maintaining consistency with the governing ODEs.

In the conventional PINN [43], a neural network U (parameterized by W_U), is used to approximate the solution u. For inverse problems where some or all parameters θ are unknown, neural networks $\Theta^{(l)}$ are used to predict these unknown values θ_l. For simplicity, the set of all parameters, including the known or ones to be learnt by neural networks, are denoted by Θ and treated as neural networks. Let $W = W_U, W_\Theta$ represent the complete set of trainable parameters. The task can then be formulated as an optimization problem, where training the parameters W aims to minimize a multi-task objective function defined in Equation 6.3.

$$W = \text{argmin}_W \mathcal{L} \tag{6.2}$$

$$\mathcal{L} = \mathcal{L}_{data} + \frac{1}{F} \sum_{i=1}^{F} \lambda_i \mathcal{L}_{f^{(i)}} \tag{6.3}$$

The loss terms in 6.3 are defined in equations 6.4 and 6.5.

$$\mathcal{L}_{\text{data}} = \frac{1}{N_u} \sum_{(t_i, u_i) \in \mathcal{D}_u} (U(t_i) - u_i)^2 \tag{6.4}$$

$$\mathcal{L}_{f^{(i)}} = \frac{1}{N_f} \sum_j \left| \frac{dU}{dt} - f^{(i)}\left(t_j, U(t_j), \Theta(t_j)\right) \right|^2 \tag{6.5}$$

In the Equations 6.4 and 6.5, $U(t_j)$ and $\Theta(t_j)$ represent the neural network outputs evaluated at time t_j. Minimizing $\mathcal{L}_{\text{data}}$ reduces the discrepancy between network predictions and observed data, while minimizing $\mathcal{L}_{f^{(i)}}$ ensures the neural network U adheres to the differential equations with minimal error. The weights λ_i serve as balancing factors between data fitting and physics dynamics. N_f denotes the number of residual points randomly sampled from a distribution μ, typically uniform.

We resample the residual points at every training step to ensure that the losses are minimized everywhere in the entire period. Overall, by minimizing this total loss function $\mathcal{L}$, the networks can simultaneously fit observed data and approximate the underlying dynamic rules.

The optimization of the loss function is achieved through gradient-based algorithms. These algorithms iteratively update the parameters W in directions that reduce the loss function, based on the gradients $\nabla_W \mathcal{L}$. The update rule is formulated as in Equation 6.6.

$$W_{\text{new}} = W_{\text{old}} - \eta \nabla_W \mathcal{L} = W_{\text{old}} - \eta \left(\nabla_W \mathcal{L}_{data} + \sum_{i=0}^{F} \lambda_i \nabla_W \mathcal{L}_{f^{(i)}} \right) \tag{6.6}$$

where η denotes the learning rate, which determines the magnitude of each step taken in the direction opposite to the gradient. For this optimization process, the Adam variant of gradient descent in PINN training [203] is used. The computation of gradients, whether it involves differentiating the network U with respect to time t or the loss function $\mathcal{L}$ with respect to W, is computed by automatic differentiation provided by deep learning frameworks such as PyTorch [8].

The universal approximation theorem [30] posits that Multi-Layer Perceptrons (MLPs) [244] can approximate any continuous function on a given domain, provided sufficient complexity characterized by the number of units. For an MLP comprising L layers, the mathematical formulation describing its layer-wise operation is presented in Equation 6.7.

$$\begin{aligned} h^{(0)} &= x \\ h^{(l)} &= \sigma(W^{(l)} h^{(l-1)} + b^{(l)}), l = 1, \ldots, L-1 \\ h^{(L)} &= W^{(L)} h^{(L-1)} + b^{(L)} \end{aligned} \tag{6.7}$$

Here, x denotes the network input, $h^{(l)}$ represents the hidden state at layer l, $W^{(l)}$ and $b^{(l)}$ are the parameters of layer l, and σ is the activation function. We employ the GELU activation function [245] due to its smooth properties, which are crucial for

differential problems and offer advantages over the ReLU function. The initialization of the MLP's weights and biases is performed using the Glorot scheme [246].

While the "vanilla" PINN framework described above demonstrates adequate approximation capabilities for relatively simple ODE systems, it often encounters difficulties in converging to satisfactory solution when dealing with more complex systems, particularly those exhibiting extreme stiffness, chaotic behavior, or multi-scale characteristics. For these reasons, approaches for enhancing PINN training in both forward and inverse problems involving ODE systems, are now presented.

6.2.2 ODE Normalization

Neural networks typically operate optimally with inputs and outputs in the range [-1, 1]. For values outside this range, data normalization plays a critical role in the machine learning workflow, ensuring that both input and output variables remain within a suitable range. In the context of PINNs, this normalization process requires careful consideration, as any transformation necessitates corresponding modifications to the ODE systems. For this reason, a systematic approach to normalization using PINNs with ODEs was designed during this research. Specifically, we employ the MIN-MAX normalization scheme for input and output variables, namely t, u, and θ.

For the input variable (time t), we apply the transformation presented in Equation 6.8, where $[T_{\min}, T_{\max}]$ denotes the time domain over which the models are trained. This transformation effectively normalizes the time variable t to the range [-1, 1].

$$t' = 2 \cdot \frac{t - T_{\min}}{T_{\max} - T_{\min}} - 1 \tag{6.8}$$

For the normalization of neural network outputs, we define $\mathfrak{L}_u, \mathfrak{U}_u, \mathfrak{L}_\theta$, and $\mathfrak{U}_\theta$ as the lower and upper bounds for u and θ, respectively. These bounds are dimension-specific within the dynamical system. In cases where the ranges are small, we adjust

the bounds around the mean as $\mathfrak{L} = \min(\mathfrak{L}, \mathfrak{M} - 1)$ and $\mathfrak{U} = \min(\mathfrak{U}, \mathfrak{M} + 1)$ where $\mathfrak{M} = \frac{\mathfrak{L}+\mathfrak{U}}{2}$ is the middle point. These bounds can be estimated through various methods, including analysis of collected data, inference from domain knowledge, or via approximate simulations. The normalization of variables u and θ at a normalized time t' is subsequently computed using Equations 6.9 and 6.10, respectively.

$$u'(t') = \frac{u(t') - \mathfrak{L}_u}{\mathfrak{U}_u - \mathfrak{L}_u} \cdot 2 - 1 \Longleftrightarrow u(t') = (u'(t') + 1)\frac{\mathfrak{U}_u - \mathfrak{L}_u}{2} + \mathfrak{L}_u \tag{6.9}$$

$$\theta'(t') = \frac{\theta(t') - \mathfrak{L}_\theta}{\mathfrak{U}_\theta - \mathfrak{L}_\theta} \cdot 2 - 1 \Longleftrightarrow \theta(t') = (\theta'(t') + 1)\frac{\mathfrak{U}_\theta - \mathfrak{L}_\theta}{2} + \mathfrak{L}_\theta \tag{6.10}$$

Following the normalization above, the transformed variables t', u', and θ' now vary within the range [-1, 1]. Thus, it becomes advantageous to employ neural networks U and Θ as surrogate models for u' and θ', rather than directly approximating u and θ. After the transformations in Equations 6.9 and 6.10, it is necessary to adapt the loss functions accordingly. When computing $\frac{du'}{dt}$ and applying Equation 6.1, we arrive at Equation 6.11.

$$\frac{du'}{dt} = \frac{2}{\mathfrak{U}_u - \mathfrak{L}_u}\frac{du}{dt} = \frac{2}{\mathfrak{U}_u - \mathfrak{L}_u} f^{(i)}\left(t, u, \theta\right) \tag{6.11}$$

Instead of using objective functions 6.4 and 6.5, we now consider the alternative formulations 6.12 and 6.13.

$$\mathcal{L}_{\text{data}} = \frac{1}{N_u}\sum_j \left(U(t'_j) - \frac{u_j - \mathfrak{L}_u}{\mathfrak{U}_u - \mathfrak{L}_u} \cdot 2 - 1\right)^2 \tag{6.12}$$

$$\mathcal{L}_{f^{(i)}} = \frac{1}{N_f}\sum_j \left|\frac{dU}{dt} - \frac{2}{\mathfrak{U}_u - \mathfrak{L}_u} f^{(i)}\left(t_j, (U(t'_j) + 1)\frac{\mathfrak{U}_u - \mathfrak{L}_u}{2} + \mathfrak{L}_u, (\Theta(t'_j) + 1)\frac{\mathfrak{U}_\theta - \mathfrak{L}_\theta}{2} + \mathfrak{L}_\theta\right)\right|^2 \tag{6.13}$$

These modifications to the objective functions not only attempt to balance the

data loss and ODE terms, but also *across* ODE components. This re-scaling strategy proves particularly advantageous in scenarios where variables exhibit substantial scale differences, whether due to their intrinsic characteristics or the units of measurement. By normalizing these variables, the risk of any single variable dominating others is mitigated and thus, equilibrating the impact of each term on the training process. This is similar to the concepts and assumptions in the Glorot initialization for neural networks [246].

6.2.3 Gradient Balancing

As this is a multi-task problem, it has different objective functions (e.g. Equation 6.3). A previous study [42] identified gradient imbalance between $\nabla_W \mathcal{L}_{data}$ and $\nabla_W \mathcal{L}_{f^{(i)}}$ in the update rule 6.6 as a critical failure point in PINN training. The differential equation residual loss often dominates due to system *stiffness*, causing the model to prioritize ODE constraint optimization over matching initial, boundary, or data observations. This can lead to convergence to a trivial *null solution* that violates data conditions. We address this issue by adopting and extending the approach in [42], adjusting the weights λ based on statistics from the gradients $\nabla_W \mathcal{L}_{data}$ and $\nabla_W \mathcal{L}_{f^{(i)}}$. Our extension to ODE systems involves individually assigning and adjusting weights for each differential equation. We set the weights for data to 1 and each of the component weights λ_i is adjusted individually, to ensure that every equation in the system is given equal *importance* while still aligning with the *importance* given to data. This extension is crucial, as the scale and complexity of each equation differs, necessitating distinct treatment during training.

This gradient-balancing algorithm is described in Algorithm 6.1. All weights λ_i are initialized to 1 and updated every N steps. The weight $\hat{\lambda}_i$ is computed as the ratio between the mean of absolute values of gradient $\nabla_W \mathcal{L}_{data}$ and the maximum of the absolute values for the gradients of $\nabla_W \mathcal{L}_{f^{(i)}}$. To mitigate the inherent volatility of gradient descent, we update weights λ_i using a moving average formula (Equation 6.15). The recommended values for hyper-parameters α and N in

the original study [42] are $\alpha = [0.5, 0.9]$. However, we set $N = 100$, and tune α in extreme cases $\alpha = 0.99, 0.9, 0.5$, or set $N = 1$ with $\alpha = 0$.

Algorithm 6.1 gradient_balancing()

Require: step $\leftarrow$ current training step; α smoothing factor; updates are made every N steps

Ensure: re-calculating λ_i such that gradients from different loss terms are balanced

if step $= 0$ **then**

Initialize $\lambda_i \leftarrow 1, \forall i = 1, \ldots, F$

end if

if step $\mod N = 0$ **then**

Compute $\hat{\lambda}_i$ by

$$\hat{\lambda}_i \leftarrow \frac{\overline{|\nabla_W \mathcal{L}_{data}|}}{\max\left\{\left|\nabla_W \mathcal{L}_{f^{(i)}}\right|\right\}}, i = 1, \ldots, F \tag{6.14}$$

where $\overline{|\nabla_W \mathcal{L}_{data}|}$ is the average of the absolute gradients over all model parameters W.

Adjust the weights $\hat{\lambda}_i$ by

$$\lambda_i \leftarrow \alpha \cdot \lambda_i + (1 - \alpha)\hat{\lambda}_i, i = 1, \ldots, F \tag{6.15}$$

end if

6.2.4 Causal Training

When training PINNs, particularly in scenarios with sparse data, it is crucial to consider the causal effect [110]. Conventional PINN models are typically trained to adhere to differential equations across the entire input domain (i.e. time domain $[0, T]$) simultaneously. However, this can lead to a problematic scenario where the model satisfies these rules at later time points t without proper management at earlier points. This discrepancy creates a situation where efforts to conform to ODEs at one point may result in violations at others during the learning process. Furthermore, if the penalty for not following ODEs at earlier points outweighs the fitting at later points, the model may get stuck in a local minimum, no longer capable of satisfying ODEs over the entire period. To address this issue, we adopt a *causal* approach to training where the task of meeting the data conditions is given priority while ensuring that the model complies with ODEs on a small time window near $t = 0$

first before gradually extending the window toward $t = T$.

This study implements a three-phase training process: data fitting, progressive causal training, and final tuning described as follows:

1. Firstly, the data fitting phase focuses exclusively on matching data conditions by training with the data loss term $\mathcal{L}_{\text{data}}$. This establishes a starting point for the model, facilitating subsequent adherence to differential equations.

2. The progressive causal training phase adopts the *growing-interval* approach from [185]. We *gradually* teach the models to follow the differential equations starting from a small interval and slowly covering more domain as the training progresses. Both data and ODE loss terms are included, with ODE residual points t_j for $\mathcal{L}_{f^{(i)}}$ in Equation 6.5 sampled from an expanding interval. Given N_2 update steps in this phase, at the n_2-th step, residual points are uniformly sampled from the interval $\left[0, \frac{n_2}{N_2} \cdot T\right]$.

3. The final tuning phase employs both loss terms, with residual points sampled from the entire time period. This phase refines and enhances the learned solutions.

The training process mainly happens during the second and third stages, with the first stage being the shortest. This is because neural networks can sometimes learn, too quickly, how to fit the given data. The first phase is set to 5,000-10,000 steps. The progressive causal phase requires a longer duration as the model needs to successfully reduce the errors to a certain level before it can extend its learning to new areas. For systems that do not reliably converge, the number of steps for this phase should be sufficiently high. We set this phase to last between 50,000 and 100,000 steps to ensure thorough learning. In the fine-tuning phase, an early stopping approach is implemented. Model performance is evaluated every 1,000 steps using an evaluation loss function where all weights λ are set to 1.0. Training terminates when no performance improvement is observed after several consecutive evaluations. All step counts are selected via a trial-and-error process based on our experimental

outcomes.

6.2.5 Domain Decomposition

Domain decomposition is an effective strategy for training PINNs when the solution exhibits high complexity within specific time intervals or when the target input domain is excessively large, which is frequently encountered in real-world applications. This approach is particularly valuable in extrapolation problems where data in the period of interest is scarce or nonexistent. In such scenarios, the solution's behavior is heavily determined by the model's ability to satisfy the ODE constrains, which is a challenging task due to issues such as causal effect violation [110] or gradient imbalance [42]. By partitioning the domain, the method reduces the domain that the models are trained on and subsequently, reduces the complexity of the optimization task.

In particular, we partition the input domain into S non-overlapping subdomains, denoted as $D_s = [T_{s-1}, T_s], s = 1, \ldots, S$, with $T_1 = 0, T_S = T$. For each subdomain D_s, a neural network U_s is defined. The goal is to train each U_s to approximate the solution u within its respective subdomain D_s. Let $D_s^* = [T_{s-1} - O, T_s + O]$ be the extended subdomain of D_s. Each extended subdomain D_s^* is treated in a separate PINN problem, trained using the framework described in Section 6.2, including the normalization and gradient balancing. Starting from the first subdomain D_0, the model is trained with the initial condition data $\mathcal{D}_u$ provided by users. For subsequent subdomains D_s^*, training data $\mathcal{D}_u$ is generated by the previously trained model U_{s-1} in the overlapped domain $[T_{s-1}, T_{s-1} + O]$. The volume of generated data is potentially unlimited, we set the number from 10 to 100. This iterative process continues until the entire domain is covered. The final solution is obtained by combining the predictions as specified in Equation 6.16, in which each model U_s makes predictions on its corresponding non-overlapped domains.

$$U(t) = \sum_{s} \mathbf{1}_{D_s}(t) \cdot U_s(t), \forall t \in [0, T] \tag{6.16}$$

This divide-and-conquer scheme offers several advantages in training PINNs. Firstly, it reduces the complexity of the overall problem into many smaller less-complex problems. Secondly, it reduces the gradient explosion issue which has been a persistent challenge in PINN training. Additionally, the method allows customization of models in each subdomain, enhancing the individual and overall convergence and accuracy.

6.3 Evaluation Step 1: Ablation Study using the Lorenz System

For the first phase in our validation, we evaluate the effectiveness of the PINN-based solution using the Lorenz system [247], a set of three coupled, nonlinear differential equations widely employed for assessing ODEs.

$$\begin{cases} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z \end{cases} \tag{6.17}$$

This system is known for its chaotic behavior and is described in Equation 6.17, where where x, y, and z denote the state variables at time t, while σ, ρ, and β represent the system's physical parameters. The evaluation uses two temporal domains: a shorter duration with $T = 2.0$ and a longer duration with $T = 40.0$. The shorter duration is used to evaluate the impact of ODE Normalization, Gradient Balancing, and Causal Training on model performance while the longer duration is employed to investigate the effectiveness of Domain Decomposition in managing

extended temporal intervals.

The following set of models were created to provide as detailed a validation as possible.

1. $\text{OdePINN}_{\text{original}}$: Original framework without additional techniques.
2. $\text{OdePINN}_{\text{baseline}}$: ODE Normalization only. This model represents a baseline model for the more complex models below.
3. $\text{OdePINN}_{\text{gradient}}$: ODE Normalization, Gradient Balancing.
4. $\text{OdePINN}_{\text{causal}}$: ODE Normalization, Causal Training.
5. $\text{OdePINN}_{\text{gradient+causal}}$: ODE Normalization, Gradient Balancing, and Causal Training.
6. $\text{OdePINN}_{\text{gradient+causal+domain}}$: ODE Normalization, Gradient Balancing, Causal Training, and Domain Decomposition.

We conduct experiments on both forward and inverse problems using the Lorenz system. The first five models are evaluated with $T = 2.0$, while the sixth model $\text{OdePINN}_{\text{gradient+causal+domain}}$ is assessed on the forward problem over a duration of $T = 20.0$. The generation of training and ground-truth reference data uses a well-established numerical method for solving ODE systems [248], implemented within the Scipy library in Python [231].

6.3.1 Forward Problem with T=2

In this section, we explore how the techniques described in Section 6.2 work together, including ODE Normaliztion, Gradient Balancing, Causal Training and Domain Decomposition. They are particularly tested on their extrapolation capabilities in forward problem scenarios using the Lorenz system. The data loss condition $\mathcal{L}_{data}$ is constrained to a single data point at the initial condition $(x, y, z) = (1, 1, 1)$ at $t = 0$. The physical parameters are maintained constant throughout: $\sigma = 10$, $\rho = 28$, and $\beta = \frac{8}{3}$, all of which are known to the framework.

The framework employs an MLP, U, to approximate the solution u. The architecture of U consists of four hidden layers, each comprising 100 units, with GELU activation functions applied throughout all hidden layers. For Gradient Balancing, the hyperparameters α and N are $(0.99, 100)$ after a hyper-parameter selection process. In the absence of Gradient Balancing, λ_is is set at a constant value of 1.0 throughout training. Three phases of Causal Training are configured as follows: an initial phase of 1,000 steps; a second phase of 199,000 steps; and a final phase incorporating early stopping up to 100,000 steps. When Causal Training is deactivated, the model bypasses the first two phases, proceeding directly to the final phase with training of 300,000 steps without early stopping.

Figure 6.1 illustrates the solution approximations produced by the five models. In order of increasing performance, they are:

- $\text{OdePINN}_{\text{original}}$ exhibits a tendency to converge towards the null solution, primarily due to the dominant influence of ODE loss.

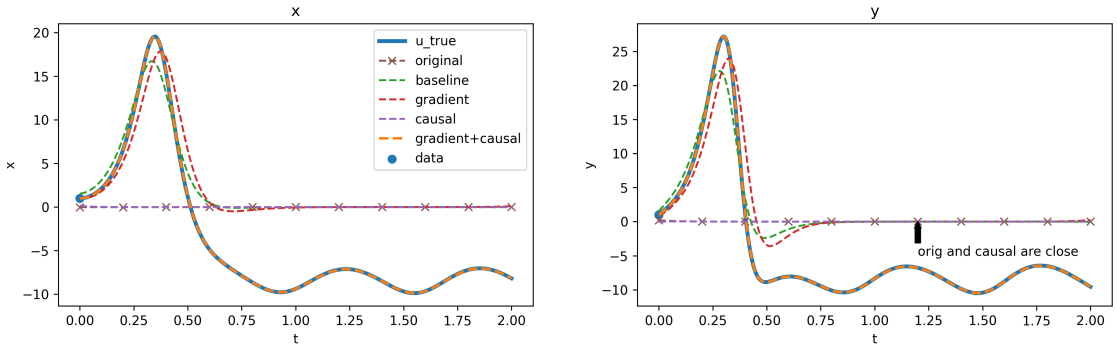
- $\text{OdePINN}_{\text{causal}}$, while effectively capturing the system's dynamics, deviates from the correct initial condition and ultimately converges to the trivial null solution, similar to the *original* model.

- The baseline model successfully captures the general shape of the solution but struggles to simultaneously satisfy both the initial condition and the governing differential equations.

- $\text{OdePINN}_{\text{gradient}}$ achieves improved alignment with the initial condition but fails to adequately satisfy the physics constraints, resulting in an inaccurate approximation of the correct solution.

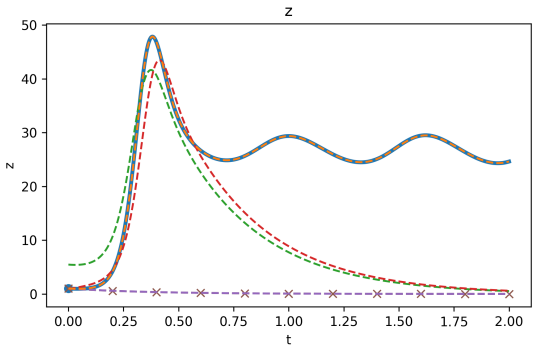
- $\text{OdePINN}_{\text{gradient+causal}}$ demonstrates superior performance, exhibiting close convergence to the true solution compared to the other models.

In summary, $\text{OdePINN}_{\text{gradient+causal}}$ effectively combines both methodologies, ensuring adherence to the initial condition while minimizing ODE loss. This dual



(a) x dimension

(b) y dimension



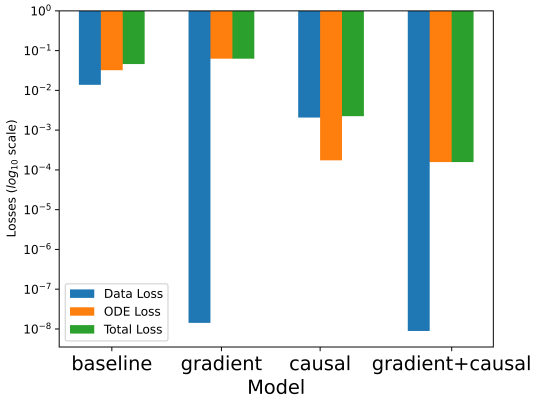
(c) z dimension

Figure 6.1: **Lorenz ODE system for the forward problem, with U approximation of the system state.** System state $u = (x, y, z)$. Subfigures (a), (b), and (c) respectively show the x, y, and z components. There is only data point (blue dot) at $t = 0$ serves as the initial condition. The blue line represents the true solution u_true while the orange dashed line $\text{OdePINN}_{\text{gradient+causal}}$ closely apprximates the target. Both $\text{OdePINN}_{\text{original}}$ (brown dashed line with cross) and $\text{OdePINN}_{\text{causal}}$ (purple dashed line) are closely aligned with the null solution.

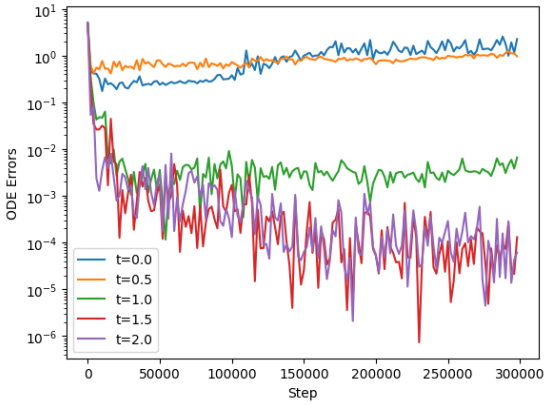
approach enables more accurate approximation of the true solution, as evidenced in Figure 6.2. Evaluation losses, computed across the entire period with weighting coefficients λ_i maintained at 1.0, are presented in Figure 6.2a; $\text{OdePINN}_{\text{original}}$ is not shown as the losses are at different scales but its approximation is similar to $\text{OdePINN}_{\text{causal}}$. The figure shows that the $\text{OdePINN}_{\text{gradient+causal}}$ model achieves a remarkably low final total loss of $1.6 \cdot 10^{-4}$, significantly lower than $\text{OdePINN}_{\text{causal}}$ at $2.2 \cdot 10^{-3}$. The remaining models exhibit higher losses of 0.046 and 0.056, with the baseline model showing marginally better performance. Comparative error analysis reveals that Gradient Balancing substantially enhances convergence towards the

data loss term $\mathcal{L}_{data}$. Data losses for $\text{OdePINN}_{\text{gradient}}$ and $\text{OdePINN}_{\text{gradient+causal}}$ are recorded at $1.4 \cdot 10^{-8}$ and $8.9 \cdot 10^{-9}$ respectively, again significantly lower than the approximately 10^{-2} or 10^{-3} observed in other models. Although $\text{OdePINN}_{\text{gradient}}$ achieves a notably lower data loss, its total loss is $6.3 \cdot 10^{-2}$, marginally higher than the baseline model's $4.5 \cdot 10^{-2}$ due to an increased ODE loss. This technique enables the neural network to more effectively satisfy the initial condition but simultaneously makes it more challenging to minimize the ODE loss, thereby contributing to higher ODE and overall losses. Further details of this method are provided in [42]. Ultimately, the root mean squared error between the gradient model and the true solution is 11.9, slightly better than the baseline model's RMSE of 12.18, though both values remain considerably far from the true solution.

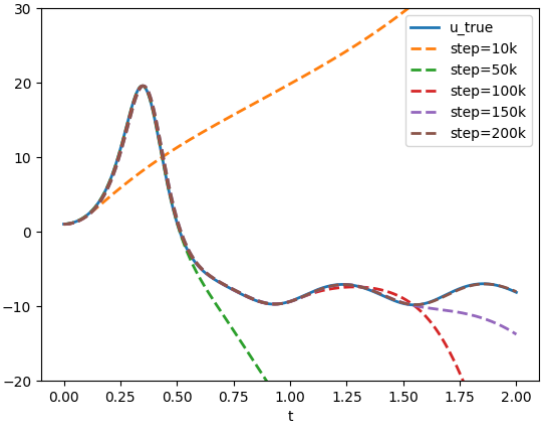
While Gradient Balancing improves performance considerably, it does not address causal effects, necessitating the implementation of Causal Training. The Gradient Balancing technique averages gradients across different time points, resulting in scenarios illustrated in Figure 6.2b. At later times (t=1.5 or t=2.0), the model rapidly minimizes physics constraints to a negligible level of 10^{-5} early in the training process. Within the domain $t \in [1.0, 2.0]$, the model converges to the null solution (as shown in Figure 6.1), satisfying the ODE system. However, at earlier times (t=0 or t=0.5), the model consistently struggles throughout the training duration. We hypothesize that this premature convergence of the ODE constraints at later times traps the model in a local minimum, preventing it from satisfying the constraints at earlier times and thus resulting in an inaccurate solution approximation. To address this, we employ a Causal Training strategy that trains the physics-laws term using residual points drawn from a progressively expanding interval. This approach ensures adherence to system dynamics at earlier times before progressing to later times, thereby respecting causal effects. Figure 6.2c illustrates the expanding domain in which the model complies with the differential equations as training advances, with behavior outside this domain remaining arbitrary. Ultimately, the $\text{OdePINN}_{\text{gradient+causal}}$ model effectively combines these techniques, minimizing both



(a) **Loss Terms across final models selecting through Early Stopping.** The bar is upside down, the lower the better.



(b) **ODE errors of the OdePINN$_{\text{gradient}}$ model at different time t during the training**. The model struggles at $t = [0, 0.5]$, motivating the need for Causal Training.



(c) **The x-value Approximation Solution U of the model OdePINN$_{\text{gradient+causal}}$ at different steps during the training**. The plot shows the progression of the neural network U during training.

Figure 6.2: **Loss analysis of OdePINN framework with Lorenz system, $t \in [0, 2.0]$**

data and physics losses to achieve a highly accurate solution approximation.

6.3.2 Forward Problem with $T = 20$

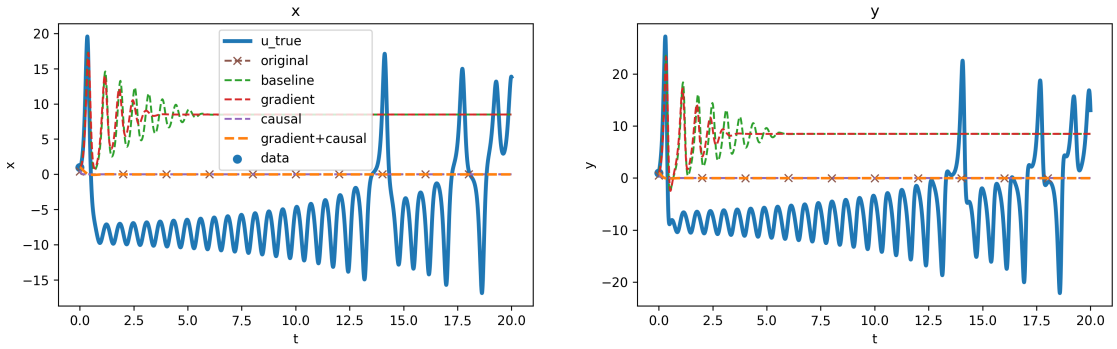
The effectiveness of domain decomposition is demonstrated in this section by solving the Lorenz system over an extended temporal domain from $t = 0$ to $T = 20.0$. The initial condition is set at $(1, 1, 1)$, with constant, known parameters $\sigma = 10$, $\rho = 28$,

and $\beta = \frac{8}{3}$.

Figure 6.3 presents the results the larger domain. None of the models closely approximate the reference solution. The models $\text{OdePINN}_{\text{baseline}}$ and $\text{OdePINN}_{\text{gradient}}$ exhibit some capability to learn the initial condition and partially adhere to the ODE equations. However, their errors grow substantially by $t = 0.75$, causing their divergence from the true solution and leading to convergence toward a trivial constant solution beyond $t = 3$. The remaining models,$\text{OdePINN}_{\text{original}}$, $\text{OdePINN}_{\text{causal}}$ and $\text{OdePINN}_{\text{gradient + causal}}$ display a similar behavior to that observed in the $T = 2$ experiment, converging toward a null solution. These inaccuracies can be attributed to the expanded domain, which increases the complexity of the solution. Therefore, Domain Decomposition is applied to manage the challenges introduced by the larger domain size.

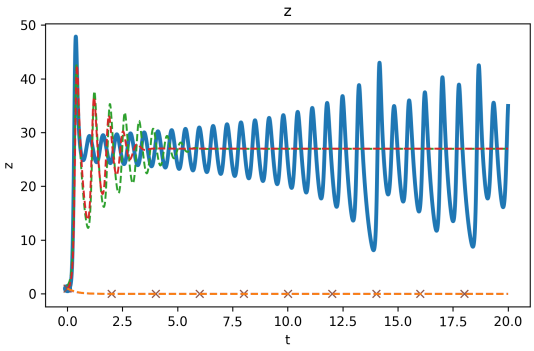
The domain is partitioned into 40 subdomains, each spanning 0.6 units with an overlap of 0.05 at both ends. Each subdomain is independently modeled and trained using a distinct neural network. The predictions from the preceding subdomain, uniformly distributed over 100 points within the overlapped region, serve as data conditions for the subsequent subdomain. Hyper-parameters remain consistent across all subdomains. The neural network U is structured as an MLP with four hidden layers, each comprising 100 units and utilizing the GELU activation function. The training process is limited to a maximum of 150,000 steps, initiating with a 5,000-step phase dedicated to data fitting, followed by 100,000 steps of causal training, with the remaining steps allocated for final tuning. The gradient balancing weights, λ_i, are updated every 100 steps using a smoothing factor of $\alpha = 0.99$.

Figure 6.4 presents the solution approximated by the proposed framework, alongside a plot of training losses and Root Mean Squared Error (RMSE) relative to ground truth data. The model demonstrates reasonably accurate prediction of the system's evolution. Initially, the RMSE is approximately $5 \cdot 10^{-4}$, but it increases exponentially as t progresses, escalating to 10^{-3} at $t = 8$, 10^{-1} at $t = 13$, and reaching an error magnitude of 10^{1} by the period's end. The final four subdomains (12.5



(a) x dimension

(b) y dimension



(c) z dimension

Figure 6.3: **Lorenz ODE system, forward problem, U approximation of the system state, using the first five models** (excluding the Domain Decomposition model). System state $u = (x, y, z)$. Subfigures (a), (b), and (c) respectively show the x, y, and z components. There is only data point (blue dot) at $t = 0$ serves as the initial condition. None of the models successfully capture the dynamics of the reference solution.

to 20.0) exhibit notable approximation errors. This substantial error accumulation towards the period's end is understandable given that the Lorenz system is highly sensitive to initial conditions where minor predictive inaccuracies can significantly alter future states. As a result, initial training errors rapidly accumulate over time, resulting in an RMSE of up to 10 by the end of the time period. The training data loss consistently remains below 10^{-4}, frequently dropping to the 10^{-5} level. Errors related to physical constraints are maintained at the 10^{-3} level, indicating satisfaction of physical laws with ODE loss approaching zero. Notably, losses peak in solution regions characterized by sharp changes, correspondingly resulting in steep RMSE increases. On a positive front, the framework exhibits consistent performance across

all subdomains However, the evident accumulation of errors highlights a potential limitation of the technique, suggesting areas for further refinement and improvement.

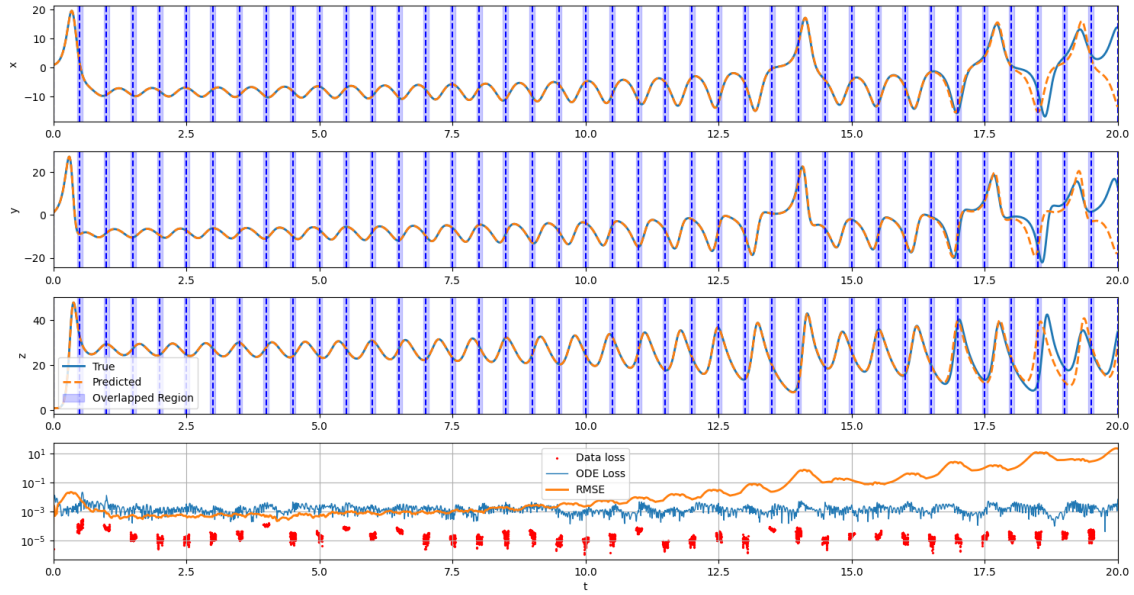


Figure 6.4: **Lorenz ODE system, forward problem, with U approximating the system state.** System state $u = (x, y, z)$. The first three plots depict the x, y, and z components, respectively, while the last plot shows the loss and RMSE over the input time domain. The PINN approximation (orange dashed line) closely follows the reference solution (blue line) up to $t = 17$.

Figure 6.5 illustrates the relationship between the number of training steps and the RMSE across various subdomain configurations. The number of training steps scales linearly with the increase in subdomains, with each subdomain requiring approximately 200,000 to 300,000 steps—equivalent to around 30 minutes on the NVIDIA GeForce RTX 4090 GPU used for all experiments. The figure reveals a trade-off between the number of subdomains and model accuracy: as the number of subdomains increases, the RMSE decreases exponentially, though this comes at the expense of longer training durations. When the number of subdomains are low, such as 1 or 5, the training domain remains overly large, leading the model to converge towards the null solution and yielding a high RMSE of 13.4. As the number of subdomains increases, starting from 10, each subdomain's training area is reduced, simplifying the solution complexity within each subdomain. This reduction leads to a decrease in RMSE, dropping to 4.1 with 10 subdomains and reaching 1.13 with 40 subdomains. These findings suggest that cumulative errors can be effectively

minimized by subdividing the domain further, enabling the model to achieve finer accuracy levels within each subdomain.

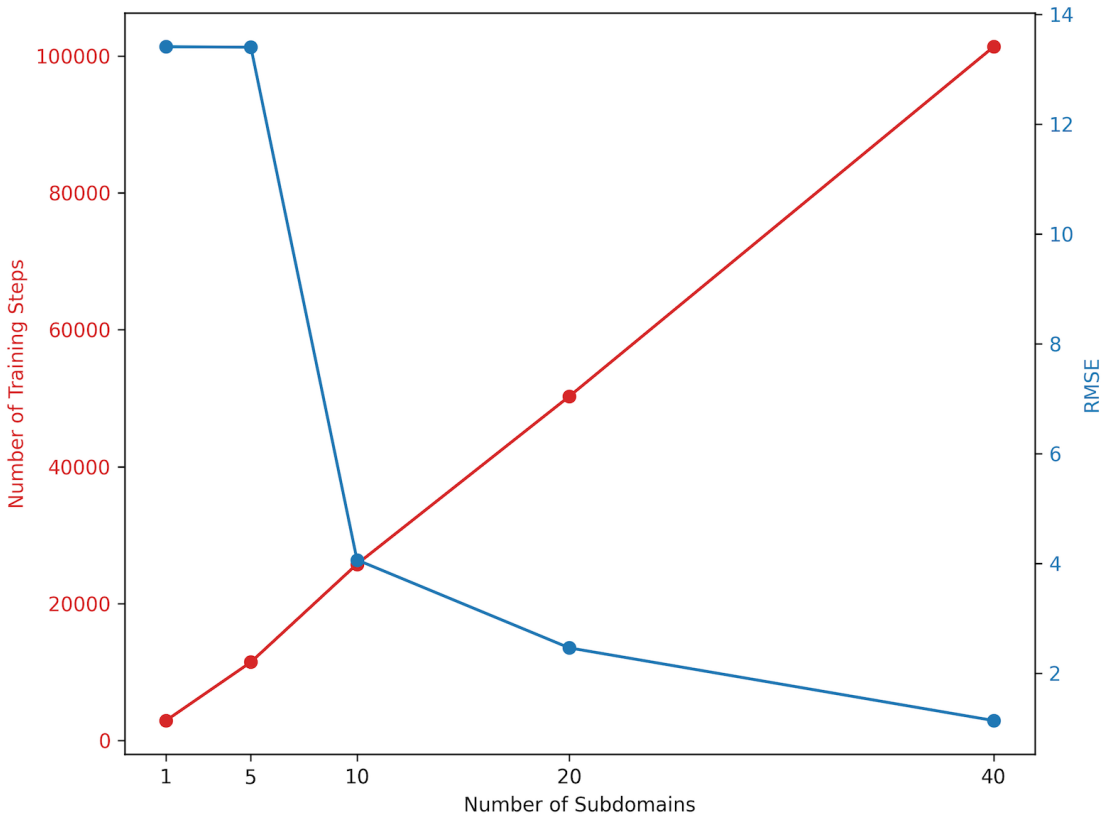


Figure 6.5: Trade-off between the number of subdomains and accuracy.

6.3.3 Inverse Problem

In this section, the inverse problem is addressed by using a substantially larger dataset. The framework solution simultaneously predicts physics parameters and interpolates the system state from several observations of the state.

Once again, experiments use the Lorenz system, on this occasion over the time domain $t \in [0, 2.0]$, with time-varying physical parameters defined as $\sigma = \frac{10}{2} \sin(2\pi t) + 10$, $\rho = \frac{28}{5} \sin\left(2\pi t + \frac{\pi}{2}\right) + 28$, and $\beta = \frac{8}{3}$. These formulas are unknown to the framework and must be learned. Initial conditions are set to $(1, 1, 1)$, consistent with previous experiments. The training dataset comprises 21 simulated data points evenly distributed across the input domain, resulting in a dataset dimension of $(21, 3)$. Figure 6.6 illustrates the reference solution achieved from the simulation described

and the 21 data points provided to the PINN system. This solution, with varying parameters in this experiment, displays a notably more chaotic behavior compared to the solution in the forward problem setting in Section 6.3.1. However, the data points available provide general information about the solution for the framework to solve the problem.

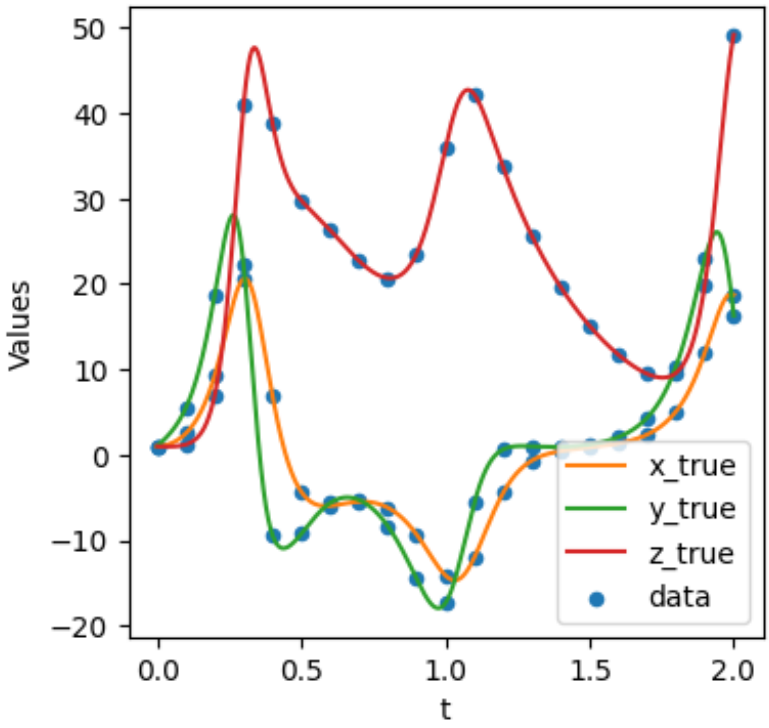


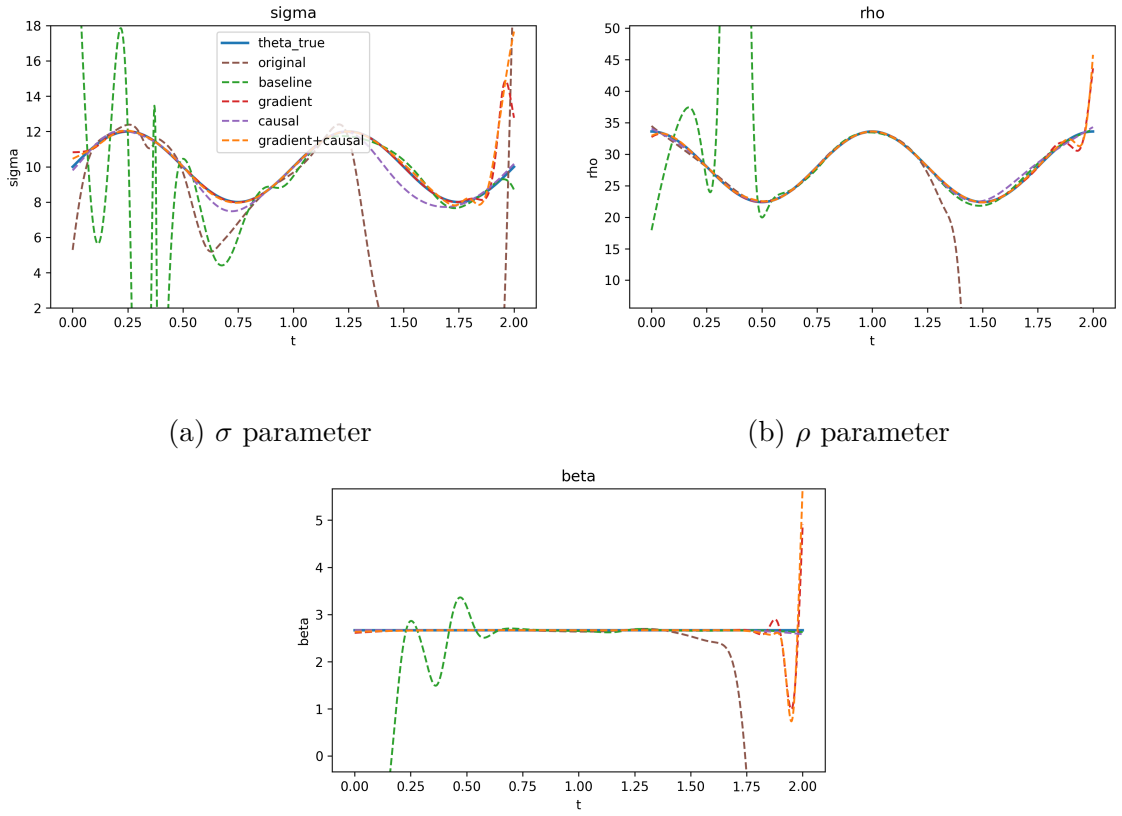
Figure 6.6: Ground truth u and data provided to solve the Lorenz system inverse problem.

The framework configuration is consistent with previously described experiments. The main neural network U maintains its structure, comprising an MLP with four hidden layers, each containing 100 units and using GELU activation. Three additional, distinct neural networks Θ are implemented to model the three physical parameters of the dynamical system. These networks each consist of four hidden layers with 10 units, employing GELU activation for all hidden layers. All networks incorporate the time variable t as input to capture the temporal dynamics of the parameters. In terms of training enhancements, Gradient Balancing is used with the weights of the objective function, λ_i, adjusted every $N = 100$ steps using a smoothing factor of $\alpha = 0.99$. The Causal Training approach involves an initial data fitting phase

of 10,000 steps, reflecting the increased data availability, followed by progressive causal training and a final tuning phase, conducted over 200,000 and 100,000 steps, respectively.

Figure 6.7 illustrates the approximation of the dynamical system's parameters by the neural network Θ. While all models demonstrate the ability to capture general parameter trends, they exhibit inherent inaccuracies. The original and baseline models show substantial errors, particularly in regions of high derivatives of u with respect to time t, as observed within the interval $[0, 0.5]$ and at $t = 2.0$ in Figure 6.6. Among the models, $\text{OdePINN}_{\text{gradient}}$ achieves the most accurate parameter approximations across most of the domain, but encounters difficulties near $t = 2.0$ In this inverse setting, with variable unknown system parameters, non-unique solutions for u and θ arise in the absence of sufficient data. Consequently, $\text{OdePINN}_{\text{gradient}}$ tends to favor solutions with smaller magnitude derivatives, resulting in significant discrepancies from the reference solution. Conversely, $\text{OdePINN}_{\text{causal}}$ typically exhibits error accumulation towards higher values of t, with noticeable inaccuracies in estimating parameters such as σ and ρ beyond $t = 1.15$, as shown in Figure 6.7. This pattern of error propagation is consistently observed throughout our study. The combined model, $\text{OdePINN}_{\text{gradient+causal}}$, mirrors the performance of $\text{OdePINN}_{\text{gradient}}$, maintaining reasonable accuracy until the domain's far end.

Table 6.1 presents further detail of the errors associated with the interpolation of u and the estimation of parameters θ, benchmarked against the ground truth data. Table rows are organized by error metrics which are calculated as described in Appendix A.1. Each group shows the corresponding errors for the approximated solutions (columns x, y, z and the average u) and for the learned parameters (columns α, β, γ and the average θ) across the 5 models. The *original* model exhibits the highest errors, with RMSE values of 3.9564 for u and 52.4141 for θ, alongside MDAPE of 2.6219 and 24.5114, respectively. The baseline model shows marginal improvements over the *original* model but still incurs relatively high errors. In contrast, $\text{OdePINN}_{\text{gradient}}$ demonstrates significant accuracy improvements, recording



(a) σ parameter

(b) ρ parameter

(c) β parameter

Figure 6.7: **Θ approximation of the physics parameters $\theta = (\sigma, \rho, \beta)$ in the Lorenz system inverse problem.** Subfigures (a), (b), and (c) respectively show the parameters σ, ρ, and β. The models $\text{OdePINN}_{\text{gradient}}$ (red dashed line) and $\text{OdePINN}_{\text{gradient+causal}}$ (orange dashed line) provide reasonably accurate estimates, closely following the true solution (blue line).

the lowest errors across all evaluated metrics for both u and θ. It achieves an RMSE of 0.2525 for u and 0.7195 for θ, with MAE values of 0.0472 and 0.1602, and MDAPE scores below 0.01% for both variables. The causal model records the best RMSE for u at 0.1039, although its performance on other metrics does not surpass that of the gradient model. $\text{OdePINN}_{\text{gradient+causal}}$, which integrates the approaches of the preceding two models, does not achieve top scores in any specific category but secures the second-best results, including impressive MDAPE values below 0.01% and 0.06%.

The final training stage of $\text{OdePINN}_{\text{gradient+causal}}$ replicates the entire training process of the model. The initial two phases of causal training contribute minimally to overall performance improvement. This is a characteristic that is mainly attributed

Table 6.1: Approximation errors in the inverse problem with Lorenz system where smallest error values are best. **Bold text** highlight the best performing models with respect to a specific metric and Underlined numbers represent the second best performing model.

Model	Metric	x	y	z	u	σ	ρ	β	θ
original		4.4453	4.9325	1.6939	3.9564	5.4294	90.5381	3.8850	52.4141
baseline		3.4995	2.8182	0.6591	2.6219	8.4507	41.5828	1.3683	24.5114
gradient	RMSE	0.1011	0.3556	0.2336	0.2525	0.8727	0.8498	0.2629	**0.7195**
causal		0.1363	0.0949	0.0692	**0.1039**	1.3853	2.4592	0.0740	1.6302
gradient+causal		0.0862	0.3760	0.2659	0.2705	1.2593	0.8922	0.3316	0.9114
original		1.7002	1.0028	0.3667	1.0232	3.6059	47.0012	1.4000	17.3357
baseline		0.9579	0.6687	0.2430	0.6232	3.1751	10.9855	0.5066	4.8891
gradient	MAE	0.0207	0.0759	0.0449	**0.0472**	0.2116	0.2078	0.0611	**0.1602**
causal		0.0844	0.0654	0.0452	0.0650	0.7099	1.3002	0.0472	0.6858
gradient+causal		0.0192	0.0904	0.0543	0.0546	0.2845	0.2590	0.0827	0.2087
original		0.0063	0.0064	0.0015	0.0025	0.1271	0.0127	0.0058	0.0200
baseline		0.0019	0.0029	0.0016	0.0019	0.0382	0.0114	0.0086	0.0162
gradient	MDAPE	0.0001	0.0002	0.0001	**0.0001**	0.0005	0.0005	0.0003	**0.0004**
causal		0.0004	0.0012	0.0016	0.0013	0.0175	0.0044	0.0074	0.0090
gradient+causal		0.0000	0.0001	0.0001	**0.0001**	0.0006	0.0003	0.0007	0.0006
original		0.2562	0.2450	0.0703	0.2086	2.7147	16.1675	3.8850	9.7271
baseline		0.2017	0.1400	0.0274	0.1426	4.2253	7.4255	1.3683	4.9955
gradient	nRMSE	0.0058	0.0177	0.0097	0.0121	0.4364	0.1517	0.2629	0.3069
causal		0.0021	0.0025	0.0008	**0.0019**	0.2210	0.0608	0.0156	**0.1326**
gradient+causal		0.0079	0.0162	0.0140	0.0132	0.5519	0.1572	0.3124	0.3772

to the ability of a sufficiently large dataset to establish a robust initialization, thereby reducing the relative advantage of the first two phases. On the other hand, $\text{OdePINN}_{\text{gradient+causal}}$, with its shorter training duration compared to others, does not achieve performance levels comparable to those of the Gradient-Balancing model.

Summary. These ablation experiments indicate that the proposed method individually enhance the performance of PINNs. ODE Normalization outperforms the *original* framework in both forward and inverse settings. Integrating Gradient Balancing and Causal Training in the forward problem with limited data addresses issues of gradient imbalance and temporal causality, resulting in more accurate approximations. However, in the inverse problem with more data, Gradient Balancing alone is enough to achieve good parameter estimation and state interpolation, as the abundance of data diminishes the advantage provided by Causal Training. This suggests that the role of Causal Training is more pronounced in scenarios with sparse data. In the larger domain setting ($T = 20$), domain decomposition proved essential for managing errors in solving the Lorenz system. There is a trade-off between accuracy and training time with a greater number of subdomains yielding better results but requiring more computational resources.

6.4 Validation Step 2: Mosquito Case Study

In addition to using the Lorenz system, our PINN framework is also validated in the practical environment of dynamical modeling of the mosquito population. We apply our approach to the ODE-based model of mosquito population dynamics proposed by [128]. The model divides the mosquito life cycle into 10 stages: Egg (E), Larva (L), Pupa (P), Emerging Adults (A_{em}), Nulliparous Bloodseeking Adults (A_{b1}), Nulliparous Gestating Adults (A_{g1}), Nulliparous Ovipositing Adults (A_{o1}), Parous Bloodseeking Adults (A_{b2}), Parous Gestating Adults (A_{g2}) and Parous Ovipositing Adults (A_{o2}). These stages are interconnected through a system of ordinary differential equations, as presented in Equation 6.18.

Parameters for the experiment are explained in Table 6.2. These system param-

eters vary with temperature and are based on data specific to the *Culex pipiens* spieces [44]. This ODE system serves as a useful validation, as it is both practically relevant and exemplifies multi-scale behavior across the mosquito life stages, with a relatively large input domain. These characteristics present significant challenges for the original PINN framework, which struggles to achieve convergence for this system.

$$\begin{cases} \frac{dE}{dt} & = \gamma_{Ao}(\beta_1 A_{o1} + \beta_2 A_{o2}) - (\mu_E + f_E)E \\ \frac{dL}{dt} & = f_E E - \left(m_L\left(1 + \frac{L}{\kappa_L}\right) + f_L\right)L \\ \frac{dP}{dt} & = f_L L - (m_P + f_P)P \\ \frac{dA_{em}}{dt} & = f_P \sigma e^{-\mu_{em}\left(1+\frac{1}{\kappa_P}\right)} P - (m_A + \gamma_{Aem})A_{em} \\ \frac{dA_{b1}}{dt} & = \gamma_{em} A_{em} - (m_A + \mu_r + \gamma_{Ab})A_{b1} \\ \frac{dA_{g1}}{dt} & = \gamma_{Ab} A_{b1} - (m_A + f_{Ag})A_{g1} \\ \frac{dA_{o1}}{dt} & = f_{Ag} A_{g1} - (m_A + \mu_r + \gamma_{Ao})A_{o1} \\ \frac{dA_{b2}}{dt} & = \gamma_{Ao}(A_{o1} + A_{o2}) - (m_A + \mu_r + \gamma_{Ab})A_{b2} \\ \frac{dA_{g2}}{dt} & = \gamma_{Ab} A_{b2} - (m_A + f_{Ag})A_{g2} \\ \frac{dA_{o2}}{dt} & = f_{Ag} A_{g2} - (m_A + \mu_r + \gamma_{Ao})A_{o2} \end{cases} \tag{6.18}$$

This part of the evaluation comprises two problems: first is a forward problem solving the mosquito population dynamics given an initial condition; and second is an inverse problem determining mortality and growth rates from available data. Experiments are conducted under varying temperature conditions, where the temperature changes according to a sine function $\tau = 10\sin\left(2\pi\frac{t}{365}\right) + 10$ with time t measured in days.

6.4.1 Forward Problem

For the forward problem, the objective is to solve the mosquito population dynamics using only a single data point: the *initial condition*. We employ a numerical method [248] for simulating the ODEs, available in Python Scipy library [231] The data point at $t = 730$ (two years into the simulation) is selected as the initial condition

Table 6.2: ODE Model Parameters. The unit of τ is Celsius degree. All other parameters have the unit of day^{-1}, except the σ and β.

Parameter	Description	Value
τ	Temperature	
γ_{Aem}	Development rate of emerging adults	1.143
γ_{Ab}	Development rate of bloodseeking adults	0.885
γ_{Ao}	Ovipositing adult development rate	2
$f_E(>0)$	Egg development rate	$0.16 \cdot \left(e^{[0.105(\tau-10)]} - e^{\left[0.105(38-10)-\frac{1}{5.007}(38-\tau)\right]}\right)$
f_P	Pupa development rate	$0.021 \cdot \left(e^{[0.162(\tau-10)]} - e^{\left[0.162(38-10)-\frac{1}{5.007}(38-\tau)\right]}\right)$
f_L	Larva development rate	$f_P * 1.65$
$f_{Ag}(>0)$	Development rate of gestating adults	$\frac{\tau-9.8}{64.4}$
m_E	Egg mortality rate	$m_E = \mu_E$
m_L	Larval mortality rate	$\exp[-\tau/2] + \mu_L$
m_P	Pupa mortality rate	$\exp[-\tau/2] + \mu_P$
$m_A(>\mu_A)$	Mortality rate of Ab,	$-0.005941 + 0.002965 \cdot \tau$
μ_E	Minimum egg mortality rate	0
μ_L	Minimum larval mortality rate	0.0304
μ_P	Minimum pupa mortality rate	0.0146
μ_{em}	Mortality rate during emergence	0.1
μ_r	Mortality rate during bloodseeking	0.08
μ_A	Minimum adult mortality rate	$\frac{1}{43}$
κ_L	Carrying capacity for larvae	$8 \cdot 10^8$
κ_P	Carrying capacity for pupae	10^7
σ	Sex ratio at emergence	0.5
β	Number of eggs per Ao	$\beta_1 = 141(np), \beta_2 = 80(p)$

for training a PINN over the interval $[730, 1096]$ days. We use the data point at $t = 730$ (two years into the simulation) as the starting point and train a PINN for the period from 730 to $1,096$ days. The simulated data within this period serve as the ground truth for model performance evaluation. The advanced techniques presented in the Section 6.2, including ODE normalization, domain decomposition, gradient balancing and causal training, are implemented as part of this evaluation.

The time domain is divided into 12 subdomains where for each, they undergo a three-phase training process: 10,000 steps of data fitting followed by 100,000 steps of causal training; the final phase uses early stopping, stops after 100 evaluations without improvement. Lower and upper bounds for each domain are derived from ground truth data. The solution neural network U is configured as an MLP with 4

Table 6.3: Errors for Mosquito ODE Approximation Solution. The **bold text** highlights the two lowest errors across the stages, representing the best approximations, while the underline text identifies the two highest errors, indicating the least accurate stages.

Stage	RMSE	MAE	MDAPE	nRMSE
E	56107.947877	28619.218663	0.001900	0.000407
L	144952.348476	82367.593703	**0.000759**	0.000308
P	243295.690709	123936.043785	0.002170	0.001325
Aem	398.308537	175.715612	0.001821	0.001106
Ab1	342.155798	161.987210	0.001782	0.000850
Ag1	42095.722155	20403.707941	0.002158	0.019350
Ao1	**34.564301**	**16.196719**	0.003921	0.000293
Ab2	102.456176	50.194130	0.002831	**0.000217**
Ag2	7853.480625	4466.414135	**0.001567**	0.004155
Ao2	**19.229490**	**9.978838**	0.002394	**0.000138**
Overall	92296.312508	26020.705074	0.001918	0.006291

hidden layers, each comprising 100 units and utilizing GELU activation functions. The subdomains are trained sequentially, with predictions from the overlapping region of preceding subdomain models serving as initial conditions for subsequent domains.

Figure 6.8 presents the results of solving the mosquito population dynamics as a forward problem, with corresponding error metrics detailed in Table 6.3. The trained neural network demonstrates remarkable performance in extrapolation, providing a reasonably accurate approximation of the solution, as illustrated in Figure 6.8. However, towards the end of the time period, notable accumulated error can be observed in A_{g1} and A_{g2} stages.

Table 6.3 reveals a substantial overall RMSE of 92,296. Error magnitudes exhibit significant variation across mosquito life stages, ranging from 19 in the $Ao2$ stage to 243,295 in the P stage, likely attributable to scale differences among these stages. MAE follows a similar pattern, spanning from 10 ($Ao2$ stage) to 123,936 (P stage), with a mean of 26,021. Despite these high RMSE and MAE values, the Median Absolute Percentage Error (MDAPE) remains relatively low, averaging 0.19%, indicating generally robust performance from a machine learning perspective. MDAPE tends to be lower for stages with larger scales, suggesting that stages with

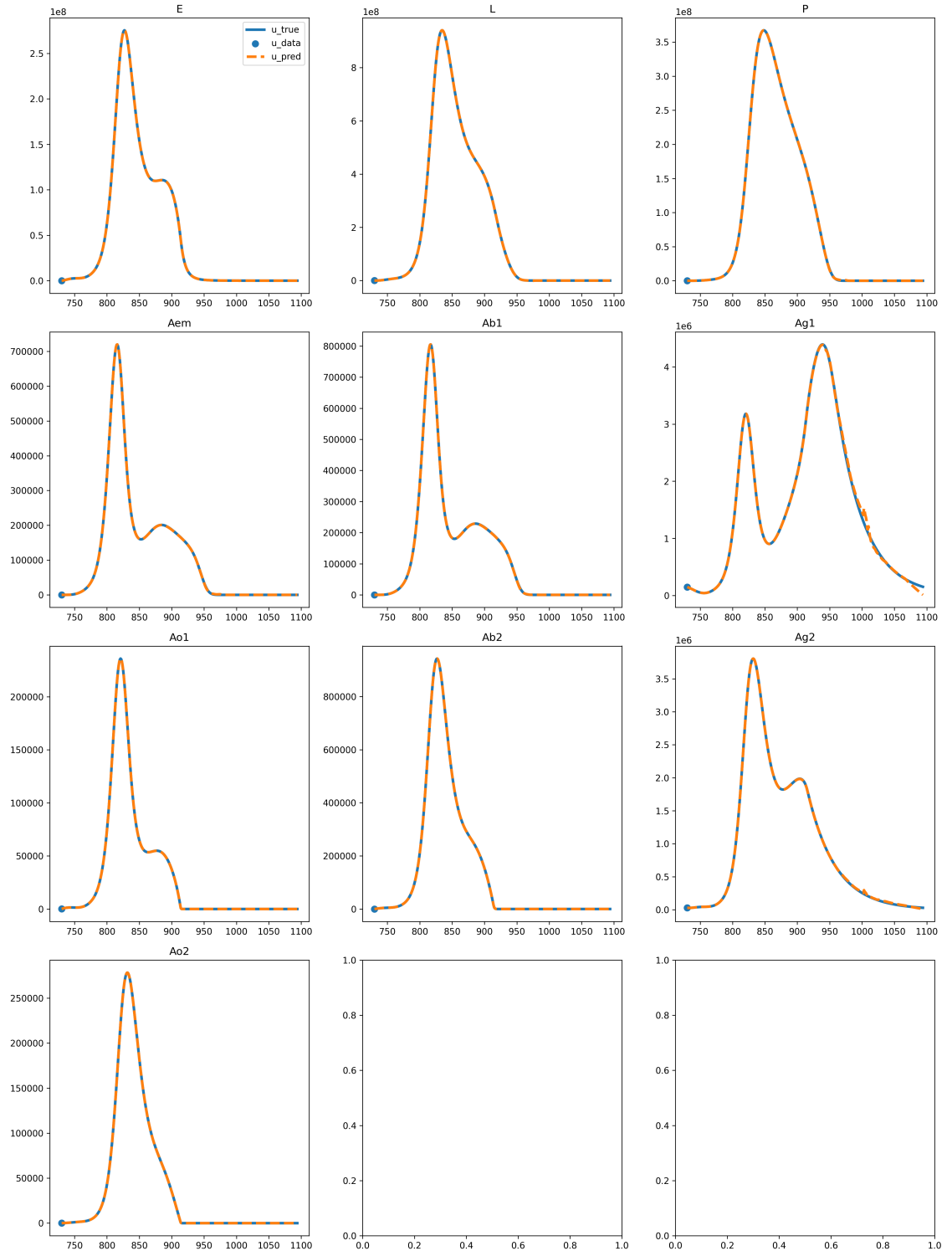


Figure 6.8: **Mosquito ODE system, forward problem, U approximation of the solution.** Each plot depicts the evolution of a specific state in the mosquito life cycle, with time on the x-axis and organism count on the y-axis. Only one data point (blue dot) is provided to PINN at $t = 730$. The PINN predictions (orange dashed line) accurately track the reference solution (blue line) over time.

larger scales are more tolerant of minor errors. For instance, the L stage exhibits the lowest MDAPE (0.0759%) despite having the second-highest RMSE and MAE. Conversely, the $Ao1$ stage presents the highest MDAPE (0.3921%) while maintaining relatively lower RMSE and MAE values.

6.4.2 Inverse Problem

In solving inverse problems, the goal is to use available data to estimate 10 of parameters identified in Table 6.2. Three parameters (γ_{Aem}, γ_{Ab} and γ_{Ao}) are treated as constants and are represented by learnable parameters that remain constant over time. The seven parameters (f_E, f_P, f_L, f_{Ag}, m_L, m_P and m_A) vary over time with neural networks using time as input to estimate these parameters.

This experiment is conducted under identical conditions and temperatures as previous evaluation studies. Simulation data is generated over a three-year period, spanning from day 0 to day 1096, employing the numerical method described in [248]. For data conditioning, daily observations from $t = 730$ to $t = 1096$ are utilized, encompassing 367 days with 10 data points each. The solution u and the ODE parameters θ are obtained from the simulation data as ground truth.

With this setup, the high data volumes allow for the use of simpler techniques. Due to significant differences in the number of instances across various stages, ODE normalization and gradient balancing are implemented. Domain decomposition and causal training are not employed (the number of subdomains is set to 1, and the count for the first two phases is set to 0), as the data condition with sufficient observations provides a robust foundation for framework convergence. Boundaries for the system states and ODE parameters are determined based on simulation data. For the neural network solution U, a MLP architecture is deployed, comprising 4 hidden layers with 100 units each and employing the GELU activation function. Constant parameters are represented by individual trainable weights, while time-dependent parameters are each modeled using a smaller MLP structure. These smaller networks consist of four hidden layers with 10 units per layer, also using the GELU activation function.

Table 6.4: Mosquito ODE System's Parameter Approximation Errors. The **bold text** highlights the two lowest errors across the learned parameters while the underline text identifies the two highest errors.

Parameter	RMSE	MAE	MDAPE
γ_{Aem}	0.042639	0.042639	0.037304
γ_{Ab}	0.034216	0.034216	0.038662
γ_{Ao}	0.072417	0.072417	**0.036208**
f_E	0.015570	0.012633	0.039964
f_P	0.004646	0.003270	0.078140
f_L	**0.002816**	0.001982	0.078126
f_{Ag}	**0.002769**	**0.001784**	0.094168
m_L	0.286959	0.137099	0.125888
m_P	0.291548	0.139703	0.259069
m_A	0.002848	**0.001828**	**0.033566**
Mean	0.132615	0.044757	0.059045

Figure 6.9 illustrates the system parameters learned through the PINN framework. The plots generally demonstrate accurate parameter approximations, although notable inaccuracies are observed, particularly for parameters m_L and m_P within the domain $[943, 1064]$. Error metrics, as detailed in Table 6.4, where the mean RMSE is 0.132615. Parameters m_L and m_P contribute the highest errors at 0.286959 and 0.291548, respectively, while f_{Ag} and f_L present the lowest RMSE values at 0.002769 and 0.002816, respectively. The MAE closely aligns with RMSE trends, averaging 0.044757. The highest MAE values correspond to m_L (0.137099) and m_P (0.139703), while the lowest are associated with f_{Ag} (0.001982) and f_L (0.001784). he MDAPE averages 5.9%, with m_A demonstrating the lowest error at 3.3566%. The three constants (γ_{Aem}, γ_{Ab}, and γ_{Ao}) range between 3.62% and 3.87%. The errors for m_L and m_P are notably higher at 12.59% and 25.91%, respectively.

The results can be interpreted as follows: during colder periods, specifically within the day range of 943 to 1,064 where air temperature falls below $5.0°C$, the mosquito population rapidly declines to zero (with the exception of $Ag1$ and $Ag2$), as illustrated in Figure 6.8. This situation makes it challenging to gather useful information, negatively affecting the ability to determine several system parameters, especially m_L and m_P which are primarily derived from equations related to $\frac{dL}{dt}$

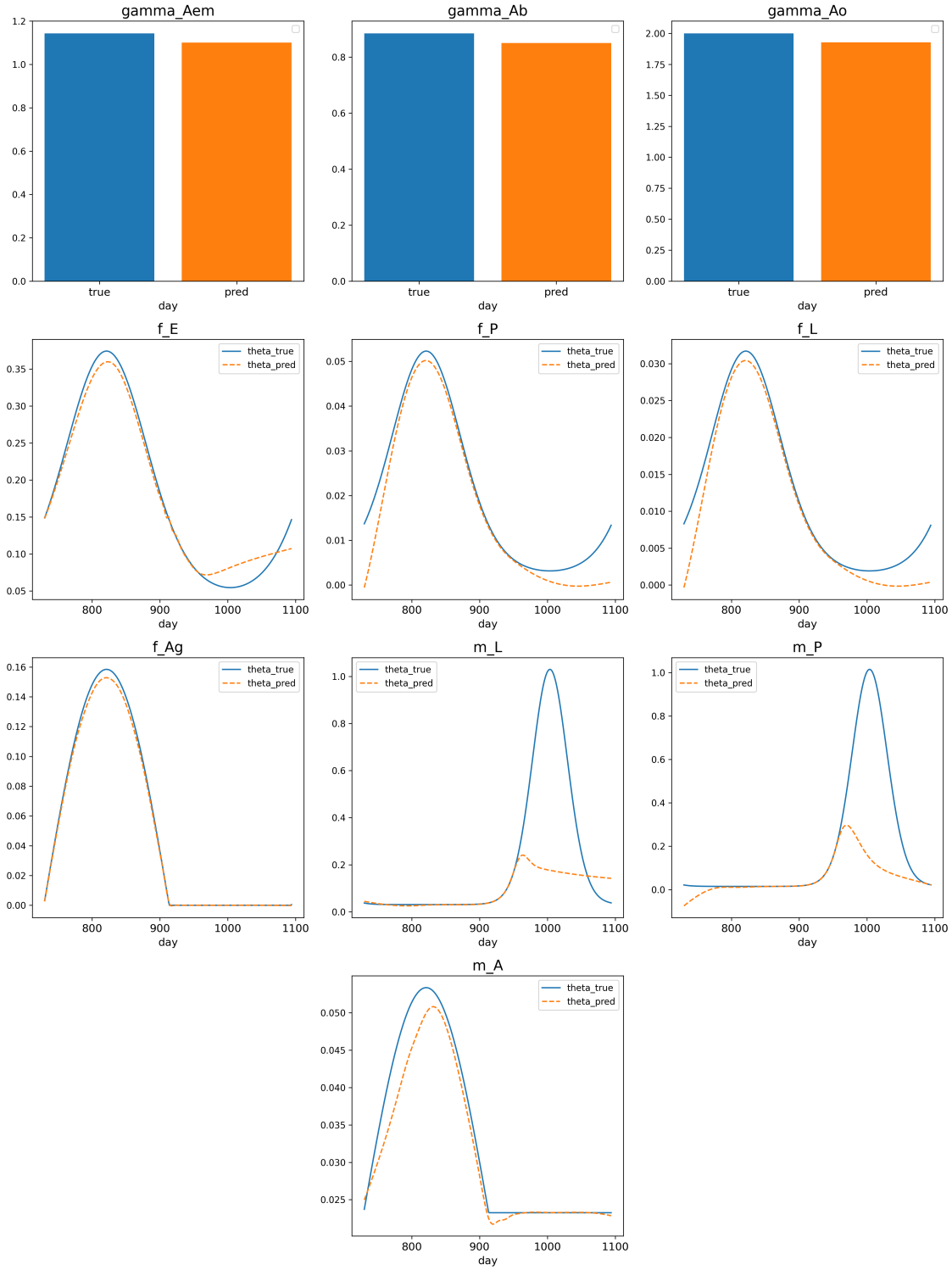


Figure 6.9: **Mosquito ODE system, inverse problem, Θ approximation of the system's parameters.** Each plot compares the approximated values (orange) with the true values (blue) for different parameters governing the mosquito population dynamics.

and $\frac{dP}{dt}$. Conversely, parameters γ_{Aem}, γ_{Ab} and γ_{Ao} remain constant over time, which makes them simpler to accurately estimate, as evidenced by their relatively

low MDAPE. Meanwhile, parameters such as m_A and f_{Ag}, which are involved in multiple differential equations, tend to receive more consistent and stable information and gradients. From a mathematical perspective, the target parameters are not structurally identifiable across the entire input domain due to the system configuration (see Appendix C.1). This lack of structural identifiability can lead to inaccuracies in the approximations generated by PINNs.

Summary. The PINN framework developed for this research demonstrated a quite accurate approximation of the mosquito dynamical system, despite the challenges posed by extreme multi-scale behaviors and a large input domain. Our approach successfully normalizes the system states and parameters onto the same scale with an even optimization. The results show strong accuracy, particularly in high-temperature domains where information is redundant. However, due to inherent structural identifiability limitations, the model fails in terms of accurately approximating parameters within the low-temperature domain.

6.5 Conclusion

In this chapter, we introduced a hybrid framework based on Physics informed Neural Networks that integrates physical laws into data-driven machine learning models. This framework is specifically designed to solve systems of ordinary differential equations and was validated using both the Lorenz system and a case study modeling mosquito population dynamics. The approach comprises a multi-task learning strategy, incorporating multiple components in the objective function: one for data fitting and several for weakly enforcing physical constraints. The framework also includes several advanced techniques, including domain decomposition, ODE normalization, gradient balancing, and causal training. To evaluate the effectiveness of this approach, we conducted an ablation study using the Lorenz system before addressing the complex problem of modeling mosquito population dynamics.

Our findings demonstrate that ODE Normalization and Gradient Balancing techniques played a crucial role in stabilizing the training process. These meth-

ods effectively prevent individual components of the loss function from exerting disproportionate influence on the optimization and thus, mitigate against premature convergence to suboptimal solutions. Causal Training preserves the temporal causality inherent in the dynamical system. This aspect is crucial for achieving accurate model predictions, especially in scenarios requiring extrapolation beyond the scope of the training data. Furthermore, Domain Decomposition demonstrates its effectiveness in managing significantly large input domains, particularly in forward problem scenarios. The results also confirm the framework's efficacy in modeling mosquito population dynamics, highlighting its potential for application within the field of ecology.

However, the PINN framework presented in this chapter still reveals certain limitations. In the inverse problem setup, the PINN tends to favor solutions with smaller gradients that still met the data and ODE constraints. The current implementation considers only time as an input, neglecting external factors. This restriction significantly limits the model's predictive performance in real-world applications, as the coordination inputs and encoded physical laws may not fully capture the underlying mechanisms of the processes. Allowing external variables would not only enhance performance but also increase the framework's flexibility. While the framework has demonstrated promising results on test systems, its compatibility with real-world data remains to be tested. In Chapter 7, we will address two of these limitations by allowing external variables as inputs and testing the method against real-world mosquito count data, thereby advancing the practical applicability of our approach.

Chapter 7

PINN Optimization: Incorporating External Factors

In Chapter 6, we developed a methodology to incorporate ordinary differential equations (ODEs) into a machine learning framework, one which incorporated solutions for both forward and inverse problems in dynamical systems. However, the framework's inability to consider external variables may limit its applicability in certain scenarios. Furthermore, the framework's evaluation was limited to simulated data in the chapter, potentially under-representing its true capabilities when applied to real-world observations. This chapter aims to address these limitations through an inverse problem case study using the same mosquito population case study presented in Chapter 6.

7.1 Introduction

We begin with an overview of our approach to the final research question which had two parts.

1. By incorporating external variables into the parameter surrogate models, PINNs can learn the mapping from external factors to the dynamical system's parameters. This part directly addresses the problem by establishing a connection between external factors and system parameters.

2. Additional modifications to the neural network architecture, including fixed Fourier transformations, customized non-negative output activation functions, and adoption of a multi-branch structure, can enhance PINN's training process and accuracy. This second part attempts to improve the stability and efficiency of training, as well as the generalizability of the resulting models, particularly in scenarios with limited observational data.

To meet these goals, a novel approach using physics-informed neural networks was developed to learn the mapping from external conditions to internal parameters of dynamical systems. This approach replaces time-dependent parameters with neural networks that map external variables to system parameters. The external-to-internal neural networks are designed with a dual-branch architecture to capture both year-to-year patterns and residual fluctuations. Furthermore, we implement a modified absolute activation function to ensure parameter positivity. These neural networks, along with the system state's neural network, are jointly trained to optimize a multi-task objective function, which not only learns from observational data but also enforces adherence to underlying dynamical laws. We evaluate this method through a case study on mosquito population modeling, comparing its performance against traditional empirical parameter estimation. Results indicate that the learned parameters improve the accuracy of the dynamical system, demonstrating the potential of this data-driven, hybrid approach in dynamical modeling.

The implementation of this framework is publicly available at `https://github.com/dinhvietcuong1996/pinn-external`.

7.2 Incorporating External Factors

Consider a dynamical system defined by a state vector $u(t)$ over the time interval $[0, T]$. The system's behavior is influenced by external factors represented by the vector $a(t)$, which affect the internal parameters $\theta(a(t))$. The evolution of the state u is described by a system of ODEs as shown in Equation 7.1.

$$\frac{du}{dt} = f^{(i)}\left(t, u, \theta\right), \quad i = 1, 2, \ldots, F. \tag{7.1}$$

A set of observations $\mathcal{D}_u = \{(t_1, u_1), (t_2, u_2), \ldots, (t_j, u_j), \ldots\}$ is provided, along with a predefined function $A : t \mapsto A(t)$ that approximates the external variables $a(t)$ with sufficient accuracy. It is important to note that some variables of the system state u may be costly or impossible to collect, so these observations may only cover part of the states. The primary objective is to determine a mapping from the external factors a to the system parameters θ, such that the resulting state u not only corresponds to the observed data $\mathcal{D}_u$ but also satisfies the ODE system described in Equation 7.1. Thus, this approach to inverse problem modeling seeks to integrate data-driven learning with the underlying physical principles represented by the ODEs.

In this framework, the state u is approximated by a neural network U that maps time t to the estimated state $u(t)$. Simultaneously, each unknown parameter is represented by a neural network Θ that takes $A(t)$ values as inputs and generates the corresponding parameter values. These neural networks are jointly optimized to minimize the objective function 7.2, which compromises loss terms 7.3 and 7.4.

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{ODE} = \mathcal{L}_{data} + \frac{1}{F}\sum_{i=1}^{F} \lambda_i \mathcal{L}_{f^{(i)}} \tag{7.2}$$

$$\mathcal{L}_{\text{data}} = \frac{1}{N_u} \sum_{(t_i, u_i) \in \mathcal{D}_u} \left(U(t_i) - u_i\right)^2 \tag{7.3}$$

$$\mathcal{L}_{f^{(i)}} = \frac{1}{N_f} \sum_{j} \left| \frac{dU}{dt} - f^{(i)}\left(t_j, U(t_j), \Theta(A(t_j))\right) \right|^2 \tag{7.4}$$

In loss functions 7.2, 7.3 and 7.4, λ_i represents weights that balance the objectives within the loss function. N_u denotes the number of state observations, while N_f

represents the number of collocation points t_j randomly sampled from the interval $[0, T]$. In cases where the observed state is incomplete, unavailable entries are masked out in the calculation of the data loss term $\mathcal{L}_{data}$.

The objective function consists of two components: $\mathcal{L}_{\text{data}}$ which ensures that the predictions generated by the model U closely align with the observed data, and $\mathcal{L}_{f^{(i)}}$ which minimizes the ODE residuals. Through the simultaneous optimization of these terms, we aim to derive a system state and parameter set that not only fits the empirical observations but also adheres to the ODE system.

The above framework is extended from the PINN framework presented in Chapter 6, and is depicted in Figure 7.1 where the function A (the white blue-bordered box) is newly added. The method introduces a novel aspect by employing parameter networks Θ that accept inputs from the function A values, illustrated by the arrow from Function A (the upper white blue-bordered box) to the neural network Θ (the lower blue box). These trained networks are reusable for predicting parameters under any external conditions, as depicted the arrows leading from inference A values (the lower blue-bordered box) through the model Θ. Additionally, the method includes the implementation of a Multi-branch Fourier-feature Multi-Layer Perceptron (Multi-branch FourierMLP) for these parameter networks to enhance their generalization capabilities. Finally, a new output activation function is introduced to enforce positivity constraints on both the system state and parameters.

FourierMLP [249] was implemented for the neural network architecture. It has demonstrated significant enhancements in both convergence speed and accuracy for PINN training [41] The FourierMLP, given an input x, produces an output y as in Equations 7.5.

$$\begin{aligned} h^{(0)} &= [\cos(Bx); \sin(Bx)] \\ h^{(l)} &= \phi_h(W^{(l)}h^{(l-1)} + b^{(l)}), l = 1, \ldots, L-1 \\ y &= \phi_o\left(W^{(L)}h^{(L-1)} + b^{(L)}\right) \end{aligned} \tag{7.5}$$

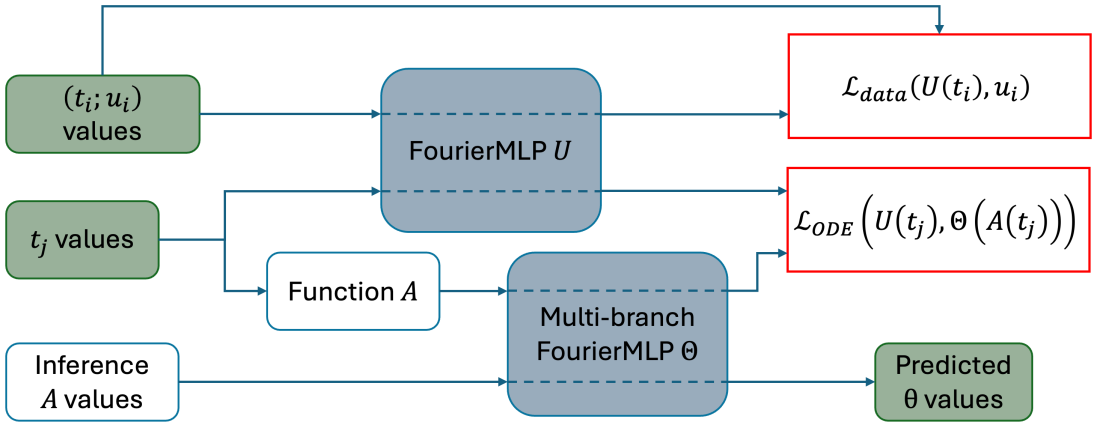


Figure 7.1: **External PINN Framework.** There are two groups of neural networks, one named U for the system state (the upper blue box), and the other named Θ for estimating the system parameters (the lower blue box). There are two data for the loss computations, the observations (t_i, u_i) (the upper green box) and the collocations points t_j (the lower green box). The data loss $\mathcal{L}_{data}$ is computed based on the observations (t_i, u_i) and the outputs of the state neural network U evaluated at t_i. The ODE loss $\mathcal{L}_{ODE}$ is calculated at random collocation points t_j, it involves the predictions of the state model U at t_j and the parameter model at $A(t_j)$ (the white blue-bordered box), which are external factors at t_j. Only the two neural networks (the two blue boxes) are trained, while the function A is fixed. After training, one can use the network Θ to predict parameters θ at any external factor values, as depicted by the bottom row of the figure.

In Equations 7.5, B denotes a random matrix with entries drawn from a normal distribution $\mathcal{N}(0, \sigma)$. $W^{(l)}$ and $b^{(l)}$ represent the weights and biases of appropriate dimensions, respectively. During training, B remains fixed while the weights and biases are optimized. L denotes the number of hidden layers, and ϕ_h represents the element-wise activation function applied to hidden layers. The Gaussian Error Linear Unit (GeLU) activation function [245] is used for ϕ_h, providing a smooth non-linearity to the model. To ensure non-negativity of state and parameters, we introduce a soft absolute function as the output activation ϕ_o in Formula 7.6, where ϵ is empirically set to 10^{-4} based on experimental results.

$$\phi(x) = \sqrt{x^2 + \epsilon} - \sqrt{\epsilon}, x \in \mathbb{R} \tag{7.6}$$

The high-dimensional nature of external variables $A(t)$, coupled with limited training data, often leads to poor generalization in trained models. To address this

issue, we propose a multi-branch architecture for the external-to-parameter networks Θ, as illustrated in Figure 7.2. This architecture comprises multiple branches, each consisting of a separate FourierMLP that processes a distinct group of inputs. The outputs from these specialized branches are then aggregated in subsequent layers to generate the final prediction.

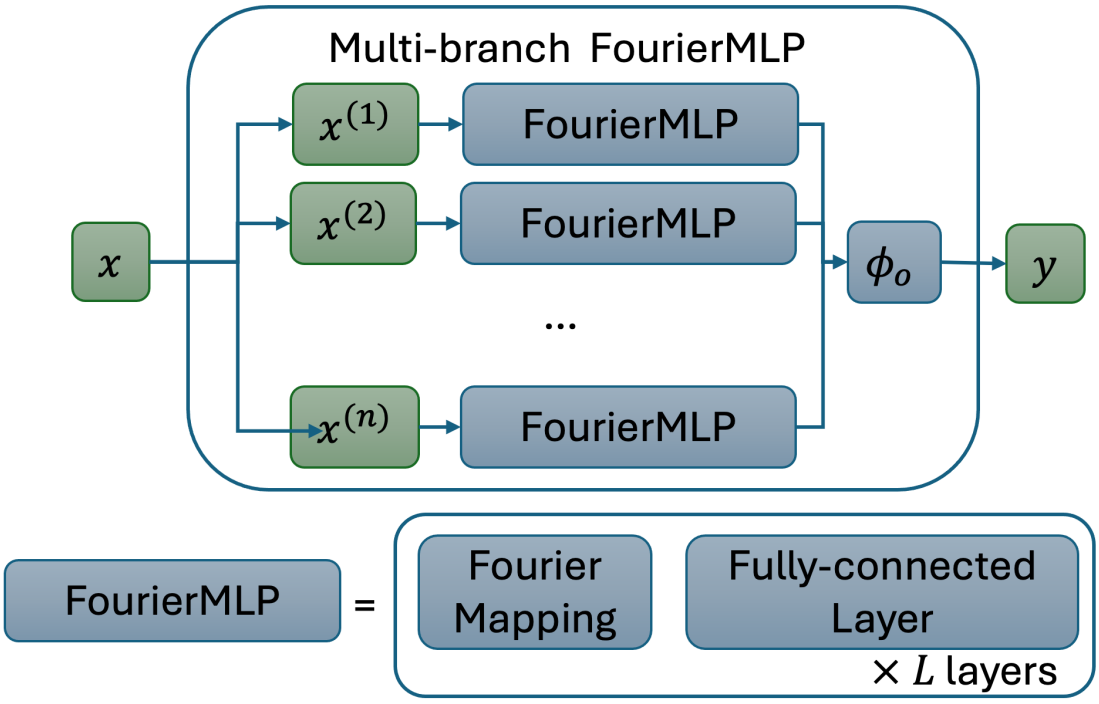


Figure 7.2: FourierMLP and Multi-branch FourierMLP architecture

The proposed architecture can be formally described as follows: Let $x^{(1)}, \ldots, x^{(n)}$ represent n distinct groups of external factors. Each input vector $x^{(i)}$ is independently processed by a corresponding FourierMLP$^{(i)}$ branch. The resultant outputs from these branches undergo summation, followed by an activation function ϕ_o. This is mathematically expressed by the Equation 7.7.

$$y = \phi_o \left(\sum_i \text{FourierMLP}^{(i)} \left(x^{(i)} \right) \right). \tag{7.7}$$

The optimization of the objective function is performed using the gradient-based Adam optimizer [203]. All differentiation operations, including ODE derivatives

and optimization gradients, are computed via automatic differentiation provided by the PyTorch framework [8]. To enhance convergence and accuracy, we implement ODE normalization and gradient balancing techniques as proposed in [243]. ODE normalization rescales neural network inputs and outputs, reformulating the ODE loss function to maintain these quantities within reasonable ranges. Gradient balancing adaptively adjusts the weights λ_i throughout training to maintain balance across tasks in multi-objective optimization. Furthermore, we resample the collocation points for ODE residual calculations at each training step, drawing from a uniform distribution over the time domain. To facilitate convergence, we initially train the networks using solely the data loss, enabling the network U to capture the general solution shape before training both objectives. Lastly, in scenarios where the function A is not directly available but its measurements are abundant, we train a separate FourierMLP to approximate this function. This neural network is frozen during the PINN training process, serving as a fixed input to the main model.

7.3 Evaluation

In this section, a series of experiments are presented which are used to evaluate the extended PINN framework. The data used for the experiments is described in Section 2.3.3.

7.3.1 ODE system

In this experiment, we apply our proposed methodology to an inverse problem using the same mosquito ODE system in Chapter 6 but repeated here to assist with interpreting Equations 7.8. The mosquito life cycle is divided into 10 stages: Egg (E), Larva (L), Pupa (P), Emerging Adults (A_{em}), Nulliparous Bloodseeking Adults (A_{b1}), Nulliparous Gestating Adults (A_{g1}), Nulliparous Ovipositing Adults (A_{o1}), Parous Bloodseeking Adults (A_{b2}), Parous Gestating Adults (A_{g2}) and Parous Ovipositing Adults (A_{o2}).

The system dynamics are described by the set of ODEs described in Equations 7.8.

$$\begin{cases} \frac{dE}{dt} & = \gamma_{Ao}(\beta_1 A_{o1} + \beta_2 A_{o2}) - (\mu_E + f_E)E \\ \frac{dL}{dt} & = f_E E - \left(m_L\left(1 + \frac{L}{\kappa_L}\right) + f_L\right)L \\ \frac{dP}{dt} & = f_L L - (m_P + f_P)P \\ \frac{dA_{em}}{dt} & = f_P \sigma e^{-\mu_{em}\left(1+\frac{1}{\kappa_P}\right)} P - (m_A + \gamma_{Aem})A_{em} \\ \frac{dA_{b1}}{dt} & = \gamma_{em} A_{em} - (m_A + \mu_r + \gamma_{Ab})A_{b1} \\ \frac{dA_{g1}}{dt} & = \gamma_{Ab} A_{b1} - (m_A + f_{Ag})A_{g1} \\ \frac{dA_{o1}}{dt} & = f_{Ag} A_{g1} - (m_A + \mu_r + \gamma_{Ao})A_{o1} \\ \frac{dA_{b2}}{dt} & = \gamma_{Ao}(A_{o1} + A_{o2}) - (m_A + \mu_r + \gamma_{Ab})A_{b2} \\ \frac{dA_{g2}}{dt} & = \gamma_{Ab} A_{b2} - (m_A + f_{Ag})A_{g2} \\ \frac{dA_{o2}}{dt} & = f_{Ag} A_{g2} - (m_A + \mu_r + \gamma_{Ao})A_{o2} \end{cases} \tag{7.8}$$

The parameters in Equations 7.8, already detailed in Table 6.2, are referred to as empirical formulas and serve as a baseline for our analysis.

The main goal is to learn a mapping from meteorological measurements to system parameters of the system and then validate this mapping by applying the trained models to infer parameters from unseen data. These predicted parameters are then used to simulate the mosquito population. Sensitivity analysis (Appendix C.2) reveals that the Pupa development rate f_P (and correspondingly, the larva development rate $f_L = 1.65 f_P$) exerts the most significant influence on the overall system state, particularly on the quantity $A_{b1} + A_{b2}$. Therefore, we choose to learn the parameter f_P.

7.3.2 Experimental Configuration

The experiment begins with the preprocessing of mosquito data and the establishment of lower and upper bounds for all data columns, a crucial step for ODE Normalization. A simulation is executed using parameters derived from climate condition data via empirical formulas. The resulting state values from this simulation serve to define the bounds for the system state. Collected mosquito counts are rescaled to align with

the bounds, and a 5-day-window Spline smoothing is applied to reduce data noise. Parameter bounds are determined by the values obtained from empirical formulas, while climate data bounds are set based on available measurements.

For the training process which used the 2-year dataset, the function A defined over the time domain plays a critical role. A FourierMLP is trained to interpolate meteorological measurements for any real-valued time t within the domain. This model's architecture consists of 256 Fourier features, followed by three hidden layers, each containing 128 units. The model generates two sets of external features: one comprising three meteorological variables (temperature, humidity, and precipitation), and another representing the day of the year, ranging from 0 to 365.

The architecture of the neural networks is structured as follows: The system state network U consists of 256 Fourier features followed by three hidden layers, each containing 128 units. For the parameter network, a dual-branch FourierMLP is implemented, where each branch has a layer of 128 Fourier features and three hidden layers of 64 units. The first branch processes 7-day historical meteorological data, while the second branch uses day-of-year as input. This configuration enables the latter branch to capture intrinsic annual patterns, while the former learns the impact of meteorological conditions on mosquito development rates. Both networks utilize GELU activation functions in their hidden layers and employ the soft absolute function (Equation 7.6) to enforce non-negativity.

The neural network A is trained for 300,000 epochs, with the checkpoint yielding the lowest root mean squared error (RMSE) being saved. PINN training initially focuses solely on data loss for 10,000 steps, followed by 290,000 steps incorporating the full objective function. PINN checkpoints are saved every 500 steps, and we select the one yielding the best RMSE when simulating with PINN-learned parameters. PINN checkpoints are saved every 500 steps, and the model producing the lowest RMSE when simulating with PINN-learned parameters is selected as the final model. This model is subsequently validated using the designated validation dataset.

Results are presented by comparing simulations using PINN-learned parame-

ters against those using baseline parameters, using graphical representations and quantitative metrics. For PINN-learned simulations, we utilize the initial conditions extracted from the trained network U in the training period for the 7-year validation dataset. Baseline simulations, in contrast, adopt an initial condition of 300 for each state vector component, aligning with the work in [44]. To ensure comparability, all simulation outputs and observational data are normalized to the [0, 1] range for metric calculations. Our validation metrics include the root mean squared error (RMSE) between $A_{b1} + A_{b2}$, the RMSE of the weekly difference in $A_{b1} + A_{b2}$, and the 7-day 0.2-prominence peak detection recall, precision, and F1-score.

The experiment is implemented in Python, leveraging PyTorch for neural network architecture and optimization tasks. We accelerate the training process using a GeForce GTX 4090 GPU. All simulations are executed via SciPy [231], employing its finite-difference ODE solver [248]. The peak detection algorithm is also from this package.

7.3.3 Results

Figure 7.3 shows the simulation results using empirical formula parameters and PINN-derived parameters, alongside smoothed observation data. Figure 7.4 illustrates the parameter f_P. The PINN simulation demonstrates a close fit to the training data, yielding an RMSE of 42,060 (0.029 normalized), which is significantly lower than the empirical formula's RMSE of 453,863 (0.248 normalized). This indicates the PINN's successful learning of the parameter f_P to match observed data. The learned development rate f_P predominantly remains near zero throughout the year, exhibiting small peaks around day-of-year 60-90, corresponding to a minor surge in mosquito populations. The most significant peaks are observed around day 160, aligning with annual peaks in mosquito counts. Notably, the prediction of f_P shows unexpected peaks at the end and beginning of the calendar year, likely a false peak due to the near-zero number of mosquitoes during this period providing limited information for learning. In contrast, the parameters derived from the empirical

formula exhibit a nearly linear relationship with temperature. This leads to an earlier-than-observed explosion in mosquito numbers in the second year and higher-than-expected populations during summer and later periods.

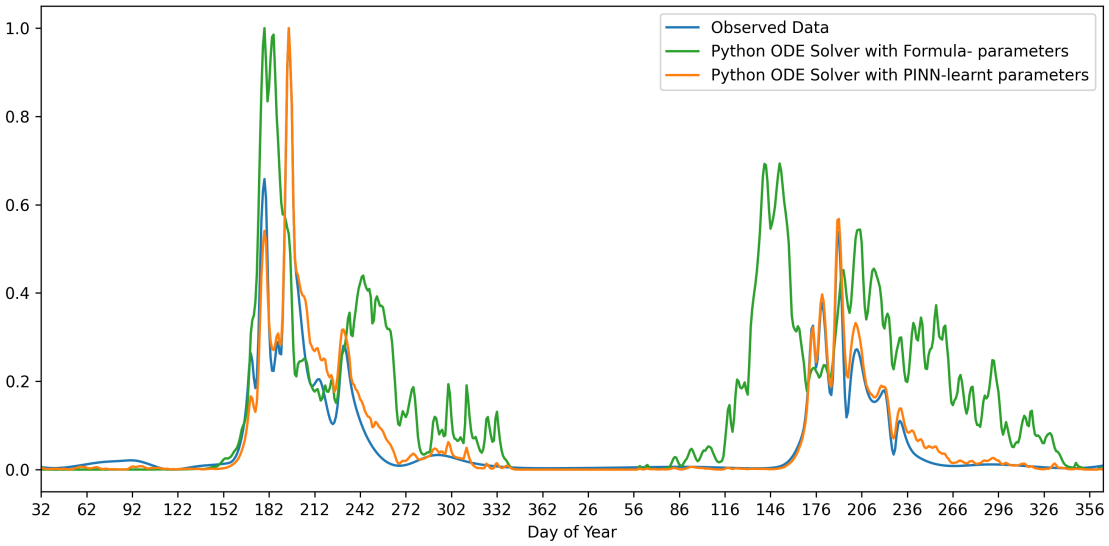


Figure 7.3: Mosquito Population Simulations, $A_{b1} + A_{b2}$, the training period.

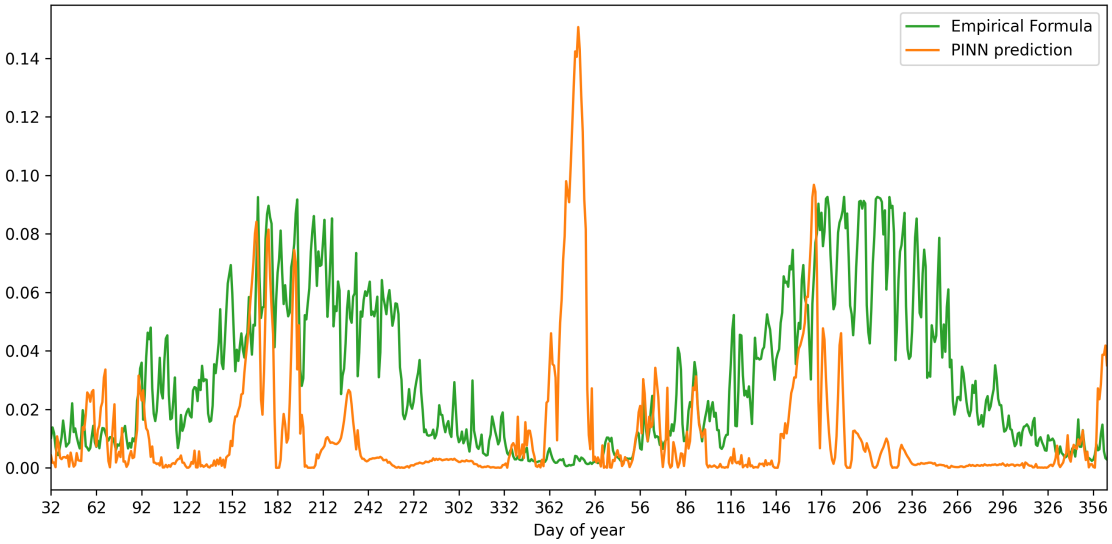


Figure 7.4: Parameter f_P prediction, the training period

Figure 7.5 illustrates the $A_{b1} + A_{b2}$ values from simulations during the validation period, along with weekly observed mosquito counts. Both simulations demonstrate the ability to capture general patterns of annual mosquito population dynamics. The Formula-based simulation predicts elevated populations over an extended period, spanning from approximately day 100 to day 300 of the year. In contrast, the

PINN simulation appears to more accurately predict population explosions, typically occurring around day 180 and rapidly decaying after day 250. However, both simulations exhibit a slower population decay compared to observed data. Regarding peak detection, the Formula-based simulation tends to identify a higher number of peaks, potentially reducing its precision, although it still fails to capture observed peaks accurately. The PINN method, on the other hand, is more conservative but still aligning more closely with significant observed peaks.

The respective f_P predictions are plotted in Figure 7.6. The parameter predictions for the validation period exhibit similar characteristics to those of the training period. Notable features include a prominent peak around days 160-180, minor peaks between days 60-90, and a substantial spike at the beginning of the year. The differences across the years are visually hard to identify, primarily attributable to minor fluctuations in climatic conditions. However, these seemingly small differences lead to significant variations in mosquito population dynamics. A more comprehensive analysis of these predictions, beyond the scope of this study, would be necessary to draw further conclusions.

Table 7.1 presents error metrics between the Formula and PINN parameters for validation simulations. The PINN method demonstrates superior performance across all metrics. Notably, PINN achieves lower RMSE values for overall error (0.1791), weekly differences (0.1728), and second-order weekly differences (0.2927) compared to the Formula approach, with the figures 0.2622, 0.2054 and 0.3328, respectively. In peak detection, PINN substantially outperforms the Formula method, with higher recall (0.5625 vs 0.1250), precision (0.6250 vs 0.0938), and F1 score (0.5667 vs 0.1071). These results suggest that PINN offers added information into the parameter prediction, expressed by more accurate simulations and peak prediction capabilities.

7.4 Ablation Study

For an ablation study, we investigate how the additions of neural network architectures and the activation function affect the performance of the framework. The

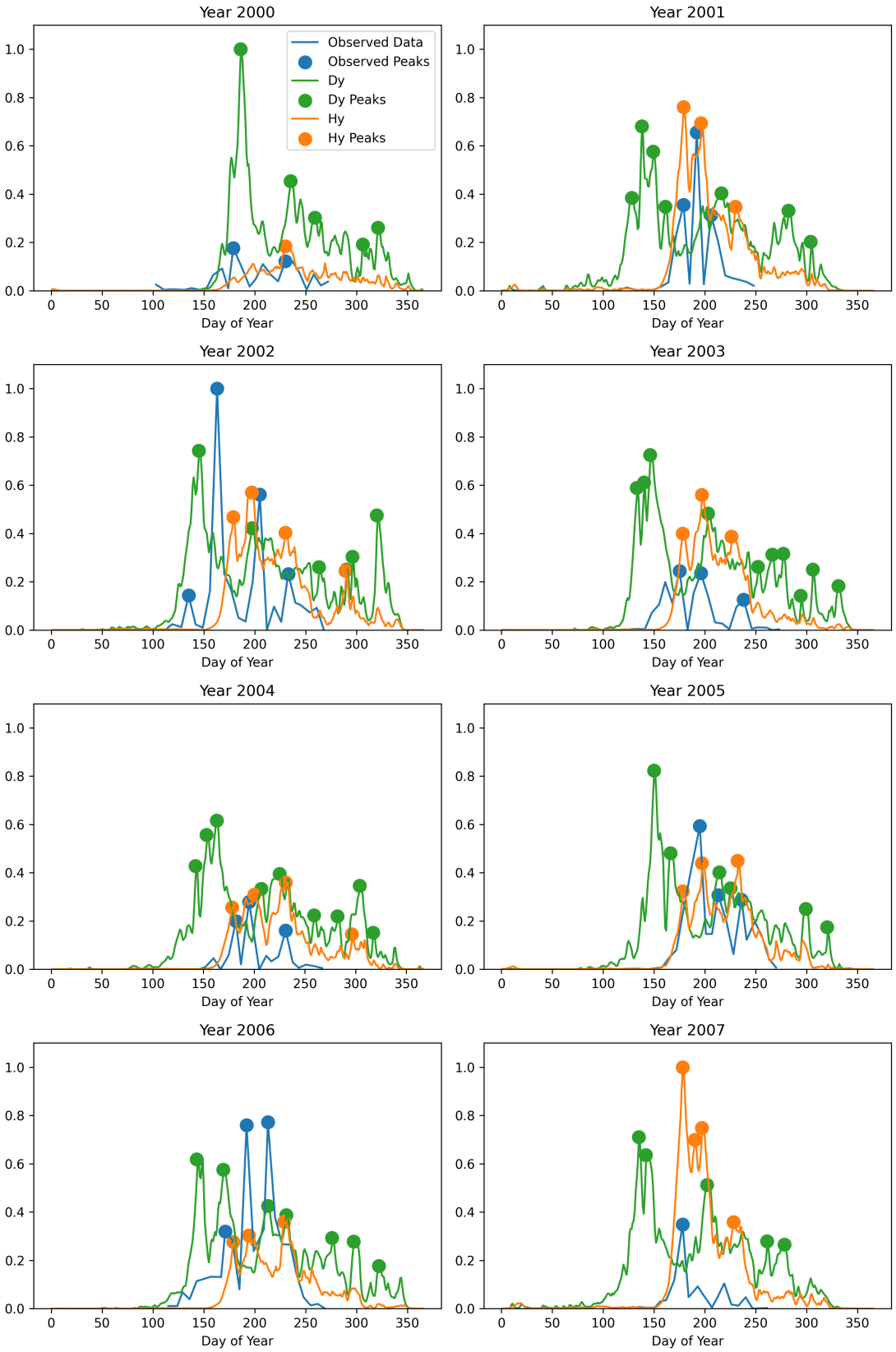


Figure 7.5: Mosquito Population Simulations, $A_{b1} + A_{b2}$, the validation period.

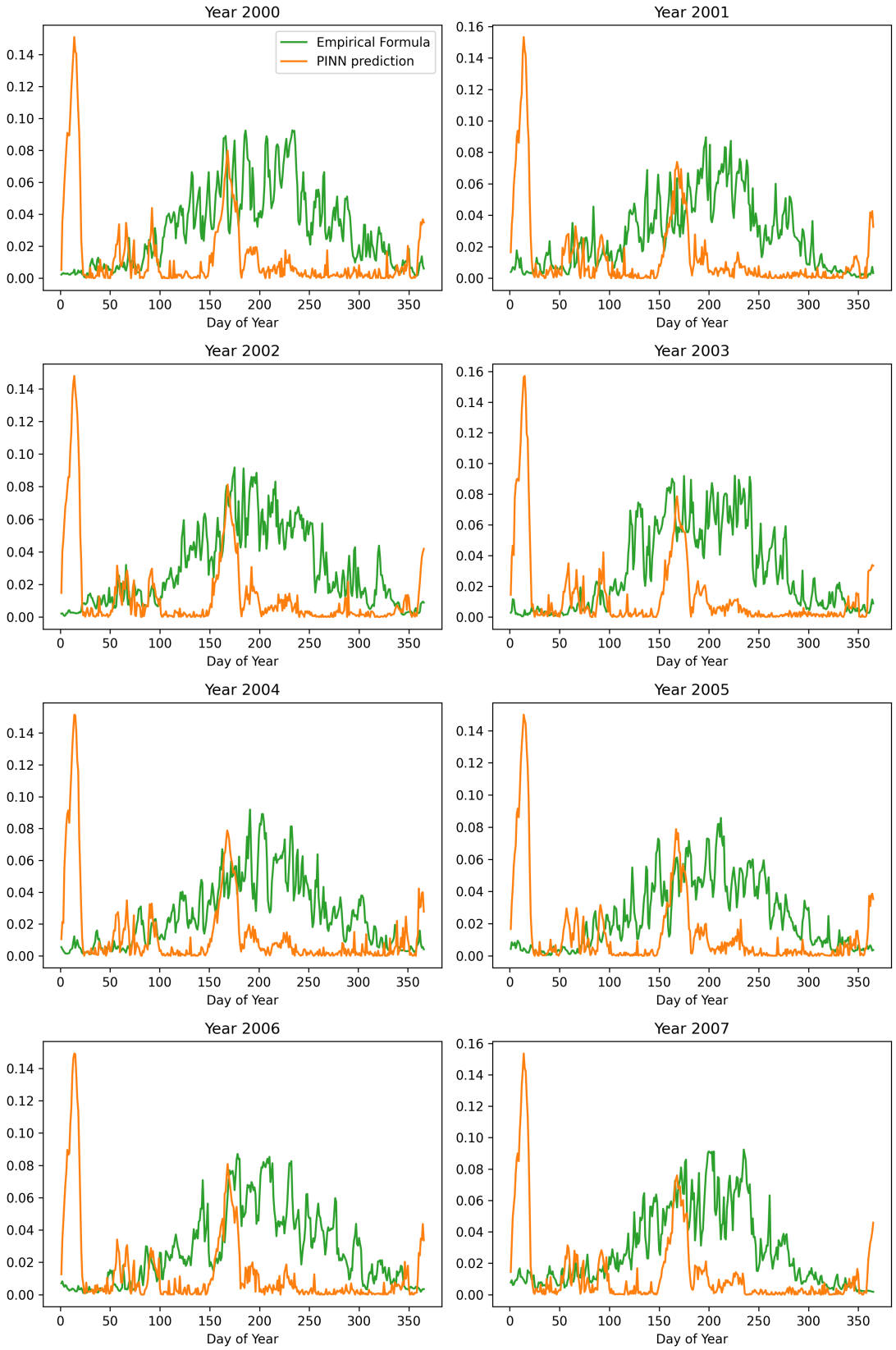


Figure 7.6: Parameter f_P prediction, the validation period.

Table 7.1: **Error Metrics**. Six metrics are presented: the first three are RMSE metrics where lower values indicate better performance, and the last three are peak metrics where higher values are better. The best results for each metric are highlighted in **bold**.

	Formula	PINN
RMSE	0.262200	**0.179100**
Weekly Diff RMSE	0.205400	**0.172800**
Weekly 2nd Diff RMSE	0.332800	**0.292700**
Recall Peak	0.125000	**0.562500**
Precision Peak	0.093800	**0.625000**
F1 Peak	0.107100	**0.566700**

experimental procedure is the same as in Section 7.3. Specifically we first train PINNs using data from the training period, during which the model learns how meteorological variables affect the mosquito parameters. Then, the trained parameter network is used to predict parameters for the test period, and these parameters are substituted into the ODE system to simulate the mosquito population which is the population predictions. The difference is that we vary one component at a time while keeping all other factors constant to observe the effect of each change. We first examine the architecture of the neural networks and then the non-negativity activation function.

7.4.1 Model architectures

In this experiment, we aim to assess the effect of Fourier feature layers and the branching architecture of neural networks on the performance of the framework. We consider four configurations of neural network architectures:

1. **MLP**: All of the neural networks for the system state U and system parameters Θ use standard MLPs, similar to those in Chapter 6 and in conventional PINNs [43].

2. **FourierMLP**: All of the neural networks U and Θ use FourierMLPs, similar to those in [41].

3. **Branched MLP**: For comparison, we use a branched version of the MLP,

which is similar to the multi-branch Fourier MLP described in Section 7.3, but the Fourier layer is replaced with a normal fully connected layer, making each branch a standard MLP. This multi-branch MLP is used for the parameter networks Θ, while the state network remains an MLP.

4. **Branched FourierMLP**: Finally, our proposed architecture, which is the same as in Section 7.3, uses a FourierMLP for the network U and a multi-branch FourierMLP for the parameters Θ.

As the accuracy of PINNs does not heavily depend on the capacities of the neural networks [108], it was only necessary to experiment with one hyperparameter setting for each configuration. The hyperparameters for the neural networks of the four configurations are set as follows: for all the state networks, we utilize a network with four hidden layers; the first hidden layer has 256 units, and the three subsequent hidden layers each have 128 units. For the parameter networks, the non-branched versions use a similar setting: one layer of 256 units and three layers of 128 units. For the branched versions, each branch uses half the number of units, that is, 128 units for the first hidden layer and 64 units for the other three. All other hyperparameters are set the same as in Section 7.3, including the use of the GELU activation function for hidden layers and the soft absolute function with $\epsilon = 10^{-4}$ for the output.

Table 7.2 presents the simulation error metrics obtained when simulating the mosquito dynamical models using the parameter f_P learned from the four different neural network configurations. It can be seen that PINNs across all architectures outperform the empirical formula. Our proposed architecture, the Branched FourierMLP, achieves the best overall performance, exhibiting the lowest RMSE and the highest scores in peak detection metrics. When comparing the standard MLP to its branched counterpart, we observe that the RMSE values are similar (0.1937 for MLP vs 0.1979 for Branched MLP), but the Branched MLP exhibits marginal improvements in peak detection metrics. Specifically, the peak recall increases from 0.1458 to 0.2708, the precision peak from 0.3750 to 0.4375, and the peak F1-score from 0.2083 to 0.2917. This suggests that the branching architecture enhances the model's

capacity to capture features relevant to peak occurrences, such as annual patterns. Interestingly, the standard MLP outperforms the FourierMLP in terms of RMSE (0.1937 vs 0.2479), suggesting that the FourierMLP may be overfitting the data due to its higher complexity. However, when the FourierMLP is integrated into a branched architecture, as in Branched FourierMLP, the model not only mitigates overfitting but also leverages the Fourier features to capture periodic patterns more effectively. The Branched FourierMLP achieves a lower RMSE of 0.1791 compared to both the standard MLP and the FourierMLP, and it significantly improves peak detection metrics over both models. These results suggest that the branching architecture allows the neural network to generalize better while the Fourier features enable it to capture periodic components in the data. The combination between the branching structure and Fourier features in the Branched FourierMLP contributes to its superior performance, making it a robust framework for learning system parameters in PINNs.

Table 7.2: **Error Metrics from Parameters learned from PINNs with different network architectures**. The best results for each metric are highlighted in **bold**, while the second best results are underlined.

	MLP	FourierMLP	Branched MLP	Branched FourierMLP
RMSE	0.193658	0.247858	0.197923	**0.179100**
Diff RMSE	0.187189	0.207561	0.191858	**0.172800**
2nd Diff RMSE	0.314825	0.348473	0.312093	**0.292700**
Recall Peak	0.145833	0.166667	0.270833	**0.562500**
Precision Peak	0.375000	0.085714	0.437500	**0.625000**
F1 Peak	0.208333	0.101010	0.291667	**0.566700**

7.4.2 Activation Functions for Non-negativity

In this experiment, the aim is to explore different activation functions to enforce the non-negativity of system parameters and states. All of the considered functions are plotted in Figure 7.7. The baseline activation function is the identity function, which returns the exact input provided to it. Other popular activation functions that express the non-negativity property are ReLU and its smoothed version, Softplus [250]. While ReLU and Softplus are popular activation functions, they suffer from the critical drawback of causing dying neurons [251]. This phenomenon occurs when the

input to the ReLU is negative, resulting in zero gradient propagation and causing the neuron to stop learning. This problem is exacerbated when applied to PINNs. As discussed in Chapter 6, PINNs often converge to the trivial zero solution, which is precisely how dying neurons are triggered. This convergence causes PINNs to become stuck at a local minimum and prevents further learning.

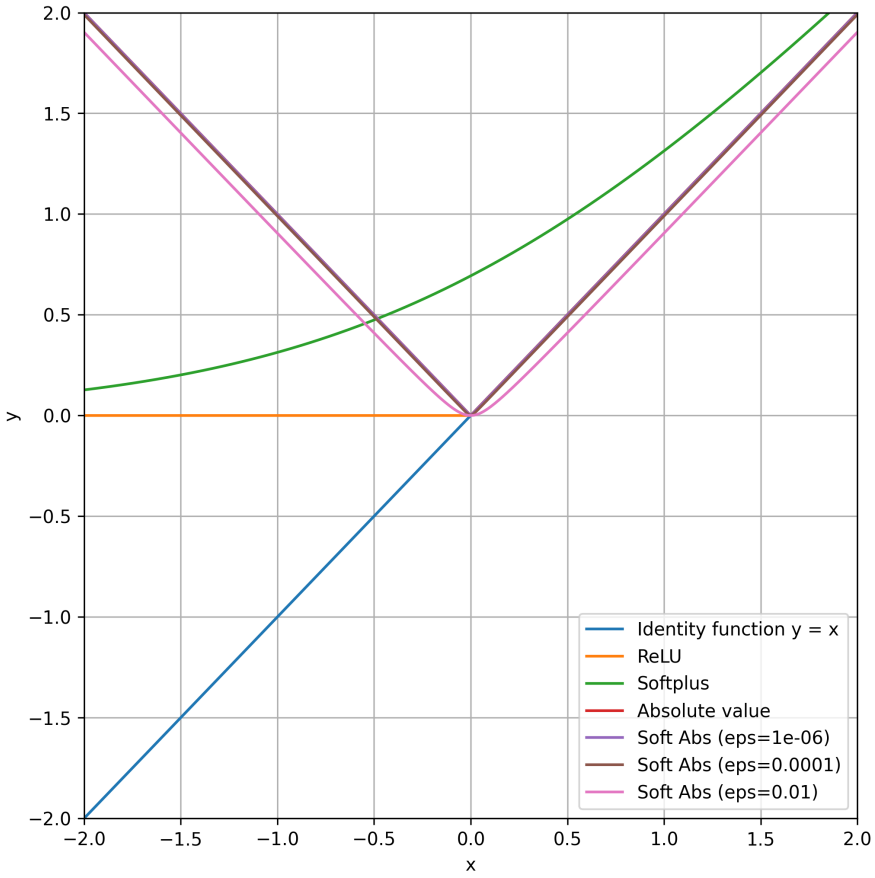


Figure 7.7: **Non-negativity Activation Functions**. The Soft Abs functions with $\epsilon = 10^{-6}$ and $\epsilon = 10^{-4}$ appear very close in the plot, closely resembling the positive part of ReLU and the identity function.

To address this issue, we propose using the absolute value function to impose non-negativity on the states and parameters. The absolute value function effectively fixes the zero-gradient problem by ensuring a non-zero gradient in the negative domain. However, the original absolute value function makes it difficult for the model to optimize values to zero, as the gradient is discontinuous at 0.0. Therefore, we propose using a soft version of the absolute value function, called SoftAbs, defined in Equation 7.6. When ϵ is small, the differences from the standard absolute value

function are minimal, as seen in Figure 7.7, but SoftAbs is significantly more effective, as we will demonstrate in our experiments.

Table 7.3 shows the error metrics obtained from simulations using parameters learned by PINNs with different output activation functions. Overall, the SoftAbs activation function with $\epsilon = 10^{-4}$ demonstrates superior performance across multiple metrics. It achieves the highest scores in all three peak detection metrics, and ranks second in RMSE and third in first-order difference error. Other configurations of the SoftAbs function also exhibit significant improvements, particularly in peak detection and second-order difference errors. For example, SoftAbs with $\epsilon = 10^{-6}$ attains the lowest second-order difference RMSE of 0.2853 and achieves the second-highest peak F1-score of 0.5083. Abs improves peak detection compared to ReLU and the identity function; however, its performance in RMSE metrics is limited. These results suggest that softening the derivatives of the absolute value function aids the model in better detecting peaks while maintaining comparable RMSE values. ReLU achieves the lowest first-order difference RMSE at 0.1706, while the identity activation function comes second in overall RMSE with a value of 0.1637. This performance can be attributed to our re-selection procedure, which involves simulating training period after training.

Table 7.3: **Error Metrics from Parameters learned from PINNs with different final activation functions**. The best results for each metric are highlighted in **bold**, while the second best results are underlined.

	Identity	ReLU	Softplus	Abs	$\epsilon = 10^{-6}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-2}$
RMSE	0.1637	0.1868	0.1981	0.2008	0.1903	0.1791	**0.1558**
Diff RMSE	0.1768	**0.1706**	0.1916	0.1787	0.1748	0.1728	0.1809
2nd Diff RMSE	0.3031	0.2927	0.3138	0.3080	**0.2853**	0.2927	0.3194
Recall Peak	0.2708	0.0416	0.2083	0.2500	0.4583	**0.5625**	0.3333
Precision Peak	0.2708	0.1250	0.3125	0.4375	**0.6250**	**0.6250**	0.5000
F1 Peak	0.2500	0.0625	0.2083	0.3005	0.5083	**0.5667**	0.3625

For comparative analysis, Figure 7.8 plots the predictions of the learned pupa development rate f_P over the training period using three different activation functions: Identity, ReLU, and SoftAbs with $\epsilon = 10^{-4}$. In the intervals from day 160 to 210 in the first year and from day 140 to 200 in the second year, all three models

approximately agree on the values learned. This convergence is attributed to the high mosquito activity during these periods, which results in increased observational mosquito counts and thus, provides information for the models to learn. The critical differences among the models lie in the range of about days 50 to 90 in both years. During these periods, the SoftAbs activation function effectively captures the small peaks in mosquito counts by predicting surges in the corresponding f_P. In contrast, the ReLU activation function encounters difficulties in learning values near zero, with predictions remaining mostly zero during these intervals. This issue arises due to the dying neuron phenomenon, where neurons become inactive and are unable to learn once their outputs reach zero. The Identity activation function fails to satisfy the non-negative constraints in the physical modeling of mosquito populations, making its predictions unrealistic. Nevertheless, the Identity function achieves the lowest RMSE on the training data, a performance attributable to its relaxed constraints, which allow for a better fit to the data despite violating physical plausibility.

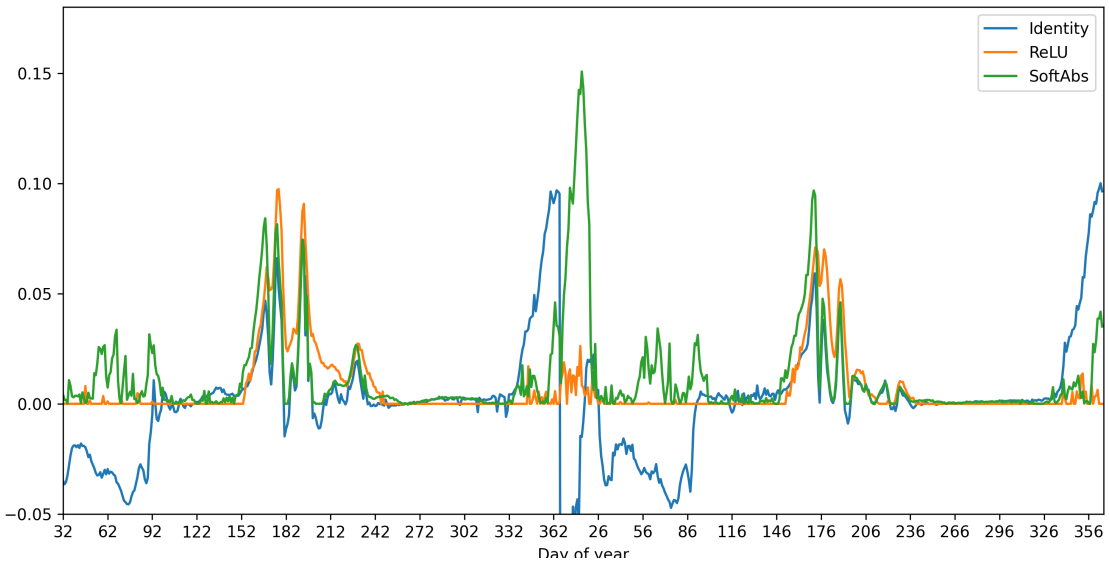


Figure 7.8: Parameter f_P predictions with different output activation functions, training period

It is important to note that the models are selected based on the simulation errors evaluated on the training data, which may not be the fully converged checkpoints of the training process. In our experiments, the selected models utilizing ReLU and Softplus activation functions correspond to early training stages In experiment,

the models with ReLU and Softplus are selected in the first few training steps, at 11,000 and 12,000 out of a total of 300,000 training steps, when the training loss had not yet been fully minimized. This observation suggests that the target checkpoints for these two activation functions may not reside at local minima in the loss landscape, introducing uncertainty into the PINN training process. In contrast, models employing the Identity, Abs, and SoftAbs activation functions were selected at later stages of training, around 250,000 out of 300,000 training steps, where the training loss was more thoroughly minimized.

7.5 Conclusions

In this chapter, novel enhancements to the PINN approach were introduced for inverse problems. The proposed modifications not only facilitate learning the influence of external factors on the internal parameters of dynamical systems but also enhance the generalization capability of the learned model to unseen data. Where parameter networks incorporate external values as inputs rather than time as in conventional PINN approaches, this enables direct learning of parameters with respect to external factors while trying to adhere to observed data and physical laws simultaneously. Secondly, the multi-branch architecture for parameter networks allows for separate processing of different input feature types, thereby improving the capture of each feature group's impact on the parameters. The proposed methods were applied to a case study in mosquito population modeling, aiming to learn the impact of meteorological conditions on mosquito development rates. Comparative analysis revealed that PINN-derived parameters outperformed traditional parameter formulas in several metrics, including the errors of the population curve, its first- and second-order derivatives, and peak detection metrics (recall, precision, and F1-score). These results demonstrate the successful integration of a data-driven, highly complex neural network model with a dynamical system governed by ODEs, specifically in the context of mosquito population modeling.

An ablation study further reinforces the effectiveness of our proposed modifications.

By comparing different neural network architectures, we observe that the multi-branch FourierMLP architecture significantly outperforms traditional MLP, FourierMLP and multi-branch MLP, achieving lower RMSE and higher peak detection metrics. The results show that the branching structure greatly improves the ability to predict population peaks; and when combined with Fourier features, the model's capability to capture these peaks is enhanced even further. These findings from the ablation study validate our proposals in PINN frameworks for dynamical systems.

Overall, these experiments demonstrate the significant benefit of incorporating prior knowledge into neural networks as demonstrated by improved predictive accuracy across a substantial set of experimental evaluations.

Chapter 8

Conclusions

In this final chapter, we conclude the dissertation by briefly summarizing the different steps taken in this research. We then highlight the contributions and novel aspects of our work. Finally, we suggest some opportunities for further research in this area.

8.1 Dissertation Overview

In this section, we summarize the outputs from this body of research by re-examining the initial hypothesis and the degree to which the research question have been effectively addressed. In broad terms, it is our belief that we have successfully demonstrated our hypothesis that integrating prior knowledge into neural networks does enhance overall predictive performance. We will now briefly revisit those initial research questions and how they were addressed.

8.1.1 Chapter 4: Neural Networks in Real-Life Applications

Neural networks, inspired by biological neurons, have become a cornerstone of modern artificial intelligence and machine learning. It has advanced numerous fields, achieving state-of-the-art results in many applications. In the first part of this research, we applied neural networks to two new machine learning challenges: exosome classification and oxygen uptake modeling to address the first research question: How effectively can generic neural networks be deployed as machine learning solutions in

areas such as health and sports?

Exosome Classification. In the first task, we addressed the challenge of classifying exosome-derived SERS spectra into healthy, hyperglycemic, and hypoglycemic classes. The aim was to evaluate if neural networks could successfully distinguish between exosome signals derived from endothelial cells cultured under different conditions. We adopted a multi-step preprocessing approach that included smoothing, background signal removal, and normalization to prepare the spectral data for machine learning models. The classification model used in this study was a Multi-Layer Perceptron (MLP).

Results indicated that the MLP model achieved a moderate overall accuracy of approximately 66.7%. While it demonstrated promising performance, particularly for identifying hypoglycemic samples, challenges emerged in distinguishing between normal and hyperglycemic conditions. This difficulty could be attributed to inherent similarities in the spectral profiles of normal and hyperglycemic samples, possibly due to the presence of normal-like tissue in hyperglycemic samples. The approach showed positive results in capturing distinguishing features from heterogeneous and complex Raman signals. These findings underscored the potential for using machine learning in general and importantly, the specific application of neural networks for health diagnostics.

Oxygen Uptake Estimation. The second task attempted to predict individual oxygen uptake using wearable sensor data collected during jogging and team based activities. Several neural network models are evaluated, including linear regression, MLP, long short-term memory networks (LSTM), and one-dimensional convolutional neural networks (1D-CNN). The results highlighted that MLP model performed well, achieving a validation RMSE of 3.18. However, the BiLSTM model demonstrated the best generalizability, achieving a test RMSE of 4.98.

Research Question 1: How effectively can generic neural networks be deployed as machine learning solutions in areas such as health and sports? In conclusion, our empirical studies in Exosome Classification and Oxygen

Uptake Estimation indicate that neural networks can indeed provide decent predictive performance in health diagnostics and sports science. Fro exosome classification, a MLP achieved an over accuracy of approximately 66.7% in distinguishing healthy, hyperglycemic and hypoglycemic classes, while for oxygen uptake estimation, an MLP and a BiLSTM demonstrated strong performance (RMSE of 3.18 on validation and 4.98 on test, respectively). However, in both cases, dataset size and data complexity limited the models' ability to generalize fully. Hence, while generic neural networks are feasible and beneficial, they often demand careful data processing, feature engineering, and sufficient training samples to achieve practical performance levels. This led us to experiment with a different form of neural network for the next step in our research.

8.1.2 Chapter 5: Graph Neural Networks

To approach the next question, if graph neural networks could be deployed to exploit the structural information in graphs, we began by modeling data as graph with a case study of bike sharing sytems. We developed three types of network structures: Spatial Bike Graph Networks (SBiGN), Temporal Bike Graph Networks (TBiGN), and Spatio-Temporal Bike Graph Networks (STBiGN). For analysis, we applied traditional graph metrics such as degree, strength, closeness, and betweenness, along with community detection algorithms to identify groups of stations with similar characteristics. The analysis revealed valuable insights beyond what traditional flat data could offer as we could understand not only the activity levels of individual stations but also the relationships and community structures within the network. Thus, graph-based representations were demonstrated to enhance spatio-temporal data analysis.

In the second part of our study, we developed a novel neural network architecture to predict air quality using an attention-based graph neural network. The weighting in attention layer helped learning the graph structure and the weighted sum of transformed features aggregated information from neighboring stations. The results

showed that the Spatio-Temporal Attention model achieved marginal improvements in predictive accuracy compared to simpler models like conventional neural networks. The attention mechanism allowed the model to learn adaptive weights between locations and historical steps, which contributed to slightly better performance. However, the gains were modest due to the limited size of the training dataset, suggesting that these advanced features might be more effective in scenarios with richer data.

Research Question 2: Can graph neural networks be deployed to exploit the structural information inherent in graph-based data? In Chapter 5, modeling bike-sharing data and air-quality data as graphs showed that leveraging structural information yields additional insights and marginally improved predictions over simpler models. By constructing Spatial Bike Graph Networks, Temporal Bike Graph Networks, and Spatio-Temporal Bike Graph Networks, we uncovered relationships and community structures that would have been difficult to observe with traditional, "flat" datasets. Furthermore, an attention-based GNN for air-quality forecasting provided slightly better predictive accuracy—although the improvement was modest, partly due to a limited training set. Overall, these results confirm that GNNs can indeed exploit graph structure, but they also highlight the need for more robust regularization and richer data to realize their full potential.

8.1.3 Chapter 6: Physics-Informed Neural Networks

The third research question asked if it was possible to train neural networks to accurately represent and predict the behavior of dynamical systems, when governed by a system of ordinary dynamical equations. For this task, we investigated the usage of Physics Informed Neural Networks (PINNs). Specifically, we developed a PINN framework with enhanced techniques to not only train neural networks to approximate the solution of systems of ODEs, but also solve related inverse problems. Methodological contributions included ODE normalization for balancing scales, gradient balancing to minimize the imbalance in gradients across multiple

scales of ODEs, multi-phase training to preserve temporal causality in the system's dynamics, and simplifed domain decomposition for problems with large domains. These techniques were ablationally validated using both the Lorenz system and a complex mosquito population dynamics model.

Research Question 3: Can neural networks be extended to accurately represent and predict the behavior of dynamical systems governed by a system of ordinary differential equations? We demonstrated in Chapter 6 that Physics-Informed Neural Networks can be successfully adapted to solve forward and inverse problems for systems of ODEs. Through techniques such as ODE normalization, gradient balancing, multi-phase training, and simplified domain decomposition, our approach provided accurate approximations of both the Lorenz system and a mosquito population model. These enhancements mitigated common challenges such as multi-scale imbalances and convergibility. We showed that neural networks integrated with domain knowledge (i.e., the governing ODEs) can indeed learn and predict complex dynamical behaviors. This bridged the gap between data-driven and physics-informed approaches, demonstrating great potential for real-world applications in ecological systems and beyond. However, a final step would require that we attempt to apply the framework with real-life observations and validations.

8.1.4 Chapter 7: PINN Optimization

In the final part of this study, we attempted applying PINNs into real world applications using actual observations. Specifically, our objective was to estimate the parameters in the case study of mosquito population modeling. This research addressed the final research question regarding the ability of neural networks to learn the effects of external factors on dynamic system parameters. Firstly, parameter neural networks take the external factors as input directly, in contrast to traditional PINNs that use coordinates. Secondly, these networks are designed as a multi-branch MLP, augmented with Fourier features. Lastly, a non-negative activation function is proposed for all the neural networks requiring non-negativity constraints. These

methodological proposals are to improve the generalization of the learned parameters on unseen external factors.

Research Question 4: Can we determine how neural networks incorporate external factors on dynamic system parameters and validate any solution using real-world observational data? In Chapter 7, we extended PINNs by introducing parameter neural networks to map external factors (e.g., temperature) directly to system parameters. Using multi-branch architectures and Fourier features, this method learned to generalize parameter values for unseen external conditions in a real-world mosquito dynamics case. Despite some limitations—such as interpretability issues, inaccuracies under zero-value observations, and the challenge of identifying initial conditions—our results show that neural networks can capture the impact of external factors on ODE-based system parameters more effectively than traditional empirical formulas. The final piece of research further enhanced the ability of physics-informed neural networks to solve inverse problems, highlighting the potentials of data-driven parameter identification in guided-physics dynamics modeling.

8.2 Contributions

- **Exosome Classification.** This research presents a novel combination of surface-enhanced Raman spectroscopy (SERS) with a neural network to classify exosomes derived from normal and dysfunctional human aortic endothelial cells. The key contributions in the machine learning aspect include: (1) the application of a feed-forward neural network for spectral classification, a three-step data preprocessing pipeline (smoothing, background removal, and normalization) to enhance SERS signal quality, and an extensive hyperparameter search (1,260 configurations) to fine-tune the MLP model. The results demonstrate that the model effectively distinguishes hypoglycemic samples but faces challenges in differentiating hyperglycemic and normal samples, highlighting the complexity of exosome-derived spectral signatures. Other novelty includes

the leverage of AI with peptide modified plasmonic pore arrays for exosome classification in the context of endothelial dysfunction.

- **Oxygen Uptake Estimation.** This pilot study represents the first application of ML models to predict VO_2 during simulated team sports activities using wearable sensor data. We examine the influence of different sensor configurations, revealing that multi-sensor setups (e.g., combining torso with leg or arm sensors) can improve prediction accuracy. We compare multiple regression techniques, from traditional linear regression to advanced deep learning models such as LSTM, CNN, and MLP, and evaluates their performance using both raw and engineered (MAD-based) features. By employing a leave-one-subject-out cross-validation approach, the study closely mimics real-world scenarios, demonstrating that these non-invasive, real-time monitoring techniques can provide personalized physiological feedback critical for optimizing athletic performance and reducing injury risk.

- **Bike Sharing System.** This study introduces a unified, graph-based framework for analyzing bike-sharing usage by integrating both spatial and temporal dimensions into a multi-level network model. Its contributions include the development of three types of networks—Spatial Bike Graph Networks, Temporal Bike Graph Networks, and Spatio-Temporal Bike Graph Networks, that progressively incorporate greater granularity and complexity to capture overall traffic flow, temporal dynamics, and hidden patterns of station similarity, respectively. Algorithms are proposed to optimize network construction, such as thresholding methods for trimming weak edges while preserving strong connectivity, and clustering techniques for automatically grouping time intervals with similar network structures. The framework not only enhances visualization and quantitative analysis through centrality metrics and community detection but also provides insights for network optimization.

- **Graph Neural Networks for Air Quality Forecasting.** The work intro-

duces a novel neural network architecture that integrates attention mechanisms in both spatial and temporal dimensions to enhance air quality forecasting. By stacking temporal attention layers before and after a spatial attention layer, the model simultaneously learns and fuses complex spatio-temporal dependencies while offering interpretability through adaptive weight assignments. This approach overcomes key limitations of traditional recurrent and graph neural networks—such as slow training, limited long-term dependency capture, and fixed spatial relations, and demonstrates improved predictive performance on real-world air quality data from Hanoi, Vietnam.

- **Adapting Physics-Informed Neural Networks to Improve ODE Optimization in Mosquito Population Dynamics.** This research introduces a novel PINN framework specifically designed for complex ODE systems, incorporating several key innovations to enhance stability and accuracy. The approach normalizes the differential equations and assigns individualized gradient-balancing weights to each equation, ensuring a well-balanced optimization process. It employs a three-phase training strategy along with a progressive causal training method, which gradually expands the time interval to preserve temporal causality throughout training. Additionally, the framework simplifies domain decomposition while still effectively managing large temporal domains. These enhancements stabilize training, prevent convergence to trivial solutions, and enable robust solutions for both forward and inverse problems, as demonstrated on the Lorenz system and a mosquito population dynamics model.

- **Incorporating External Factors to PINNs.** The last piece of work introduces a novel enhancement to PINNs for inverse problems by incorporating external factors into parameter neural networks, enabling PINNs to learn the mapping from external conditions to system parameters. Additionally, it proposes a multi-branch FourierMLP architecture for parameter networks, effectively capturing different feature types separately, and introduces a SoftAbs

activation function to enforce non-negativity while avoiding dying neuron issues common in ReLU and Softplus. The approach is validated through a mosquito population modeling case study, demonstrating superior accuracy over traditional empirical parameter estimation, particularly in capturing seasonal peaks and generalizing to unseen data. An ablation study further confirms the effectiveness of these contributions, establishing a more robust and generalizable PINN framework for dynamical systems.

8.3 Suggestions for Further Research

Despite the promising findings, the studies and results presented in this dissertation have several areas for improvement. When exploring neural networks in health in Chapter 4, the **exosome classification** model showed a positive but impractical accuracy. The degraded performance was also likely due to the small dataset size that restricted the model capabilities. Additionally, the use of only raw spectra with minimal feature engineering makes neural networks harder to effectively learn the underlying information. Furthermore, apart from the MLP architecture, other neural network architectures were not fully explored. Future research could focus on applying feature engineering approaches, such as detecting peaks, which represent important characteristics in Raman spectra. Classical statistical features or Fourier-based features may also be valuable for improving the model. Moreover, exploring more advanced architectures, such as convolutional neural networks or long short-term memory networks, could uncover deeper features, thereby enhancing classification accuracy for spectra data.

In the investigation of neural networks in sports in Chapter 4, the **oxygen uptake** project also had similar weaknesses. The small dataset size limited the ability of deep neural network models, such as LSTMs and CNNs, to generalize and increased the risk of overfitting. The manual feature engineering for some models introduced potential biases and limited scalability. Also the predefined input window sizes may not have fully captured the variability in movement patterns and physiological responses.

Additionally, the computational efficiency of the models for real-time VO_2 prediction was not assessed, leaving potential latency concerns unaddressed for wearable device applications. Directions for further research could be expanding the dataset to include more diverse participants and fitness levels. Leveraging data augmentation and transfer learning could also potentially improve models' performance and efficiency. Exploring more windows of data could optimize computational efficiency, making the models more suitable for deployment in wearable systems. Moreover, other advanced architectures, like Transformers, temporal convolution networks, could better capture complex temporal dependencies.

When modeling and analyzing **bike sharing** data as graphs, certain limitations of the proposed framework prevent it from practical usage. Firstly, there is no fully functional software tool to automate the construction and analysis of these networks. The time scales used, such as hourly or daily analysis, are selected based on heuristics, potentially missing more complex temporal patterns that data-driven methods could capture. Moreover, only correlation was used for the spatio-temporal networks, other possible ways to uncover relationships could provide deeper insights. The available graph metrics and algorithms are not rich, which may result in an incomplete understanding of the network's structure. In future work, one can develop a software tool to automate these analyses, adopting adaptive data-driven approaches for time scale selection, exploring advanced spatio-temporal methods such as causality detection, and developing graph metrics and algorithms to gain a knowledge out of the networks. These enhancements will make the framework more robust and applicable to a wider range of real-world problems.

In Chapter 5, the proposed attention-based spatio-temporal **graph neural network** for air quality forecasting has critical limitations of overfitting, making its improvement over other models negligible. Moreover, the model's reliance on predefined spatial and temporal thresholds reduces its adaptability to varying geographical contexts or unseen data, as these thresholds may not universally capture the underlying relationships. Additionally, while attention mechanisms provide some degree of

interpretability, the explainability of the model's decisions is underexplored, making it difficult to fully understand the impact of specific features or temporal patterns. Furthermore, the attention mechanism itself is limited to a restricted number of time steps due to the computations needed, which restricts its capacity to capture longer temporal dependencies. To address these limitations, stronger regularization techniques should be incorporated to mitigate overfitting, potentially leveraging domain-specific knowledge such as physical laws governing air quality dynamics. Also, developing adaptive methods to learn spatial and temporal thresholds from data is needed to increase the model's flexibitility. Finally, an exploration of the model's explainability is essential to provide deeper insights into the air quality forecasting process and to enhance trust in the model's predictions.

In the **development of PINNs** in Chapter 6, some areas require further improvement. Firstly, the domain decomposition approach was only evaluated on forward problems, its applicability to inverse problems was untested. Secondly, the framework was validated on a limited set of systems, the Lorenz system and a mosquito population model, its effectiveness across a broader range of ODE systems is still in question. Moreover, the focus of the development has primarily been on accuracy, without sufficient attention to computational efficiency or training time. Furthermore, significant error accumulation over extended prediction domain hindered long-term predictive accuracy. In the future, we would like to evaluate the framework on a more diverse set of systems, including systems related to climate. Future development of the method should take into account the convergence speed and resource requirements. Lastly, the error accumulation suggests for further refinement and improvement of the framework.

When attempting to apply **PINNs to actual mosquito data** in Chapter 7, the current method has certain drawbacks. Firstly, the learned mapping remains a black box, lacking the explainability factor is crucial for mosquito control and entomology in general. Although the parameter network decomposes its predictions by separating feature groups, its potential for explainability is highly unexplored.

Secondly, the learned models exhibit certain inaccuracies, particularly during periods of zero-value observations. Potential solutions may involve regularization techniques that leverage information from more informative training periods. Lastly, dynamical system simulations are inherently sensitive to initial conditions. The present study employs a fixed initial condition, which may not accurately represent the actual state of the system. Further investigation is required to develop more robust methods for identifying initial conditions based on available information. These limitations underscore the need for continued refinement of the proposed approach to enhance its applicability and reliability in real-world scenarios.

Appendix A

Error Metrics

A.1 Regression Error

Let $y_i, i = 1 \ldots, M$ be M reference values and $\hat{y}_i, i = 1 \ldots, M$ are the corresponding predictions. Let $\mathfrak{L}$ and $\mathfrak{U}$ be the pre-defined lower and upper bounds for the values. The error metrics used in this paper, including Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Median Absolute Percenrage Error (MDAPE) and Root Mean Squared normalized Error (nRMSE) are defined as followed

$$RMSE = \sqrt{\frac{1}{M}\sum_{i}^{M}(y_i - \hat{y}_i)^2} \tag{A.1}$$

$$MAE = \frac{1}{M}\sum_{i}^{M}|y_i - \hat{y}_i| \tag{A.2}$$

$$MDAPE = \text{median}_i\left(\frac{|y_i - \hat{y}_i|}{|y_i|}\right) \tag{A.3}$$

$$nRMSE = \frac{RMSE}{\mathfrak{U} - \mathfrak{L}}. \tag{A.4}$$

The values of these metrics range from 0 to ∞, with lower values indicating better model performance.

A.2 Classification Error

In the context of classification problems, let y_i and $\hat{y}_i$, where $i = 1, \ldots, M$, denote the true class labels and corresponding predictions for M samples, respectively. The classification accuracy is defined as

$$\text{accuracy} = \frac{|\{i|y_i = \hat{y}_i, i = 1, \ldots, M\}|}{M} \tag{A.5}$$

where $|A|$ is the cardinality of set A.

For a specific class c, three metrics, precision, recall, and F1-score are formulated as follows:

$$\text{precision}_c = \frac{|\{i|y_i = c, \hat{y}_i = c, i = 1, \ldots, M\}|}{|\{i|\hat{y}_i = c, i = 1, \ldots, M\}|} \tag{A.6}$$

$$\text{recall}_c = \frac{|\{i|y_i = c, \hat{y}_i = c, i = 1, \ldots, M\}|}{|\{i|y_i = c, i = 1, \ldots, M\}|} \tag{A.7}$$

$$\text{f1-score}_c = \frac{2 \cdot \text{precision}_c \cdot \text{recall}_c}{\text{precision}_c + \text{recall}_c} \tag{A.8}$$

In case there are multiple classes, the overall score can be averaged from class-specific metrics in two scenarios: macro or weighted. Macro-averaging simply calculate the unweighted average, does not take into account the number of samples in each class. In weighted averaging, the weights are the number of samples of the class, emphasizes the class with larger sample size.

All of the above metrics are bounded between 0 and 1, with higher values indicating better classification performance.

Appendix B

Graph-based Bike Sharing System Analysis

B.1 Temporal Bike Graph Networks (Daily TBiGN)

Table B.1 the top 5 stations with the highest **strength**, categorized by weekdays and weekends, and sorted by their weekday strength. Station strength, used as a measure of popularity, has been normalized to a daily value to facilitate comparison between the 5-day weekday period and the 2-day weekend period. The table reveals that the five busiest weekend stations also rank among the top 20 on weekdays, while the five busiest weekday stations remain within the top 50 during weekends. This indicates a notable difference in station usage patterns between weekdays and weekends. However, it is worth noting that three of the top 5 weekday stations also are also in top 5 of weekends. Exceptions to these trends include *Warehouse* and *Mountjoy Square*, where most trips are concentrated during weekdays, likely due to commuting patterns. Conversely, *Dun Laoghaire Dart (station)* and *Blackrock Main St.* serve as popular weekend destinations.

When analyzing the most frequently traveled routes, as illustrated by their **weights** in Table B.2, a similar variance is observed. Notably, the top five weekend routes are also found within the top seven weekday routes. Additionally, the node

Table B.1: Node Strength: Weekday vs Weekend

Station	**Weekday**		**Weekend**	
	Strength	**Rank**	**Strength**	**Rank**
Fairview Avenue Lower	155.0	**1**	145.5	**3**
Mountjoy Square South	116.8	**2**	110.5	8
Warehouse	113.2	**3**	15.5	50
Criminal Courts of Justice	102.2	**4**	191.0	**1**
Ranelagh Village	102.0	**5**	123.0	**5**
Dun Laoghaire Dart	97.8	7	169.5	**2**
Blackrock Main St.	48.8	20	134.5	**4**

strength values in both weekday and weekend networks exhibit an exponential decline, similar to the trend observed in the SBiGN network, though with a steeper rate of decline. Likewise, edge weights in the weekday and weekend TBiGN networks show an exponential decrease comparable to that of the SBiGN network. As with Table B.1, edge weight values have been normalized to allow for a comparison between the 5-day weekday and 2-day weekend periods.

Table B.2: Edge Weights: Weekday vs Weekend

Source	Target	Weekday		Weekend	
		Wgt.	Rank	Wgt.	Rank
Dun Laoghaire Dart	Honeypark Neptune Way	36.0	**1**	31.5	**3**
Criminal Courts of Justice	Phoenix Park Gate	29.6	**2**	47.5	**1**
Drumcondra Road Upper	Fairview Avenue Lower	26.2	**3**	24.0	**5**
DCU Glasnevin	Drumcondra Rd	23.4	**4**	9.0	38
Warehouse	Old Finglas Road	23.0	**5**	5.5	73
Dun Laoghaire Dart	At The Forty Foot	20.8	**6**	30.5	**4**
Blackrock Main St.	Dun Laoghaire Dart	19.8	**7**	44.5	**2**

Figures B.1 and B.2 illustrate the **strength** of stations and the **weight** of edges for the weekday and weekend networks, respectively. The main difference between these networks is that weekday connections displayed by large red edges are

predominantly between residential areas and office areas, particularly in northern Dublin. In contrast, weekend networks show more trips between the city center and the Blackrock-Monkstown area, likely representing leisure activities.

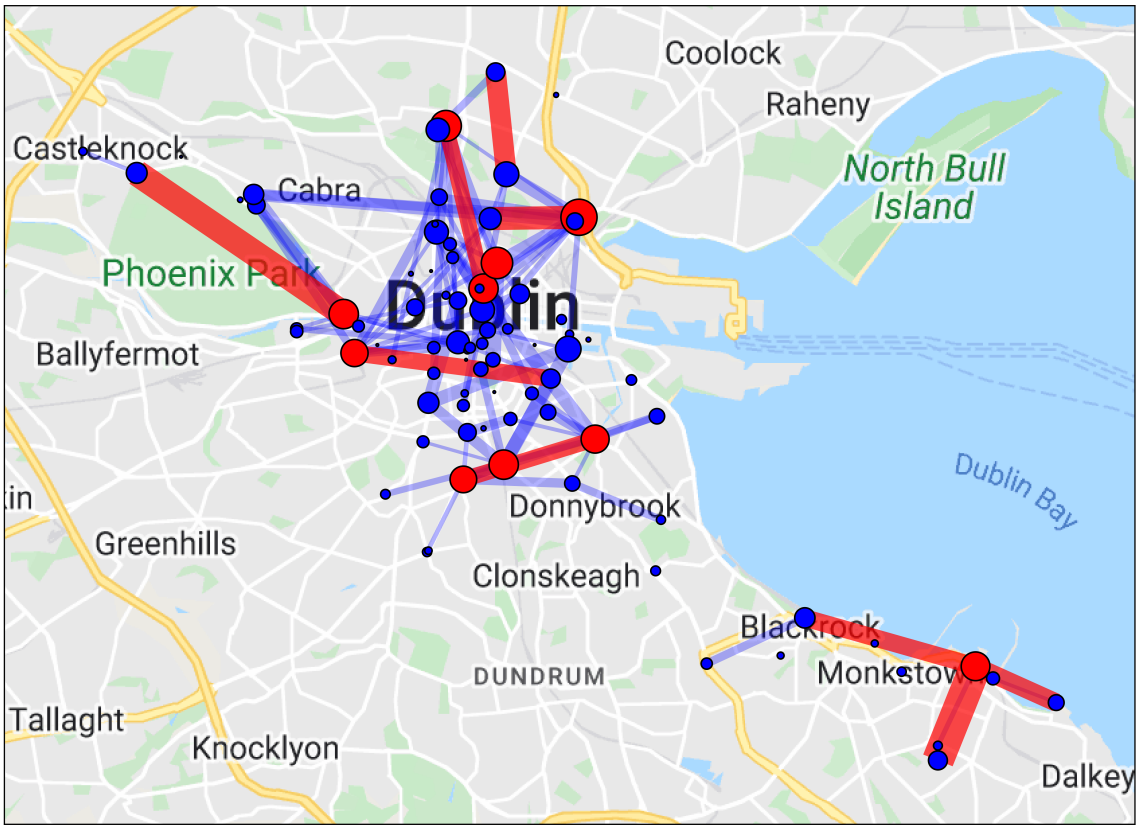


Figure B.1: **Daily Activity Networks. (Representation Networks of Weekday Clusters)**. Each circle represents a bike station and is sized according to its trip volume, with the 10 busiest stations (by trip volume) shown in red and all others in blue. Lines represent routes between stations and are also sized by trip volume; the 10 most frequently used routes are highlighted in red, and the remaining routes are shown in blue.

Figures B.3 and B.4 depict the community structures within the weekday and weekend networks, respectively. The weekend network exhibits four distinct communities, whereas the weekday network shows more spatially mixed communities, particularly around the central areas of Dublin. In both networks, the Blackrock-Monkstown area forms an independent community. Moreover, stations with strong suburban connections belong to the same community throughout the week. This suggests that on weekdays, users tend to follow established routes, supported by a few highly connected edges, which in turn generate many distinct communities. In contrast, on weekends, users travel over a broader spatial area in a less predictable

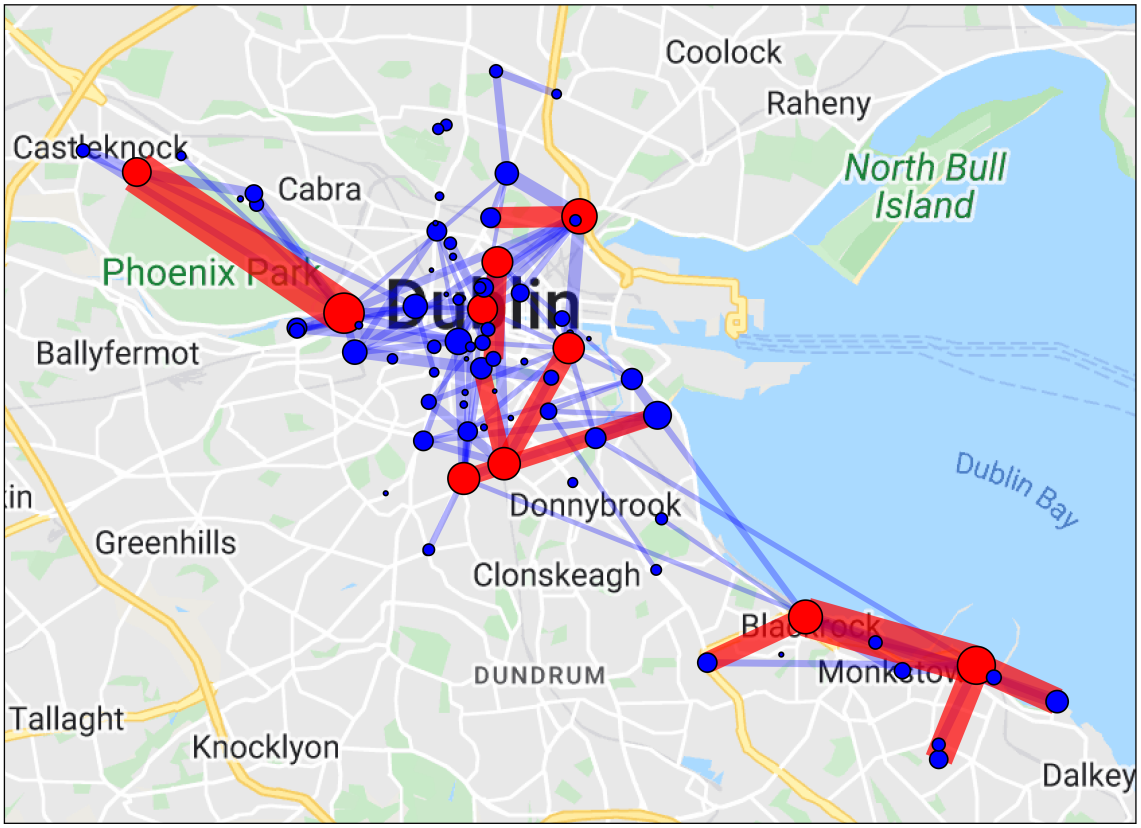


Figure B.2: **Daily Activity Networks. (Representation Networks of Weekend Clusters)**. Each circle represents a bike station and is sized according to its trip volume, with the 10 busiest stations (by trip volume) shown in red and all others in blue. Lines represent routes between stations and are also sized by trip volume; the 10 most frequently used routes are highlighted in red, and the remaining routes are shown in blue.

manner, leading to larger and more interconnected communities.

B.2 Spatio-Temporal Bike Graph Networks (Monthly STBiGN)

It is crucial to clarify that large nodes in STBiGNs do not necessarily indicate that the corresponding stations are popular with a high volume of trips. Rather, they reflect a greater number of stations share similar characteristics, even two low-activity stations can exhibit large node values. Furthermore, the edges in these correlation networks are less influenced by geographical locations and more by the properties of the surrounding station areas. In essence, this graph highlights stations with

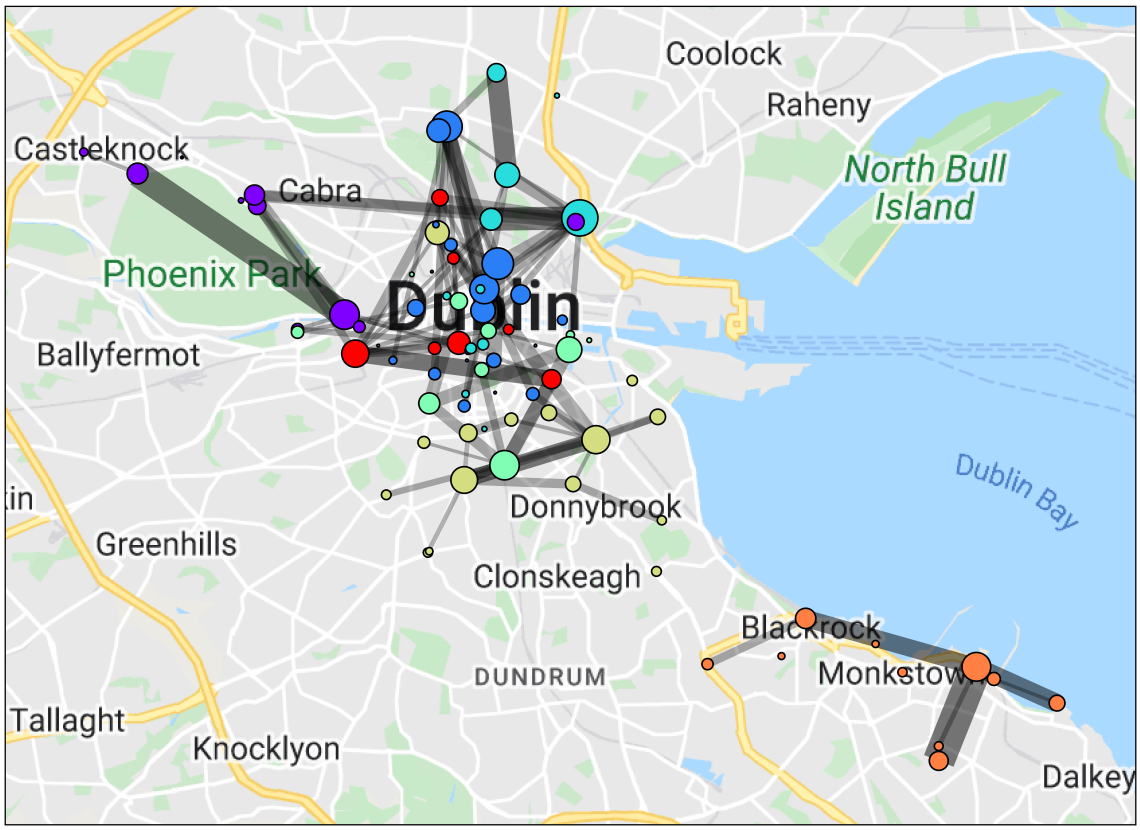


Figure B.3: **Daily Activity Communities. Weekday**. Each circle represents a bike station and is sized according to its trip volume. The node colors indicate different communities. Lines represent routes between stations and are also sized by trip volume.

analogous temporal patterns, where similar trip patterns occur at corresponding time intervals. Figure B.5 illustrates this network overlaid on a map of Dublin, revealing the detection of five distinct communities.

Overall, the transportation system experiences growth up until August 2020, followed by a decline in early 2021. For the remainder of the year, the total number of trips nearly returns to its previous peak, a trend reflected in the TBiGN graphs, but captured here with greater granularity. Figure B.6 plots the averaged monthly timeseries (for every month in the dataset) for the communities identified in Figure B.5, offering insights into the reasons why certain stations (nodes) clustered into communities in distinct areas of the city. It is important to note that community IDs serve merely as labels, and no inference is made from the label numbers. In terms of community size: community 15 (purple) consists of 13 stations; community 57 (bright sky blue) contains 26 stations; community 64 (mint green) includes 15

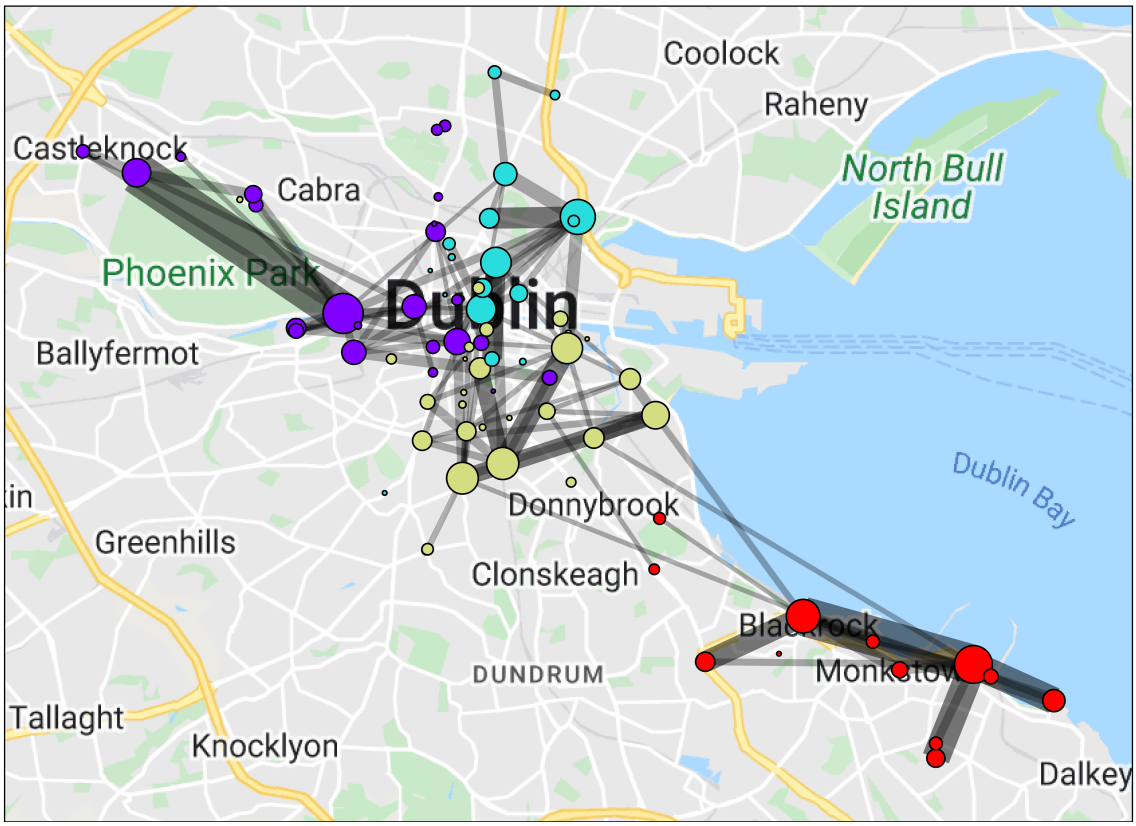


Figure B.4: **Daily Activity Communities. Weekend**. Each circle represents a bike station and is sized according to its trip volume. The node colors indicate different communities. Lines represent routes between stations and are also sized by trip volume.

stations; community 65 (pastel orange) has 20 stations; and community 69 (red) comprises 12 stations. The bright sky blue stations, primarily located in the southern part of the city center, and the red stations, which encompass the areas surrounding the city center, exhibit rapid growth during the first few months, followed by a sharp decline in activity in January 2021. While the red stations nearly fully recover their activity levels, the bright sky blue stations show only a marginal increase in trip numbers thereafter. The mint green stations, mainly situated north of the city center, and the pastel orange stations, mostly located in the suburbs, demonstrate steady growth, with a slight decline observed for the mint green stations. Lastly, the purple stations, concentrated in the southern region where Dublin connects with Blackrock, initially show growth in activity but remained relatively stable throughout most of the period, with a slight decrease noted in the final months.

To gain deeper insights, it is necessary to analyze some of the raw data in more

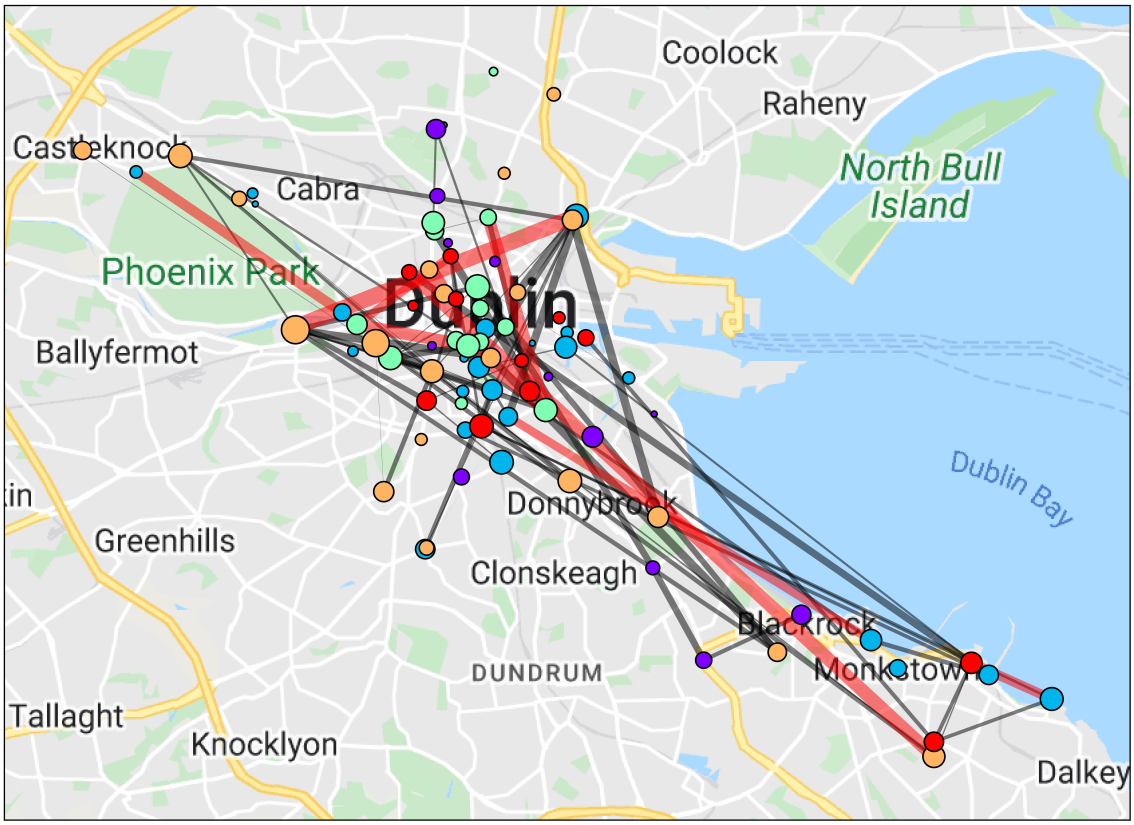


Figure B.5: **STBiGN Network: Monthly Timescale**. Each circle represents a bike station and is sized by its strength. The node colors indicate different communities, with four communities labeled: purple, bright sky blue, mint green, pastel orange, and red. Lines represent routes between stations and are sized by their correlation score.

detail. Table B.3 shows the node strength, the average weights, and the number of stations that exhibit positive, neutral, or negative correlations with each node in the monthly STBiGN. The columns are defined as follows: *Strength* refers to the sum of the weights connecting a station to all other stations; *Avg. Cor.* represents the average correlation weight for the station; *#Pos*, *#Neu*, *#Neg* indicate the number of stations that are positively, neutrally, or negatively correlated with the station. The thresholds for correlation are determined as 0.65 and 0.35. A station is considered to have a *positive* correlation with another if the weight between them is $w > 0.65$, a *negative* relationship if $w \leq 0.35$, and otherwise, the relationship is deemed *neutral*.

A full version of Table B.3 would contain 86 rows, for the purpose of this discussion we focus on stations with high, low, and neutral strength. Stations with the highest strength, such as *DCU Alpha* and *Rathmines*, are regarded as *central* nodes, showing

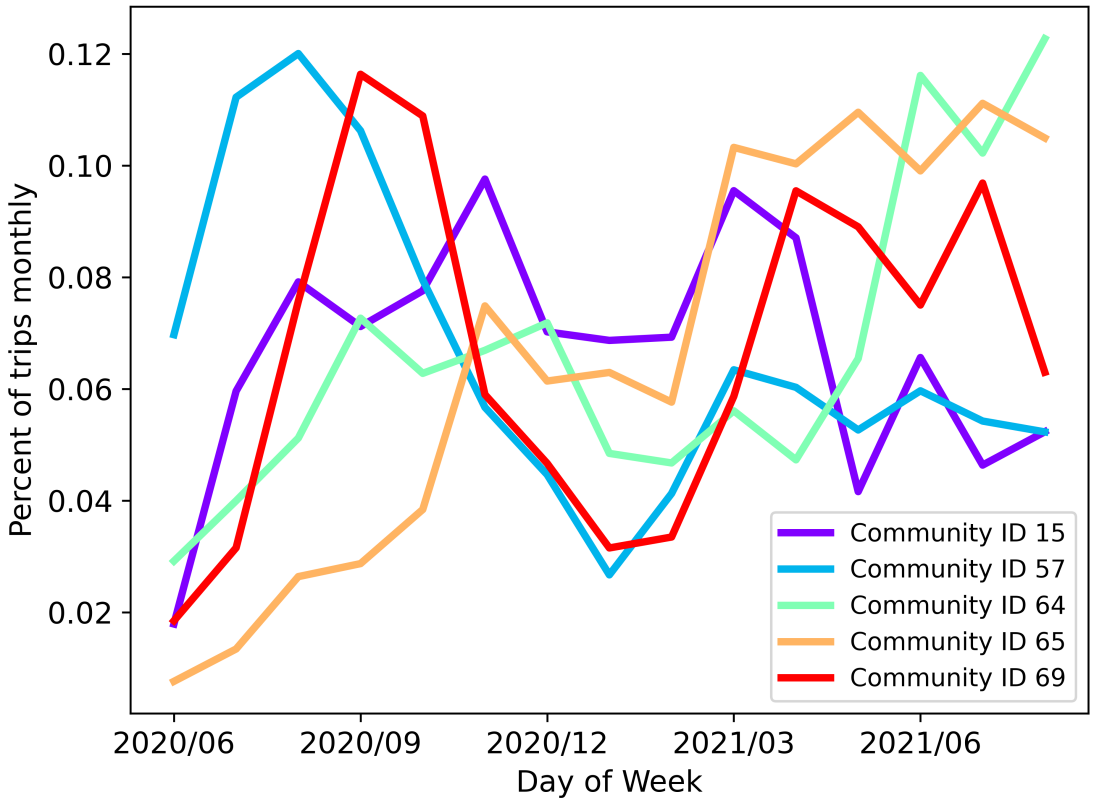


Figure B.6: Timeseries Communities in Monthly STBiGNs

Table B.3: Node (station) Strength in Monthly STBiGNs

Station	Strength	Avg. Cor.	#Pos	#Neu	#Neg
DCU Alpha	52.9	0.622	40	43	2
Rathmines	52.5	0.618	36	48	1
Dun Laoghaire Dart	52.5	0.618	33	51	1
...					
Pearse Street	43.8	0.515	15	59	11
Rathgar	43.7	0.514	23	38	24
Cathal Brugha Street	42.1	0.496	18	46	21
Sandymout Village	42.1	0.495	13	62	10
...					
Irishtown Rd	38.4	0.451	14	43	28
Parnell Street	38.3	0.451	6	56	23
Phoenix Park Gate	34.6	0.407	10	37	38

significant similarity to 47% and 42% of the other stations, respectively. This indicates that these stations are valuable for more detailed study (possibly using more granular networks) to better understand the activity patterns of a large portion of the network. Interestingly, both stations are part of the purple community, which

exhibits mixed correlation patterns with other communities. Neutral stations, with average correlation coefficients around 0.5, such as *Pearse Street* and *Sandymount Village*, show minimal correlation with other stations. This could imply that these stations are less useful for further analysis, as their activity patterns are almost equally positively and negatively correlated with other stations. Stations with low strength, such as *Phoenix Park Gate*, *Parnell Street*, and *Irishtown Road*, display temporal similarities with only a small number of other stations (up to 14) and show either no correlation or are negatively correlated with the majority of stations However, these stations may still be of interest for analysis, as they demonstrate unique and potentially distinctive activity patterns.

B.3 Spatio-Temporal Bike Graph Networks (Hourly STBiGN)

Hourly correlation networks provide an analysis of activity patterns on an hourly basis. the timeseries for each station consists of 24 points, each corresponding to one hour of the day. Each timeseries entry is proportional to the total number of trips occurring at a specific hour at the station. As before, edges between nodes are formed based on Pearson correlation coefficients, using a threshold of $T = 0.737$ and resulting in a network density of $D = 0.370$. Figure B.7 illustrates the hourly correlation network, which consists of three distinct communities. Community 13 (purple) contains 38 stations, Community 26 (mint green) includes 40 stations, and Community 77 (red) has 8 stations. The sizes of nodes and edges are proportional to the sum of the correlation coefficients of nodes and edges, respectively. Only the top 5% of edges are displayed in the figure, with node colors indicating their respective clusters. The top 10 most correlated edges are highlighted in red, while the remaining edges are shown in charcoal.

The hourly correlation network detects two large communities, alongside one smaller community. The mint green community concentrates in central Dublin,

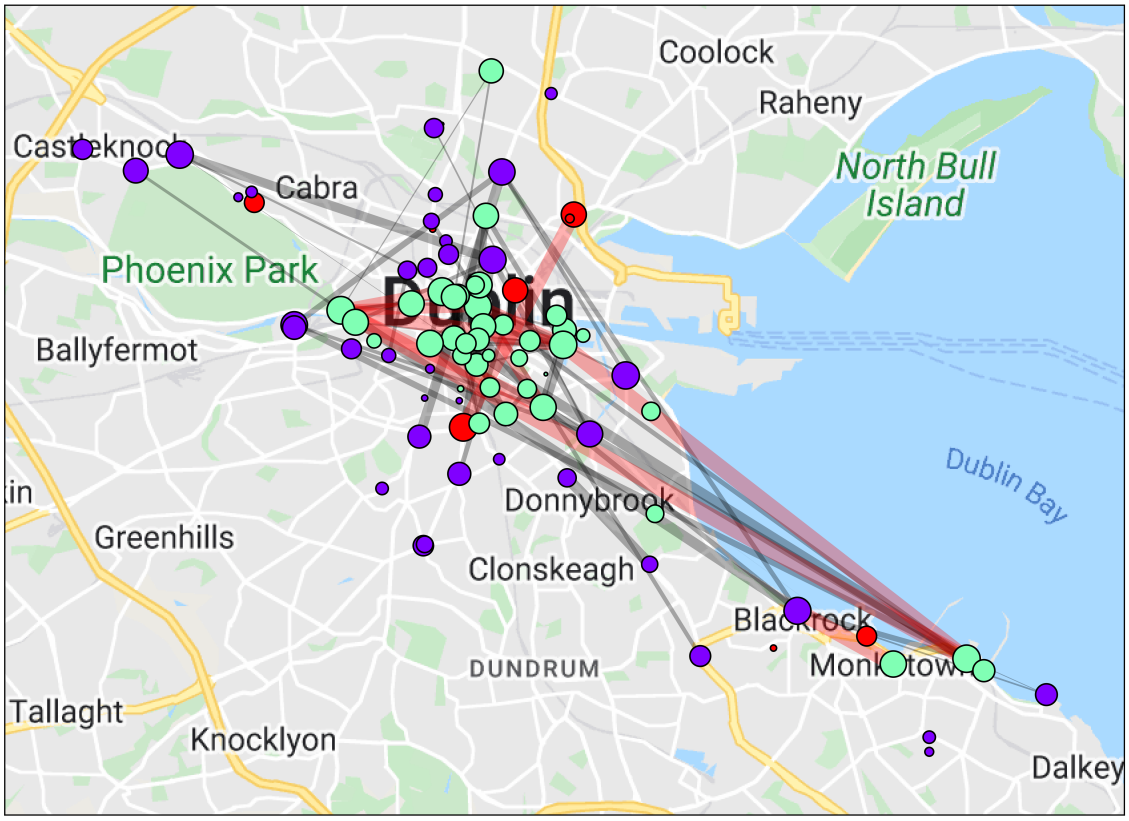


Figure B.7: **Hourly Correlation Network**. Each circle represents a bike station and is sized by its strength. The node colors indicate different communities, with three communities labeled: purple, mint green, and red. Lines represent routes between stations and are sized by their correlation score.

while the purple community surrounds it, primarily located in suburban areas. The small red community is scattered across a wide geographic area. Stations within the central mint green community exhibit the highest strength scores, indicating a well-connected group with strong similarities. However, many of the strongest edges representing the highest correlations are found within the purple community, despite the lower station strengths in this group. This indicates that the activity patterns among purple community stations are well-defined. As observed in the daily networks, stations can be strongly connected even when geographically distant, indicating that remote stations may share underlying characteristics, such as similar commuting behaviors or other social activities.

Figure B.8 plots the average timeseries for the three communities, using the same colors as in Figure B.7. The mint green community shows peak activity during the afternoons and evenings, particularly between 3 p.m. and 7 p.m., with fewer trips

occurring in the mornings. In contrast, the purple community experiences higher activity levels in the mornings and around noon, with a decrease in the afternoon. The small red community provides limited data, except for a noticeable peak around 7 p.m.

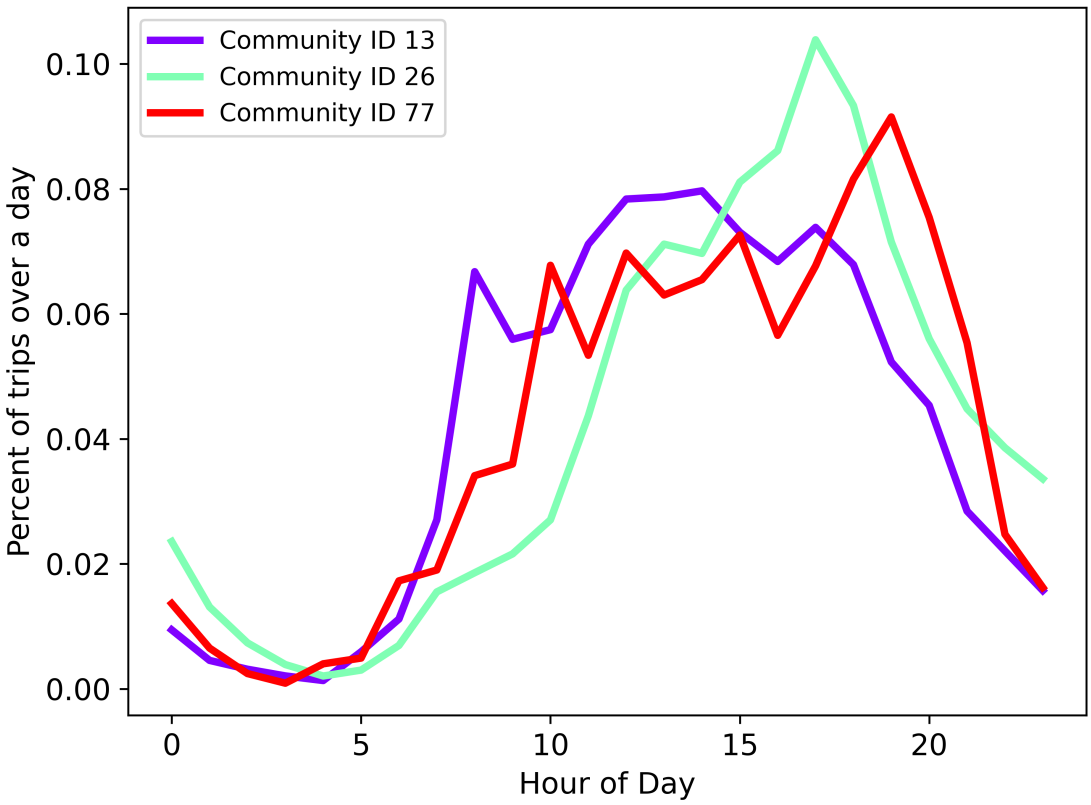


Figure B.8: Timeseries Communities in hourly STBiGNs

Similar to the monthly and daily analysis, Table B.4 displays node strength for the hourly network, with a threshold of $T = 0.65$. Given that all stations follow a similar pattern, characterized by limited activity at night and increased activity during the day, it is unsurprising that even stations with the lowest strength still reach values as high as 55.1, corresponding to an average edge weight of 0.648. This indicates that these stations are significantly correlated with half of the other stations. On the other hand, the stations with the highest strength, such as *Grand Canal Docks* and *Criminal Courts of Justice*, exhibit strong correlations with nearly all other stations. These stations are typically major hubs with high traffic volumes, making it understandable that they share diverse behaviors and patterns with other

stations, as revealed in the fine-grained hourly STBiGNs.

Table B.4: Node (station) Strength in Hourly STBiGNs

Station	**Strength**	**Avg. Cor.**	**Pos**	**Neu**	**Neg**
Grand Canal Docks	75.1	0.883	84	1	0
Irishtown Rd	75.0	0.882	85	0	0
Criminal Courts of Justice	75.0	0.882	83	2	0
...					
Dean Street	62.6	0.737	81	4	0
Clanbrassil St Lower	56.8	0.669	43	42	0
Warehouse	55.1	0.648	45	40	0

Having now generated all three types of STBiGNs using the entire dataset, the degree of heterogeneity observed between the three graphs is both surprising and unexpected. No single station consistently appears in the top 5 across all three networks. Only *Irishtown Road* ranks in the top 5 of two STBiGNs, placing 3rd in the daily network and 2nd in the hourly network. Extending the analysis to the top 10 stations yields similar results—no station appears in all three graphs. Only *Rathmines*, which ranks 2nd in the monthly network and 6th in the daily network and *Dun Laoghaire Dart*, which ranks 3rd in the monthly and 7th in the daily network, appear in two top 10 rankings. Stations exhibit distinct similarity patterns when analyzed at different levels of temporal granularity.

Appendix C

Mosquito ODE system

C.1 Structural identifiability of mosquito system's parameters

Shown in figure C.1. Structural identifiability is achieved by expressing the system as a system of linear equations, the parameters as unknown variables, and then analyzing the reduced row-echelon form of the coefficient matrix, performed separately for each time t. In the figure, identifiable parameters are defined as free parameters which can get arbitrary values. The values are rounded to 6-digit precision, aligning with the PINN's level of precision after training.

C.2 Parameter Sensitivity

In order to identify the most influential parameters within the mosquito dynamical system, we conduct a sensitivity analysis. This involves systematically modifying parameters and observing their effects on the system state. Ten parameters are examined: $\gamma_{Aem}, \gamma_{Ab}, \gamma_{Ao}, f_E, f_P, f_L, f_Ag, m_L, m_P$, and m_A, all of which are listed in Table 6.2. Each parameter was independently adjusted at all time points t by $-10\%, -5\%, +5\%$, and $+10\%$ of its original value, while maintaining all other parameters constant. The system is then simulated under these modified parameters

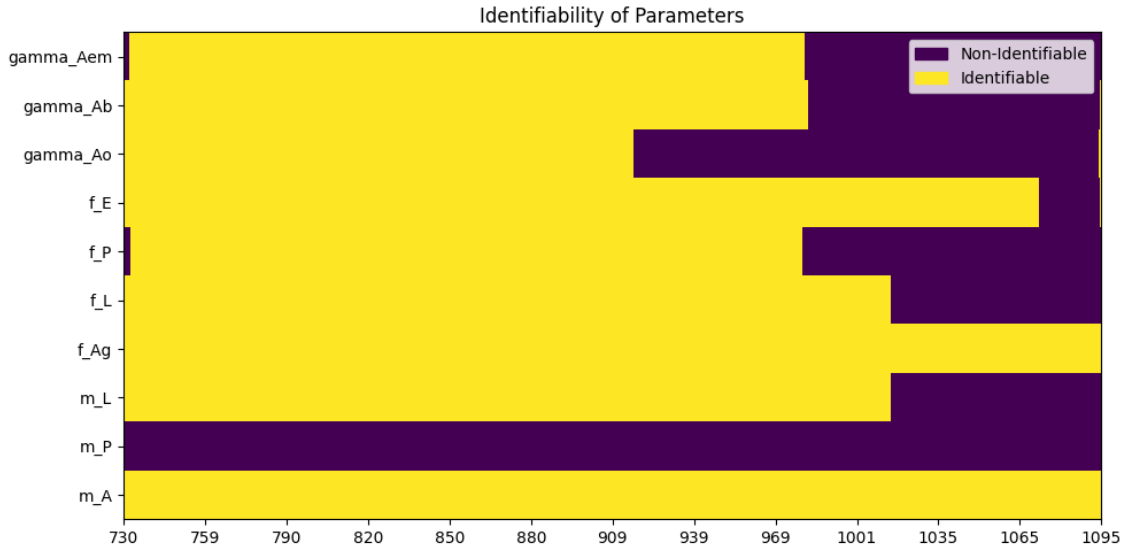


Figure C.1: **Structural identifiability of parameters over time in the Mosquito inverse problem**. The configuration is the same as the experiment in Section 6.4.2 where the temperature is sine-shaped.

using the Python ODE Solver. The root mean squared error (RMSE) is computed between the new $A_{b1} + A_{b2}$ values and those obtained from the unmodified parameter set.

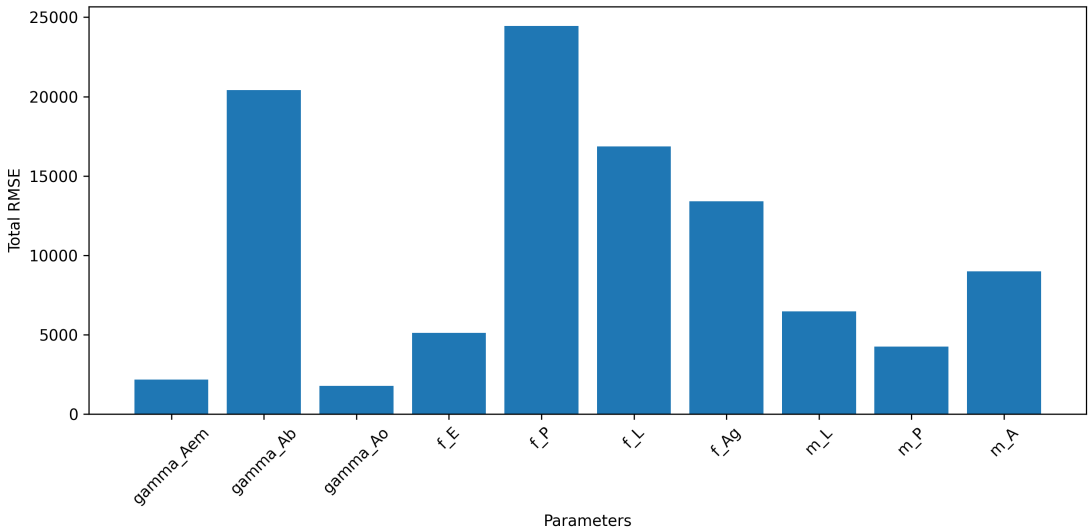


Figure C.2: **Sensitivity of parameters in mosquito dynamical system.**

The average RMSE across the four modification levels for each parameter is depicted in Figure C.2. The analysis reveals that the pupa development rate f_P has the most pronounced effect on the adult blood-seeking mosquito population $A_{b1} + A_{b2}$. The development rate of blood-seeking adults f_{Ab}, which directly influences A_{b1} and

A_{b2} in the system equations, shows a slightly less impact. Parameters f_{Ag} and m_A demonstrate minor effects, while other parameters such as γ_{Aem} and γ_{Ao} exhibit negligible influence on the $A_{b1} + A_{b2}$ quantity.

Bibliography

[1] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[2] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[3] Claude Lemaréchal. "Cauchy and the gradient method". In: *Doc Math Extra* 251.254 (2012), p. 10.

[4] Stephen M Stigler. "Gauss and the invention of least squares". In: *the Annals of Statistics* (1981), pp. 465–474.

[5] Shunichi Amari. "A theory of adaptive pattern classifiers". In: *IEEE Transactions on Electronic Computers* 3 (1967), pp. 299–307.

[6] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[7] Seppo Linnainmaa. "Taylor expansion of the accumulated rounding error". In: *BIT Numerical Mathematics* 16.2 (1976), pp. 146–160.

[8] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *ArXiv* abs/1912.01703 (2019).

[9] Martín Abadi et al. *TensorFlow: A system for large-scale machine learning.* 2016.

[10] Paul Covington, Jay Adams, and Emre Sargin. “Deep Neural Networks for YouTube Recommendations”. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys ’16. Boston, Massachusetts, USA: Association for Computing Machinery, 2016, pp. 191–198. ISBN: 9781450340359.

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “FaceNet: A unified embedding for face recognition and clustering”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015, pp. 815–823.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “ You Only Look Once: Unified, Real-Time Object Detection ”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2016, pp. 779–788.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014.

[14] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. “Attention-Based Models for Speech Recognition”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015.

[15] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. “WaveNet: A Generative Model for Raw Audio”. In: *Speech Synthesis Workshop*. 2016.

[16] OpenAI et al. *GPT-4 Technical Report*. 2024.

[17] Gemini Team et al. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. 2024.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[19] Shahab Shamshirband, Mahdis Fathi, Abdollah Dehzangi, Anthony Theodore Chronopoulos, and Hamid Alinejad-Rokny. "A review on deep learning approaches in healthcare systems: Taxonomies, challenges, and open issues". In: *Journal of Biomedical Informatics* 113 (2021), p. 103627.

[20] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. "Highly accurate protein structure prediction with AlphaFold". In: *nature* 596.7873 (2021), pp. 583–589.

[21] Shuochen Bi and Yufan Lian. "Advanced Portfolio Management in Finance using Deep Learning and Artificial Intelligence Techniques: Enhancing Investment Strategies through Machine Learning Models". In: *Journal of Artificial Intelligence Research* 4.1 (2024), pp. 233–298.

[22] Ishana Attri, Lalit Kumar Awasthi, Teek Parval Sharma, and Priyanka Rathee. "A review of deep learning techniques used in agriculture". In: *Ecological Informatics* 77 (2023), p. 102217. ISSN: 1574-9541.

[23] Veronika Eyring, William D Collins, Pierre Gentine, Elizabeth A Barnes, Marcelo Barreiro, Tom Beucler, Marc Bocquet, Christopher S Bretherton, Hannah M Christensen, Katherine Dagon, et al. "Pushing the frontiers in climate modelling and analysis with machine learning". In: *Nature Climate Change* 14.9 (2024), pp. 916–928.

[24] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. "Code llama: Open foundation models for code". In: *arXiv preprint arXiv:2308.12950* (2023).

[25] Christian Janiesch, Patrick Zschech, and Kai Heinrich. "Machine learning and deep learning". In: *Electronic Markets* 31.3 (2021), pp. 685–695.

[26] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6 (June 2021), pp. 422–440. ISSN: 2522-5820.

[27] Gary Marcus. "Deep Learning: A Critical Appraisal". In: *arXiv preprint arXiv:1801.00631* (2018).

[28] Han Yu, Jiashuo Liu, Xingxuan Zhang, Jiayun Wu, and Peng Cui. "A Survey on Evaluation of Out-of-Distribution Generalization". In: *ArXiv* abs/2403.01874 (2024).

[29] Lei Wu, Zhanxing Zhu, et al. "Towards understanding generalization of deep learning: Perspective of loss landscapes". In: *arXiv preprint arXiv:1706.10239* (2017).

[30] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". In: *Neural Networks* 6.6 (1993), pp. 861–867. ISSN: 0893-6080.

[31] Zijun Cui, Tian Gao, Kartik Talamadupula, and Qiang Ji. "Knowledge-augmented deep learning and its applications: A survey". In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[32] Benjamin D. Haeffele and René Vidal. "Global Optimality in Neural Network Training". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4390–4398.

[33] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. “Graph Neural Networks for Social Recommendation”. In: *CoRR* abs/1902.07243 (2019).

[34] Bing Yu, Haoteng Yin, and Zhanxing Zhu. “Spatio-temporal Graph Convolutional Neural Network: A Deep Learning Framework for Traffic Forecasting”. In: *CoRR* abs/1709.04875 (2017).

[35] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. “Protein Interface Prediction using Graph Convolutional Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[36] Hernan Lira, Luis Martí, and Nayat Sanchez-Pi. “A Graph Neural Network with Spatio-Temporal Attention for Multi-Sources Time Series Data: An Application to Frost Forecast”. In: *Sensors* 22.4 (2022). ISSN: 1424-8220.

[37] Sijie Yan, Yuanjun Xiong, and Dahua Lin. “Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition”. In: *CoRR* abs/1801.07455 (2018).

[38] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *CoRR* abs/1609.02907 (2016).

[39] Georgios Kissas, Yibo Yang, Eileen Hwuang, Walter R. Witschey, John A. Detre, and Paris Perdikaris. “Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks”. In: *Computer Methods in Applied Mechanics and Engineering* 358 (2020), p. 112623. ISSN: 0045-7825.

[40] Tom Beucler, Stephan Rasp, Michael Pritchard, and Pierre Gentine. *Achieving Conservation of Energy in Neural Network Emulators for Climate Modeling*. 2019.

[41] Sifan Wang, Hanwen Wang, and Paris Perdikaris. “On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks”. In: *Computer Methods in Applied Mechanics and Engineering* 384 (2021), p. 113938. ISSN: 0045-7825.

[42] Sifan Wang, Yujun Teng, and Paris Perdikaris. “Understanding and mitigating gradient pathologies in physics-informed neural networks”. In: *ArXiv* abs/2001.04536 (2020).

[43] M. Raissi, P. Perdikaris, and G.E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991.

[44] Mina Petrić. “Modelling the Influence of Meteorological Conditions on Mosquito Vector Population Dynamics (Diptera, Culicidae)”. English. PhD thesis. Belgium: Ghent University, 2020, p. 214. ISBN: 9798505568446.

[45] Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N Balasubramanian. “Neural network attributions: A causal perspective”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 981–990.

[46] R. S. Sokhi et al. “Advances in air quality research – current and emerging challenges”. In: *Atmospheric Chemistry and Physics* 22.7 (2022), pp. 4615–4703.

[47] Qiao Kang, Baiyu Zhang, Yiqi Cao, Xing Song, Xudong Ye, Xixi Li, Hongjing Wu, Yuanzhu Chen, and Bing Chen. “Causal Prior-Embedded Physics-Informed Neural Networks and a Case Study on Metformin Transport in Porous Media”. In: *Water Research* (2024), p. 121985.

[48] Aneta Zebrowska, Karol Jelonek, Sujan Mondal, Marta Gawin, Katarzyna Mrowiec, Piotr Widłak, Theresa Whiteside, and Monika Pietrowska. “Proteomic and metabolomic profiles of T cell-derived exosomes isolated from human plasma”. en. In: *Cells* 11.12 (June 2022), p. 1965.

[49] Joseph Carmicheal, Chihiro Hayashi, Xi Huang, Lei Liu, Yao Lu, Alexey Krasnoslobodtsev, Alexander Lushnikov, Prakash G. Kshirsagar, Asish Patel, Maneesh Jain, Yuri L. Lyubchenko, Yongfeng Lu, Surinder K. Batra, and Sukhwinder Kaur. "Label-free characterization of exosome via surface enhanced Raman spectroscopy for the early detection of pancreatic cancer". In: *Nanomedicine: Nanotechnology, Biology and Medicine* 16 (2019), pp. 88–96. ISSN: 1549-9634.

[50] Hyunku Shin, Hyesun Jeong, Jaena Park, Sunghoi Hong, and Yeonho Choi. "Correlation between Cancerous Exosomes and Protein Markers Based on Surface-Enhanced Raman Spectroscopy (SERS) and Principal Component Analysis (PCA)". In: *ACS Sensors* 3.12 (2018). PMID: 30381940, pp. 2637–2643.

[51] Yangcenzi Xie, Xiaoming Su, Yu Wen, Chao Zheng, and Ming Li. "Artificial Intelligent Label-Free SERS Profiling of Serum Exosomes for Breast Cancer Diagnosis and Postoperative Assessment". In: *Nano Letters* 22.19 (2022). PMID: 36149810, pp. 7910–7918.

[52] Hyunku Shin, Seunghyun Oh, Soonwoo Hong, Minsung Kang, Daehyeon Kang, Yong-gu Ji, Byeong Hyeon Choi, Ka-Won Kang, Hyesun Jeong, Yong Park, Sunghoi Hong, Hyun Koo Kim, and Yeonho Choi. "Early-Stage Lung Cancer Diagnosis by Deep Learning-Based Spectroscopic Analysis of Circulating Exosomes". In: *ACS Nano* 14.5 (2020). PMID: 32286793, pp. 5435–5444.

[53] Lauren E Jamieson, Steven M Asiala, Kirsten Gracie, Karen Faulds, and Duncan Graham. "Bioanalytical measurements enabled by surface-enhanced Raman scattering (SERS) probes". In: *Annual Review of Analytical Chemistry* 10.1 (2017), pp. 415–437.

[54] Cheng Zong, Mengxi Xu, Li-Jia Xu, Ting Wei, Xin Ma, Xiao-Shan Zheng, Ren Hu, and Bin Ren. "Surface-enhanced Raman spectroscopy for bioanalysis: reliability and challenges". In: *Chemical reviews* 118.10 (2018), pp. 4946–4980.

[55] Yacob Pinchevsky, Neil Butkow, Frederick J Raal, Tobias Chirwa, and Alan Rothberg. "Demographic and clinical factors associated with development of type 2 diabetes: A review of the literature". en. In: *Int. J. Gen. Med.* 13 (Mar. 2020), pp. 121–129.

[56] Kirsty M. Danielson and Saumya Das. "Extracellular Vesicles in Heart Disease: Excitement for the Future?" In: *Journal of Circulating Biomarkers* 2.1 (Jan. 2014).

[57] Cristian Osgnach and Pietro Enrico di Prampero. "Metabolic power in team sports-Part 2: aerobic and anaerobic energy yields". In: *International journal of sports medicine* 39.08 (2018), pp. 588–595.

[58] Robin T Thorpe, Anthony J Strudwick, Martin Buchheit, Greg Atkinson, Barry Drust, and Warren Gregson. "Monitoring fatigue during the in-season competitive phase in elite soccer players". In: *International journal of sports physiology and performance* 10.8 (2015), pp. 958–964.

[59] Shona L Halson. "Monitoring training load to understand fatigue in athletes". en. In: *Sports Med.* 44 Suppl 2.S2 (Nov. 2014), S139–47.

[60] Janina Helwig, Janik Diels, Mareike Röll, Hubert Mahler, Albert Gollhofer, Kai Roecker, and Steffen Willwacher. "Relationships between External, Wearable Sensor-Based, and Internal Parameters: A Systematic Review". en. In: *Sensors* 23.2 (Jan. 2023). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 827. ISSN: 1424-8220.

[61] Martin Buchheit and Ben Michael Simpson. "Player-tracking technology: half-full or half-empty glass?" In: *International journal of sports physiology and performance* 12.s2 (2017), S2–35.

[62] Niels Jensby Nedergaard, Uwe Kersting, and Mark Lake. "Using accelerometry to quantify deceleration during a high-intensity soccer turning manoeuvre". In: *Journal of sports sciences* 32.20 (2014), pp. 1897–1905.

[63] Henri Vähä-Ypyä, Jakob Bretterhofer, Pauliina Husu, Jana Windhaber, Tommi Vasankari, Sylvia Titze, and Harri Sievänen. "Performance of Different Accelerometry-Based Metrics to Estimate Oxygen Consumption during Track and Treadmill Locomotion over a Wide Intensity Range". en. In: *Sensors* 23.11 (Jan. 2023). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 5073. ISSN: 1424-8220.

[64] Carlos D Gómez-Carmona, José Pino-Ortega, Braulio Sánchez-Ureña, Sergio J Ibáñez, and Daniel Rojas-Valverde. "Accelerometry-based external load indicators in sport: too many options, same practical outcome?" In: *International Journal of Environmental Research and Public Health* 16.24 (2019), p. 5101.

[65] Thomas Beltrame, Robert Amelard, Alexander Wong, and Richard Hughson. "Prediction of oxygen uptake dynamics by machine learning analysis of wearable sensors during activities of daily living". In: *Scientific Reports* 7 (Apr. 2017), p. 45738.

[66] Pavel Davidson, Huy Trinh, Sakari Vekki, and Philipp Müller. "Surrogate Modelling for Oxygen Uptake Prediction Using LSTM Neural Network". en. In: *Sensors* 23.4 (Jan. 2023). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 2249. ISSN: 1424-8220.

[67] Zhao Wang, Qiang Zhang, Ke Lan, Zhicheng Yang, Xiaolin Gao, Anshuo Wu, Yi Xin, and Zhengbo Zhang. "Enhancing instantaneous oxygen uptake estimation by non-linear model using cardio-pulmonary physiological and motion signalsEnhancing instantaneous oxygen uptake estimation by non-linear model using cardio-pulmonary physiological and motion signals". In: *Frontiers in Physiology* 13 (2022). ISSN: 1664-042X.

[68] Thomas Beltrame, Robert Amelard, Rodrigo Villar, Mohammad J. Shafiee, Alexander Wong, and Richard L. Hughson. "Estimating oxygen uptake and energy expenditure during treadmill walking by neural network analysis of

easy-to-obtain inputs". en. In: *Journal of Applied Physiology* 121.5 (Nov. 2016). Number: 5, pp. 1226–1233. ISSN: 8750-7587, 1522-1601.

[69] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. "Temporal convolutional networks for action segmentation and detection". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 156–165.

[70] Robert Amelard, Eric T. Hedge, and Richard L. Hughson. "Temporal convolutional networks predict dynamic oxygen uptake response from wearable sensors across exercise intensities". en. In: *npj Digital Medicine* 4.1 (Nov. 2021). Number: 1 Publisher: Nature Publishing Group, pp. 1–8. ISSN: 2398-6352.

[71] Eric T. Hedge, Robert Amelard, and Richard L. Hughson. "Prediction of oxygen uptake kinetics during heavy-intensity cycling exercise by machine-learning analysis". In: *Journal of Applied Physiology* (May 2023). Publisher: American Physiological Society. ISSN: 8750-7587.

[72] Andrea Zignoli, Alessandro Fornasiero, Matteo Ragni, Barbara Pellegrini, Federico Schena, Francesco Biral, and Paul B. Laursen. "Estimating an individual's oxygen uptake during cycling exercise with a recurrent neural network trained from easy-to-obtain inputs: A pilot study". en. In: *PLOS ONE* 15.3 (Mar. 2020). Publisher: Public Library of Science, e0229466. ISSN: 1932-6203.

[73] Akrati Saxena, Pratishtha Saxena, Harita Reddy, and Ralucca Gera. "A Survey on Studying the Social Networks of Students". In: *CoRR* abs/1909.05079 (2019).

[74] Louise Ryan and Alessio D'Angelo. "Changing times: Migrants' social network analysis and the challenges of longitudinal research". In: *Social Networks* 53 (2018). The missing link: Social network analysis in migration and transnationalism, pp. 148–158. ISSN: 0378-8733.

[75] Luis E.C. Rocha. “Dynamics of air transport networks: A review from a complex systems perspective”. In: *Chinese Journal of Aeronautics* 30.2 (2017), pp. 469–478. ISSN: 1000-9361.

[76] Nam Huynh and Johan Barthelemy. “A comparative study of topological analysis and temporal network analysis of a public transport system”. In: *International Journal of Transportation Science and Technology* (2021). ISSN: 2046-0430.

[77] Jingfang Fan, Jun Meng, Yosef Ashkenazy, Shlomo Havlin, and Hans Joachim Schellnhuber. “Network analysis reveals strongly localized impacts of El Niño”. In: *Proceedings of the National Academy of Sciences* 114.29 (2017), pp. 7543–7548.

[78] Niklas Boers, Bedartha Goswami, Aljoscha Rheinwalt, Bodo Bookhagen, Brian Hoskins, and Jürgen Kurths. “Complex networks reveal global pattern of extreme-rainfall teleconnections”. In: *Nature* 566.7744 (Feb. 2019), pp. 373–377. ISSN: 1476-4687.

[79] Xingyi Li, Wenkai Li, Min Zeng, Ruiqing Zheng, and Min Li. “Network-based methods for predicting essential genes or proteins: a survey”. In: *Briefings in Bioinformatics* 21.2 (Feb. 2019), pp. 566–583. ISSN: 1477-4054.

[80] Yuhui Du, Zening Fu, and Vince D. Calhoun. “Classification and Prediction of Brain Disorders Using Functional Connectivity: Promising but Challenging”. eng. In: *Frontiers in neuroscience* 12 (Aug. 2018). PMC6088208[pmcid], pp. 525–525. ISSN: 1662-4548.

[81] Paola Valsasina, Milagros Hidalgo de la Cruz, Massimo Filippi, and Maria A. Rocca. “Characterizing Rapid Fluctuations of Resting State Functional Connectivity in Demyelinating, Neurodegenerative, and Psychiatric Conditions: From Static to Time-Varying Analysis”. eng. In: *Frontiers in neuroscience* 13 (July 2019). PMC6636554[pmcid], pp. 618–618. ISSN: 1662-4548.

[82] Lu Bai, Jianzhou Wang, Xuejiao Ma, and Haiyan Lu. “Air pollution forecasts: An overview”. In: *International journal of environmental research and public health* 15.4 (2018), p. 780.

[83] Marilena Kampa and Elias Castanas. “Human health effects of air pollution”. In: *Environmental pollution* 151.2 (2008), pp. 362–367.

[84] Sumita Gulati, Anshul Bansal, Ashok Pal, Nitin Mittal, Abhishek Sharma, and Fikreselam Gared. “Estimating PM2. 5 utilizing multiple linear regression and ANN techniques”. In: *Scientific Reports* 13.1 (2023), p. 22578.

[85] Mahanijah Md Kamal, Rozita Jailani, and Ruhizan Liza Ahmad Shauri. “Prediction of Ambient Air Quality Based on Neural Network Technique”. In: *2006 4th Student Conference on Research and Development.* 2006, pp. 115–119.

[86] Davor Z. Antanasijević, Viktor V. Pocajt, Dragan S. Povrenović, Mirjana Đ. Ristić, and Aleksandra A. Perić-Grujić. “PM10 emission forecasting using artificial neural networks and genetic algorithm input variable optimization”. In: *Science of The Total Environment* 443 (2013), pp. 511–519. ISSN: 0048-9697.

[87] Jorge Loy-Benitez, Paulina Vilela, Qian Li, and ChangKyoo Yoo. “Sequential prediction of quantitative health risk assessment for the fine particulate matter in an underground facility using deep recurrent neural networks”. In: *Ecotoxicology and Environmental Safety* 169 (2019), pp. 316–324. ISSN: 0147-6513.

[88] Xingcheng Lu, Yu Hin Sha, Zhenning Li, Yeqi Huang, Wanying Chen, Duohong Chen, Jin Shen, Yiang Chen, and Jimmy C.H. Fung. “Development and application of a hybrid long-short term memory – three dimensional variational technique for the improvement of PM2.5 forecasting”. In: *Science of The Total Environment* 770 (2021), p. 144221. ISSN: 0048-9697.

[89] Yijun Lin, Nikhit Mago, Yu Gao, Yaguang Li, Yao-Yi Chiang, Cyrus Shahabi, and José Luis Ambite. "Exploiting Spatiotemporal Patterns for Accurate Air Quality Forecasting Using Deep Learning". In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '18. Seattle, Washington: Association for Computing Machinery, 2018, pp. 359–368. ISBN: 9781450358897.

[90] Yanlin Qi, Qi Li, Hamed Karimian, and Di Liu. "A hybrid model for spatiotemporal forecasting of PM2.5 based on graph convolutional neural network and long short-term memory". In: *Science of The Total Environment* 664 (2019), pp. 1–10. ISSN: 0048-9697.

[91] Waddah Saeed and Christian Omlin. "Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities". In: *Knowledge-Based Systems* 263 (2023), p. 110273. ISSN: 0950-7051.

[92] Chayan Kumar Banerjee, Kien Nguyen, Clinton Fookes, and George E. Karniadakis. "Physics-Informed Computer Vision: A Review and Perspectives". In: *ArXiv* abs/2305.18035 (2023).

[93] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. "Physics-informed neural networks (PINNs) for fluid mechanics: a review". In: *Acta Mechanica Sinica* 37 (2021), pp. 1727–1738.

[94] Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. *Physics-Informed Machine Learning: A Survey on Problems, Methods and Applications*. 2023.

[95] Chayan Kumar Banerjee, Kien Nguyen, Clinton Fookes, and Maziar Raissi. "A Survey on Physics Informed Reinforcement Learning: Review and Open Problems". In: *ArXiv* abs/2309.01909 (2023).

[96] Alireza Yazdani, Lu Lu, Maziar Raissi, and George Em Karniadakis. "Systems biology informed deep learning for inferring parameters and hidden dynamics". In: *PLOS Computational Biology* 16.11 (Nov. 2020), pp. 1–19.

[97] Han Gao, Luning Sun, and Jian-Xun Wang. "PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain". In: *Journal of Computational Physics* 428 (Mar. 2021), p. 110079. ISSN: 0021-9991.

[98] Pu Ren, Chengping Rao, Yang Liu, Jian-Xun Wang, and Hao Sun. "PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs". In: *Computer Methods in Applied Mechanics and Engineering* 389 (2022), p. 114399. ISSN: 0045-7825.

[99] Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks". In: *Journal of Computational Physics* 404 (2020), p. 109136. ISSN: 0021-9991.

[100] Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. "Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476.2239 (July 2020), p. 20200334. ISSN: 1471-2946.

[101] Suryanarayana Maddu, Dominik Sturm, Christian L Müller, and Ivo F Sbalzarini. "Inverse Dirichlet weighting enables reliable training of physics informed neural networks". In: *Machine Learning: Science and Technology* 3.1 (Feb. 2022), p. 015026.

[102] Sifan Wang, Xinling Yu, and Paris Perdikaris. "When and why PINNs fail to train: A neural tangent kernel perspective". In: *Journal of Computational Physics* 449 (2022), p. 110768. ISSN: 0021-9991.

[103] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. "DeepXDE: A Deep Learning Library for Solving Differential Equations". In: *SIAM Review* 63.1 (2021), pp. 208–228.

[104] Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. "A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks". In: *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115671. ISSN: 0045-7825.

[105] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. "Efficient training of physics-informed neural networks via importance sampling". In: *Computer-Aided Civil and Infrastructure Engineering* 36.8 (2021), pp. 962–977.

[106] Kejun Tang, Xiaoliang Wan, and Chao Yang. "DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations". In: *Journal of Computational Physics* 476 (2023), p. 111868. ISSN: 0021-9991.

[107] Colby L. Wight and Jia Zhao. "Solving Allen-Cahn and Cahn-Hilliard Equations Using the Adaptive Physics Informed Neural Networks". In: *Communications in Computational Physics* 29.3 (2021), pp. 930–954. ISSN: 1991-7120.

[108] Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, and Michael W. Mahoney. *Characterizing possible failure modes in physics-informed neural networks*. 2021.

[109] Revanth Mattey and Susanta Ghosh. "A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations". In: *Computer Methods in Applied Mechanics and Engineering* 390 (2022), p. 114474. ISSN: 0045-7825.

[110] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. "Respecting causality is all you need for training physics-informed neural networks". In: *ArXiv* abs/2203.07404 (2022).

[111] Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems". In: *Computer Methods in Applied Mechanics and Engineering* 365 (2020), p. 113028. ISSN: 0045-7825.

[112] Ameya D Jagtap and George Em Karniadakis. “Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations”. In: *Communications in Computational Physics* 28.5 (2020), pp. 2002–2041.

[113] Benjamin Moseley, A. Markham, and Tarje Nissen-Meyer. “Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations”. In: *Advances in Computational Mathematics* 49 (2021), pp. 1–39.

[114] E Scott Krayenhoff and James A Voogt. “A microscale three-dimensional urban energy balance model for studying surface temperatures”. In: *Boundary-Layer Meteorology* 123 (2007), pp. 433–461.

[115] Adam J Kucharski, Timothy W Russell, Charlie Diamond, Yang Liu, John Edmunds, Sebastian Funk, Rosalind M Eggo, Fiona Sun, Mark Jit, James D Munday, et al. “Early dynamics of transmission and control of COVID-19: a mathematical modelling study”. In: *The lancet infectious diseases* 20.5 (2020), pp. 553–558.

[116] Abicumaran Uthamacumaran and Hector Zenil. “A review of mathematical and computational methods in cancer dynamics”. In: *Frontiers in oncology* 12 (2022), p. 850731.

[117] Omar Ghattas and Karen Willcox. “Learning physics-based models from data: perspectives from inverse problems and model reduction”. In: *Acta Numerica* 30 (2021), pp. 445–554.

[118] Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. “Physics-informed neural networks for inverse problems in nano-optics and metamaterials”. In: *Opt. Express* 28.8 (Apr. 2020), pp. 11618–11633.

[119] A. M. Tartakovsky, C. Ortiz Marrero, Paris Perdikaris, G. D. Tartakovsky, and D. Barajas-Solano. “Physics-Informed Deep Neural Networks for Learning Parameters and Constitutive Relationships in Subsurface Flow Problems”. In:

Water Resources Research 56.5 (2020). e2019WR026731 10.1029/2019WR026731, e2019WR026731.

[120] Xin-Yu Guo and Sheng-En Fang. "Structural parameter identification using physics-informed neural networks". In: *Measurement* 220 (2023), p. 113334. ISSN: 0263-2241.

[121] Chen Xu, Ba Trung Cao, Yong Yuan, and Günther Meschke. "Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios". In: *Computer Methods in Applied Mechanics and Engineering* 405 (2023), p. 115852. ISSN: 0045-7825.

[122] Nazanin Ahmadi Daryakenari, Mario De Florio, Khemraj Shukla, and George Em Karniadakis. "AI-Aristotle: A physics-informed framework for systems biology gray-box identification". In: *PLOS Computational Biology* 20.3 (Mar. 2024), pp. 1–33.

[123] World Health Organisation. *Chikungunya*. en. https://www.who.int/health-topics/chikungunya. Accessed: 2024-05-08.

[124] World Health Organisation. *Dengue and severe dengue*. en. https://www.who.int/news-room/fact-sheets/detail/dengue-and-severe-dengue. Accessed: 2024-05-08.

[125] World Health Organisation. *Fact sheet about malaria*. en. https://www.who.int/news-room/fact-sheets/detail/malaria. Accessed: 2024-05-08.

[126] World Health Organisation. *Zika virus disease*. en. https://www.who.int/health-topics/zika-virus-disease. Accessed: 2024-05-08.

[127] World Health Organisation. *West Nile virus*. en. https://www.who.int/news-room/fact-sheets/detail/west-nile-virus. Accessed: 2024-05-08.

[128] Priscilla Cailly, Annelise Tran, Thomas Balenghien, Grégory L'Ambert, Céline Toty, and Pauline Ezanno. "A climate-driven abundance model to assess mosquito control strategies". en. In: *Ecological Modelling* 227 (Feb. 2012), pp. 7–17. ISSN: 0304-3800.

[129] Richard A. Erickson, Steven M. Presley, Linda JS Allen, Kevin R. Long, and Stephen B. Cox. "A stage-structured, Aedes albopictus population model". In: *Ecological Modelling* 221.9 (2010), pp. 1273–1282.

[130] Giovanni Marini, Daniele Arnoldi, Frederic Baldacchino, Gioia Capelli, Giorgio Guzzetta, Stefano Merler, Fabrizio Montarsi, Annapaola Rizzoli, and Roberto Rosà. "First report of the influence of temperature on the bionomics and population dynamics of Aedes koreicus, a new invasive alien species in Europe." eng. In: *Parasites & vectors* 12.1 (Nov. 2019), p. 524. ISSN: 1756-3305.

[131] Lizhong Qiang, Bin-Guo Wang, and Xiao-Qiang Zhao. "A Stage-Structured Population Model with Time-Dependent Delay in an Almost Periodic Environment". en. In: *Journal of Dynamics and Differential Equations* 34.1 (Mar. 2022), pp. 341–364. ISSN: 1572-9222.

[132] Marcelo Otero, Hernán G. Solari, and Nicolás Schweigmann. "A stochastic population dynamics model for Aedes aegypti: formulation and application to a city with temperate climate". In: *Bulletin of mathematical biology* 68.8 (2006). Publisher: Springer, pp. 1945–1974.

[133] Ting-Wu Chuang, Edward L. Ionides, Randall G. Knepper, William W. Stanuszek, Edward D. Walker, and Mark L. Wilson. "Cross-correlation map analyses show weather variation influences on mosquito abundance patterns in Saginaw County, Michigan, 1989–2005". In: *Journal of medical entomology* 49.4 (2012). Publisher: Oxford University Press Oxford, UK, pp. 851–858.

[134] Collin B. Edwards and Elizabeth E. Crone. "Estimating abundance and phenology from transect count data with GLMs". en. In: *Oikos* 130.8 (Aug. 2021), pp. 1335–1345. ISSN: 0030-1299, 1600-0706.

[135] Olugbenga O. Oluwagbemi, Christen M. Fornadel, Ezekiel F. Adebiyi, Douglas E. Norris, and Jason L. Rasgon. "ANOSPEX: a stochastic, spatially explicit model for studying Anopheles metapopulation dynamics". In: *PloS one* 8.7 (2013). Publisher: Public Library of Science, e68040.

[136] Antoine Guisan and Wilfried Thuiller. “Predicting species distribution: offering more than simple habitat models”. In: *Ecology letters* 8.9 (2005). Publisher: Wiley Online Library, pp. 993–1009.

[137] Daniele Da Re, Wim Van Bortel, Friederike Reuss, Ruth Müller, Sebastien Boyer, Fabrizio Montarsi, Silvia Ciocchetta, Daniele Arnoldi, Giovanni Marini, Annapaola Rizzoli, Gregory L’Ambert, Guillaume Lacour, Constantianus J. M. Koenraadt, Sophie O. Vanwambeke, and Matteo Marcantonio. “dynamAedes: a unified modelling framework for invasive Aedes mosquitoes”. In: *Parasites & Vectors* 15.1 (Nov. 2022), p. 414. ISSN: 1756-3305.

[138] Daniele Da Re, Giovanni Marini, Carmelo Bonannella, Fabrizio Laurini, Mattia Manica, Nikoleta Anicic, Alessandro Albieri, Paola Angelini, Daniele Arnoldi, and Federica Bertola. *Inferring the seasonal dynamics and abundance of an invasive species using a spatio-temporal stacked machine learning model*. Publisher: EcoEvoRxiv. 2023.

[139] Ugur Parlatan, Mehmet Ozgun Ozen, Ibrahim Kecoglu, Batuhan Koyuncu, Hulya Torun, Davod Khalafkhany, Irem Loc, Mehmet Giray Ogut, Fatih Inci, Demir Akin, Ihsan Solaroglu, Nesrin Ozoren, Mehmet Burcin Unlu, and Utkan Demirci. “Label-Free Identification of Exosomes using Raman Spectroscopy and Machine Learning”. In: *Small* 19.9 (2023), p. 2205519.

[140] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 770–778.

[141] Samuel L. Wilcox, Ryan M. Broxterman, and Thomas J. Barstow. “Constructing quasi-linear Vo2 responses from nonlinear parameters”. In: *Journal of Applied Physiology* 120.2 (2016). PMID: 26565018, pp. 121–129.

[142] Andrew M. Jones and Mark Burnley. “Oxygen uptake kinetics: an underappreciated determinant of exercise performance.” In: *International journal of sports physiology and performance* 4 4 (2009), pp. 524–32.

[143] J. R. Stirling, M. S. Zakynthinaki, and V. Billat. "Modeling and Analysis of the Effect of Training on VO_2 Kinetics and Anaerobic Capacity". In: *Bulletin of Mathematical Biology* 70.5 (July 2008), pp. 1348–1370. ISSN: 1522-9602.

[144] Andrea Zignoli, Alessandro Fornasiero, Enrico Bertolazzi, Barbara Pellegrini, Federico Schena, Francesco Biral, and Paul B. Laursen. "State-of-the art concepts and future directions in modelling oxygen consumption and lactate concentration in cycling exercise". en. In: *Sport Sciences for Health* 15.2 (Aug. 2019). Number: 2, pp. 295–310. ISSN: 1824-7490, 1825-1234.

[145] Maria Cecília Moraes Frade, Thomas Beltrame, Mariana de Oliveira Gois, Allan Pinto, Silvia Cristina Garcia de Moura Tonello, Ricardo da Silva Torres, and Aparecida Maria Catai. "Toward characterizing cardiovascular fitness using machine learning based on unobtrusive data". In: *PLOS ONE* 18.3 (Mar. 2023), pp. 1–18.

[146] Pengfei Chang, Cenyi Wang, Yiyan Chen, Guodong Wang, and Aming Lu. "Identification of runner fatigue stages based on inertial sensors and deep learning". In: *Frontiers in Bioengineering and Biotechnology* 11 (2023). ISSN: 2296-4185.

[147] Hui Zhang, Chengxiang Zhuge, Jianmin Jia, Baiying Shi, and Wei Wang. "Green travel mobility of dockless bike-sharing based on trip data in big cities: A spatial network analysis". In: *Journal of Cleaner Production* 313 (2021), p. 127930. ISSN: 0959-6526.

[148] Yuanxuan Yang, Alison Heppenstall, Andy Turner, and Alexis Comber. "A spatiotemporal and graph-based analysis of dockless bike sharing patterns to understand urban flows over the last mile". In: *Computers, Environment and Urban Systems* 77 (2019), p. 101361. ISSN: 0198-9715.

[149] Il-Jung Seo and Jaehee Cho. "Structural Features of Public Bicycle Transportation Networks over Times of the Day: The Case of Seoul Public Bicycle".

In: *2022 IEEE International Conference on Big Data and Smart Computing (BigComp)*. 2022, pp. 5–8.

[150] XiaoYing Shi, Yang Wang, Fanshun Lv, Wenhui Liu, Dewen Seng, and Fei Lin. "Finding communities in bicycle sharing system". In: *Journal of Visualization* 22.6 (Dec. 2019), pp. 1177–1192. ISSN: 1875-8975.

[151] Sérgio F. A. Batista, Mostafa Ameli, and Mónica Menéndez. "On the Characterization of Eco-Friendly Paths for Regional Networks". In: *IEEE Open Journal of Intelligent Transportation Systems* 4 (2023), pp. 204–215.

[152] Lei Lin, Zhengbing He, and Srinivas Peeta. "Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach". In: *Transportation Research Part C: Emerging Technologies* 97 (2018), pp. 258–276. ISSN: 0968-090X.

[153] Zahra Ghandeharioun and Anastasios Kouvelas. "Link Travel Time Estimation for Arterial Networks Based on Sparse GPS Data and Considering Progressive Correlations". In: *IEEE Open Journal of Intelligent Transportation Systems* 3 (2022), pp. 679–694.

[154] Jianmin Jia, Chunsheng Liu, Xiaohan Wang, Hui Zhang, and Yan Xiao. "Understanding bike-sharing mobility patterns in response to the COVID-19 pandemic". In: *Cities* 142 (2023), p. 104554. ISSN: 0264-2751.

[155] Pierre Borgnat, Celine Robardet, Jean-Baptiste Rouquier, Patrice Abry, Patrick Flandrin, and Eric Fleury. "Shared Bicycles in a City: A Signal Processing and Data Analysis Perspective". In: *Advances in Complex Systems* 14 (June 2011).

[156] Martin Zaltz Austwick, Oliver O'Brien, Emanuele Strano, and Matheus Viana. "The Structure of Spatial Networks and Communities in Bicycle Sharing Systems". In: *PLOS ONE* 8.9 (Sept. 2013), pp. 1–17.

[157] Yi Yao, Yifang Zhang, Lixin Tian, Nianxing Zhou, Zhilin Li, and Minggang Wang. "Analysis of Network Structure of Urban Bike-Sharing System: A Case Study Based on Real-Time Data of a Public Bicycle System". In: *Sustainability* 11.19 (2019). ISSN: 2071-1050.

[158] Hao Lu, Mahantesh Halappanavar, and Ananth Kalyanaraman. "Parallel heuristics for scalable community detection". In: *Parallel Computing* 47 (2015), pp. 19–37. ISSN: 0167-8191.

[159] Bo Zhang, Yi Rong, Ruihan Yong, Dongming Qin, Maozhen Li, Guojian Zou, and Jianguo Pan. "Deep learning for air pollutant concentration prediction: A review". In: *Atmospheric Environment* 290 (2022), p. 119347. ISSN: 1352-2310.

[160] Bernardo S. Beckerman, Michael Jerrett, Randall V. Martin, Aaron van Donkelaar, Zev Ross, and Richard T. Burnett. "Application of the deletion/substitution/addition algorithm to selecting land use regression models for interpolating air pollution measurements in California". In: *Atmospheric Environment* 77 (2013), pp. 172–177. ISSN: 1352-2310.

[161] Zhong-hua Li and Jun Yang. "PM-25 forecasting use reconstruct phase space LS-SVM". In: *2010 The 2nd Conference on Environmental Science and Information Application Technology*. Vol. 1. 2010, pp. 143–146.

[162] P.J. García Nieto, F. Sánchez Lasheras, E. García-Gonzalo, and F.J. de Cos Juez. "PM10 concentration forecasting in the metropolitan area of Oviedo (Northern Spain) using models based on SVM, MLP, VARMA and ARIMA: A case study". In: *Science of The Total Environment* 621 (2018), pp. 753–761. ISSN: 0048-9697.

[163] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667.

[164] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[165] Yann LeCun and Yoshua Bengio. “Convolutional networks for images, speech, and time series”. In: *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258. ISBN: 0262511029.

[166] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *ArXiv* 1409 (Sept. 2014).

[167] Petar Veličkovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. *Graph Attention Networks*. 2017.

[168] Jiachen Zhao, Fang Deng, Yeyun Cai, and Jie Chen. “Long short-term memory - Fully connected (LSTM-FC) neural network for PM2.5 concentration prediction”. In: *Chemosphere* 220 (2019), pp. 486–492. ISSN: 0045-6535.

[169] Hong-Wei Wang, Xiao-Bing Li, Dongsheng Wang, Juanhao Zhao, Hong-di He, and Zhong-Ren Peng. “Regional prediction of ground-level ozone using a hybrid sequence-to-sequence deep learning approach”. In: *Journal of Cleaner Production* 253 (2020), p. 119841. ISSN: 0959-6526.

[170] Bo Zhang, Guojian Zou, Dongming Qin, Yunjie Lu, Yupeng Jin, and Hui Wang. “A novel Encoder-Decoder model based on read-first LSTM for air pollutant prediction”. In: *Science of The Total Environment* 765 (2021), p. 144507. ISSN: 0048-9697.

[171] Ricardo Navares and José L. Aznarte. “Predicting air quality with deep learning LSTM: Towards comprehensive models”. In: *Ecological Informatics* 55 (2020), p. 101019. ISSN: 1574-9541.

[172] Xiang Li, Ling Peng, Xiaojing Yao, Shaolong Cui, Yuan Hu, Chengzeng You, and Tianhe Chi. “Long short-term memory neural network for air pollutant concentration predictions: Method development and evaluation”. In: *Environmental Pollution* 231 (2017), pp. 997–1004. ISSN: 0269-7491.

[173] Xi Gao and Weide Li. “A graph-based LSTM model for PM2.5 forecasting”. In: *Atmospheric Pollution Research* 12.9 (2021), p. 101150. ISSN: 1309-1042.

[174] Hongye Zhou, Feng Zhang, Zhenhong Du, and Renyi Liu. "A theory-guided graph networks based PM2.5 forecasting method". In: *Environmental Pollution* 293 (2022), p. 118569. ISSN: 0269-7491.

[175] Jiahui Xu, Ling Chen, Mingqi Lv, Chaoqun Zhan, Sanjian Chen, and Jian Chang. "HighAir: A Hierarchical Graph Neural Network-Based Air Quality Forecasting Method". In: *CoRR* abs/2101.04264 (2021).

[176] Pengcheng Jia, Nianwen Cao, and Shaobo Yang. "Real-time hourly ozone prediction system for Yangtze River Delta area using attention based on a sequence to sequence model". In: *Atmospheric Environment* 244 (2021), p. 117917. ISSN: 1352-2310.

[177] Xin-Yu Tu, Bo Zhang, Yu-Peng Jin, Guo-Jian Zou, Jian-Guo Pan, and Mao-Zhen Li. "Longer Time Span Air Pollution Prediction: The Attention and Autoencoder Hybrid Learning Model". In: *Mathematical Problems in Engineering* 2021 (June 2021), p. 5515103. ISSN: 1024-123X.

[178] Bo Zhang, Ziyao Geng, Hanwen Zhang, and Jianguo Pan. "Densely connected convolutional networks with attention long short-term memory for estimating PM2.5 values from images". In: *Journal of Cleaner Production* 333 (2022), p. 130101. ISSN: 0959-6526.

[179] Yu Huang, Josh Jia-Ching Ying, and Vincent S. Tseng. "Spatio-attention embedded recurrent neural network for air quality prediction". In: *Knowledge-Based Systems* 233 (2021), p. 107416. ISSN: 0950-7051.

[180] Jiaqi Zhu, Fang Deng, Jiachen Zhao, and Hao Zheng. "Attention-based parallel networks (APNet) for PM2.5 spatiotemporal prediction". In: *Science of The Total Environment* 769 (2021), p. 145082. ISSN: 0048-9697.

[181] Hubert Baty and L T Baty. "Solving differential equations using physics informed deep learning: a hand-on tutorial with benchmark tests". In: *ArXiv* abs/2302.12260 (2023).

[182] Ebenezer O. Oluwasakin and Abdul Q. M. Khaliq. "Optimizing Physics-Informed Neural Network in Dynamic System Simulation and Learning of Parameters". In: *Algorithms* 16.12 (2023). ISSN: 1999-4893.

[183] Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. *An Expert's Guide to Training Physics-informed Neural Networks*. 2023.

[184] John H. Lagergren, John T. Nardini, Ruth E. Baker, Matthew J. Simpson, and Kevin B. Flores. "Biologically-informed neural networks guide mechanistic modeling from sparse experimental data". In: *PLOS Computational Biology* 16.12 (Dec. 2020), pp. 1–29.

[185] Hubert Baty. *Solving stiff ordinary differential equations using physics informed neural networks (PINNs): simple recipes to improve training of vanilla-PINNs*. 2023.

[186] Patrick Stiller, Friedrich Bethke, Maximilian Böhme, Richard Pausch, Sunna Torge, Alexander Debus, Jan Vorberger, Michael Bussmann, and Nico Hoffmann. "Large-Scale Neural Solvers for Partial Differential Equations". In: *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*. Ed. by Jeffrey Nichols, Becky Verastegui, Arthur 'Barney' Maccabe, Oscar Hernandez, Suzanne Parete-Koon, and Theresa Ahearn. Cham: Springer International Publishing, 2020, pp. 20–34. ISBN: 978-3-030-63393-6.

[187] Kamaljyoti Nath, Xuhui Meng, Daniel J. Smith, and George Em Karniadakis. "Physics-informed neural networks for predicting gas flow dynamics and unknown parameters in diesel engines". In: *Scientific Reports* 13.1 (Aug. 2023). ISSN: 2045-2322.

[188] Sigurdur Mar Valsson Ivan Depina Saket Jain and Hrvoje Gotovac. "Application of physics-informed neural networks to inverse problems in unsaturated groundwater flow". In: *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards* 16.1 (2022), pp. 21–36.

[189] Zdravko I. Botev, Dirk P. Kroese, Reuven Y. Rubinstein, and Pierre L'Ecuyer. "Chapter 3 - The Cross-Entropy Method for Optimization". In: *Handbook of Statistics*. Ed. by C.R. Rao and Venu Govindaraju. Vol. 31. Handbook of Statistics. Elsevier, 2013, pp. 35–59.

[190] Jeremías Garay, Jocelyn Dunstan, Sergio Uribe, and Francisco Sahli Costabal. "Physics-informed neural networks for parameter estimation in blood flow models". In: *Computers in Biology and Medicine* 178 (2024), p. 108706. ISSN: 0010-4825.

[191] Marco Berardi, Fabio Vito Difonzo, and Matteo Icardi. "Inverse Physics-Informed Neural Networks for transport models in porous materials". In: *ArXiv* abs/2407.10654 (2024).

[192] Ameya D. Jagtap, Zhiping Mao, Nikolaus Adams, and George Em Karniadakis. "Physics-informed neural networks for inverse problems in supersonic flows". In: *Journal of Computational Physics* 466 (2022), p. 111402. ISSN: 0021-9991.

[193] Fabio Difonzo, Luciano Lopez, and Sabrina Pellegrino. "Physics informed neural networks for an inverse problem in peridynamic models". In: *Engineering with Computers* (Mar. 2024), pp. 1–10.

[194] Enrico Schiassi, Mario De Florio, Barry D. Ganapol, Paolo Picca, and Roberto Furfaro. "Physics-informed neural networks for the point kinetics equations for nuclear reactor dynamics". In: *Annals of Nuclear Energy* 167 (2022), p. 108833. ISSN: 0306-4549.

[195] Mario De Florio, Ioannis G. Kevrekidis, and George Em Karniadakis. "AI-Lorenz: A physics-data-driven framework for Black-Box and Gray-Box identification of chaotic systems with symbolic regression". In: *Chaos, Solitons & Fractals* 188 (2024), p. 115538. ISSN: 0960-0779.

[196] Elham Kiyani, Khemraj Shukla, George Em Karniadakis, and Mikko Karttunen. "A framework based on symbolic regression coupled with eXtended Physics-Informed Neural Networks for gray-box learning of equations of mo-

tion from data". In: *Computer Methods in Applied Mechanics and Engineering* 415 (2023), p. 116258. ISSN: 0045-7825.

[197] Zhen Zhang, Zongren Zou, Ellen Kuhl, and George Em Karniadakis. "Discovering a reaction–diffusion model for Alzheimer's disease by combining PINNs with symbolic regression". In: *Computer Methods in Applied Mechanics and Engineering* 419 (2024), p. 116647. ISSN: 0045-7825.

[198] Pu Chen, Aiguo Shen, Xiaodong Zhou, and Jiming Hu. "Bio-Raman spectroscopy: a potential clinical analytical method assisting in disease diagnosis". en. In: *Anal. Methods* 3.6 (2011), p. 1257.

[199] Michael A Gimbrone Jr and Guillermo García-Cardeña. "Endothelial cell dysfunction and the pathobiology of atherosclerosis". en. In: *Circ. Res.* 118.4 (Feb. 2016), pp. 620–636.

[200] Chad A Lieber and Anita Mahadevan-Jansen. "Automated method for subtraction of fluorescence from biological Raman spectra". en. In: *Appl. Spectrosc.* 57.11 (Nov. 2003), pp. 1363–1367.

[201] Eva Ostertagová. "Modelling using polynomial regression". en. In: *Procedia Eng.* 48 (2012), pp. 500–506.

[202] Fionn Murtagh. "Multilayer perceptrons for classification and regression". en. In: *Neurocomputing* 2.5-6 (July 1991), pp. 183–197.

[203] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2017.

[204] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques.* 3rd ed. The Morgan Kaufmann Series in Data Management Systems. Oxford, England: Morgan Kaufmann, June 2011.

[205] Franco M Impellizzeri, Ian Shrier, Shaun J McLaren, Aaron J Coutts, Alan McCall, Katie Slattery, Annie C Jeffries, and Judd T Kalkhoven. "Understanding training load as exposure and dose". en. In: *Sports Med.* 53.9 (Sept. 2023), pp. 1667–1679.

[206] Steven H Doeven, Michel S Brink, Wouter G P Frencken, and Koen A P M Lemmink. "Impaired player–coach perceptions of exertion and recovery during match congestion". In: *Int. J. Sports Physiol. Perform.* 12.9 (Oct. 2017), pp. 1151–1156.

[207] Mathieu Lacome, Ben Simpson, Nick Broad, and Martin Buchheit. "Monitoring players' readiness using predicted heart-rate responses to soccer drills". en. In: *Int. J. Sports Physiol. Perform.* 13.10 (Nov. 2018), pp. 1273–1280.

[208] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[209] Evangelos Rozos, Panayiotis Dimitriadis, Katerina Mazi, and Antonis D. Koussis. "A Multilayer Perceptron Model for Stochastic Synthesis". en. In: *Hydrology* 8.2 (June 2021). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 67. ISSN: 2306-5338.

[210] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687.

[211] Rafael Schild Reusch, Leonardo Rezende Juracy, and Fernando Gehm Moraes. "Assessment and Optimization of 1D CNN Model for Human Activity Recognition". In: *2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC)*. Nov. 2022, pp. 1–7.

[212] Dinh Viet Cuong, Vuong M. Ngo, Paolo Cappellari, and Mark Roantree. "Analyzing Shared Bike Usage Through Graph-Based Spatio-Temporal Modeling". In: *IEEE Open Journal of Intelligent Transportation Systems* 5 (2024), pp. 115–131.

[213] Mark Roantree, Niamh Murphy, Dinh Viet Cuong, and Vuong M. Ngo. "Graph-Based Optimisation of Network Expansion in a Dockless Bike Sharing System". In: *2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW)*. 2024, pp. 48–55.

[214] Mark Roantree and Jun Liu. "A heuristic approach to selecting views for materialization". In: *Software: Practice and Experience* 44.10 (2014), pp. 1157–1179.

[215] James Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[216] Fouad Bahrpeyma, Mark Roantree, Paolo Cappellari, Michael Scriney, and Andrew McCarren. "A Methodology for Validating Diversity in Synthetic Time Series Generation". In: *MethodsX* 8 (2021), p. 101459. ISSN: 2215-0161.

[217] Dong Xin, Jiawei Han, Xiaolei Li, Zheng Shao, and Benjamin W. Wah. "Computing Iceberg Cubes by Top-Down and Bottom-Up Integration: The StarCubing Approach". In: *IEEE Transactions on Knowledge and Data Engineering* 19.1 (2007), pp. 111–126.

[218] Kurt Mehlhorn, Stefan Näher, and Peter Sanders. "Engineering DFS-Based Graph Algorithms". In: *ArXiv* abs/1703.10023 (2017).

[219] Hélio Almeida, Dorgival Guedes, Wagner Meira, and Mohammed J. Zaki. "Is There a Best Quality Metric for Graph Clusters?" In: *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2011, pp. 44–59. ISBN: 978-3-642-23780-5.

[220] J. M. Hernández and P. V. Mieghem. *Classification of graph metrics*. Tech. rep. Technical report. 2628 CD Delft: Faculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology, Nov. 2011.

[221] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. "The architecture of complex weighted networks". In: *Proceedings of the National Academy of Sciences* 101.11 (2004), pp. 3747–3752.

[222] Linton C. Freeman. "Centrality in social networks conceptual clarification". In: *Social Networks* 1.3 (1978), pp. 215–239. ISSN: 0378-8733.

[223] Gert Sabidussi. "The centrality index of a graph". In: *Psychometrika* 31.4 (Dec. 1966), pp. 581–603. ISSN: 1860-0980.

[224] Ulrik Brandes and Christian Pich. “CENTRALITY ESTIMATION IN LARGE NETWORKS”. In: *International Journal of Bifurcation and Chaos* 17.07 (2007), pp. 2303–2318.

[225] Linton C. Freeman. “A Set of Measures of Centrality Based on Betweenness”. In: *Sociometry* 40.1 (1977), pp. 35–41. ISSN: 00380431.

[226] Duncan J. Watts and Steven H. Strogatz. “Collective dynamics of ‘small-world’ networks”. In: *Nature* 393.6684 (June 1998), pp. 440–442. ISSN: 1476-4687.

[227] Santo Fortunato. “Community detection in graphs”. In: *Physics Reports* 486.3 (2010), pp. 75–174. ISSN: 0370-1573.

[228] Mikail Rubinov and Olaf Sporns. “Complex network measures of brain connectivity: uses and interpretations”. en. In: *Neuroimage* 52.3 (Oct. 2009), pp. 1059–1069.

[229] Karsten Steinhaeuser, Nitesh V. Chawla, and Auroop R. Ganguly. “Complex networks as a unified framework for descriptive analysis and predictive modeling in climate science”. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 4.5 (2011), pp. 497–511.

[230] J. F. Donges, Y. Zou, N. Marwan, and J. Kurths. “Complex networks in climate dynamics”. In: *The European Physical Journal Special Topics* 174.1 (July 2009), pp. 157–179. ISSN: 1951-6401.

[231] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272.

[232] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[233] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring network structure, dynamics, and function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference (SciPy2008)*. Ed. by Gäel Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, Aug. 2008, pp. 11–15.

[234] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020.

[235] Google. *Google Maps API*. `https://developers.google.com/maps`. Accessed: 2023-12-01.

[236] Neo4j, Inc. *Neo4j Graph Database*. `https://neo4j.com/`. Accessed: 2023-12-01.

[237] Neo4j, Inc. *Neo4j's Graph Data Science Library*. `https://neo4j.com/docs/graph-data-science/current/`. Accessed: 2023-12-01.

[238] P. Baby K. Sasirekha. "Agglomerative Hierarchical Clustering Algorithm- A Review". In: *International Journal of Scientific and Research Publications* 3 (Mar. 2013).

[239] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley, 2011.

[240] Bing Yu, Haoteng Yin, and Zhanxing Zhu. "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, July 2018.

[241] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2.

[242] Shuai Han, Lukas Stelz, Horst Stoecker, Lingxiao Wang, and Kai Zhou. "Approaching epidemiological dynamics of COVID-19 with physics-informed neural networks". In: *Journal of the Franklin Institute* 361.6 (2024), p. 106671. ISSN: 0016-0032.

[243] Dinh Viet Cuong, Branislava Lalić, Mina Petrić, Nguyen Thanh Binh, and Mark Roantree. "Adapting physics-informed neural networks to improve ODE optimization in mosquito population dynamics". In: *PLOS ONE* 19.12 (Dec. 2024), pp. 1–30.

[244] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[245] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2023.

[246] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256.

[247] Edward Norton Lorenz. "Deterministic nonperiodic flow". In: *Journal of the Atmospheric Sciences* 20 (1963), pp. 130–141.

[248] Linda Petzold. "Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations". In: *SIAM Journal on Scientific and Statistical Computing* 4.1 (1983), pp. 136–148.

[249] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 7537–7547.

[250] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[251] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. “Dying relu and initialization: Theory and numerical examples”. In: *arXiv preprint arXiv:1903.06733* (2019).