UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Informáticos

Beyond the Error Metric: Enhancing Spatio-Temporal Sensor Data Forecasting with Non-Functional Features

D O C T O R A L T H E S I S

Submitted for the degree of Doctor by:

Ignacio Iker Prado Rujas
MSc in Data Science

Madrid, 2025


UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Informáticos

Doctoral Degree in Artificial Intelligence

Beyond the Error Metric: Enhancing Spatio-Temporal Sensor Data Forecasting with Non-Functional Features

D O C T O R A L T H E S I S

Submitted for the degree of Doctor by:

Ignacio Iker Prado Rujas
MSc in Data Science

Under the supervision of:
Dr. Emilio Serrano Fernández
Dr. Antonio García Dopico

Madrid, 2025

Title: Beyond the Error Metric: Enhancing Spatio-Temporal Sensor Data Forecasting with Non-Functional Features

Author: Ignacio Iker Prado Rujas

Doctoral Programme: Artificial Intelligence

Thesis Supervision:

Dr. Emilio Serrano Fernández, Associate Professor, Universidad Politécnica de Madrid

Dr. Antonio García Dopico, Associate Professor, Universidad Politécnica de Madrid

External Reviewers:

Thesis Defense Committee:

Thesis Defense Date:

This thesis has been partially supported by the Autonomous Region of Madrid, Spain, through the program CABAHLA-CM (GA No. P2018/TCS-4423).

To all the teachers and mentors who led me here.

Acknowledgement

Doing research, especially for a PhD thesis, is at times like waking up disoriented in a dark room. In the following, I do my best to thank everyone who, in one way or another, helped me find the light switch.

To begin with, I would like to thank my supervisors Antonio and Emilio for their support in this process and their patience. Although I was not the fastest runner, I hope the voyage was worthwhile. Additionally and even if not listed as an official supervisor, María S. Pérez was truly the cornerstone of this journey. Thank you for trusting and supporting me tirelessly since the beginning. I would also like to thank Marisa Córdoba for her support and manifold fruitful discussions. Thanks also to Javier Campoy for sharing part of this journey with me. It was great having someone else who was deep into the hands-on work.

Talking about hands-on work, all the models and experiments were developed in machines of the Departamento de Arquitectura y Tecnología de Sistemas Informáticos (DATSI) of UPM, managed by my supervisor Antonio. Without those servers and GPUs I would have probably melted my laptop, repeatedly, so thank you for letting me use these resources. I would also like to thank UPM's Ontology Engineering Group (OEG) for hosting a big share of my PhD. I will always keep good memories of those DiversiTeam football afternoons. Thank you, José Ángel Ramos (aka JARG), Ana Ibarrola, and Reyes Riera for the administrative support, and to Raúl Alcázar for the IT support. Additionally, I would like to thank the professors who allowed me to assist them with teaching: Emilio, Javier Bajo, and Martín Molina.

About halfway through my PhD, I was lucky enough to undertake a research internship at Galway. For that experience, I am deeply grateful to Edward Curry, who selflessly hosted me at Insight. I have great memories of that period, many of them thanks to and shared with Mirza and Rishab. You really made me feel welcome from day one, and I learned a lot during all our kitchen talks. Also, thanks to the external evaluators for the time spent on this thesis, providing meaningful observations and comments that greatly helped me improve the final result.

This journey entwines with my work at JRC, Ispra, which was a vital lifeline for me. Even if the endeavors were unrelated to the PhD, they provided me a way to switch hats and clear my mind. Foremost, thank you Giorgos for your neverending guidance and help, I truly cherish both. For their support and encouragement, I would like to thank Kostis, Nikiforos, Stijn, Vangelis, and all my LEGENT(dary) colleagues and friends.

On a personal level, I would like to thank my friends who have been an integral part of my workaday life. From Ispra, thank you Alessandro, Bernat, Dimitris, and Kostas. Your encouragement, support, and the time we spend together are always bliss. Por supuesto, gracias también a mis compañeros y amigos del doble grado, Kike, Jaime, Jesús, Mayra, y Peñas. Pero ya está bien de preguntar por la tesis, hombre ya, ¡por favor! Gracias a Alex (aka Lexas) por esas charlas en el banco y por instarme a que acabara. Aunque nos veamos poco, siempre es como si no hubiera pasado tiempo. Sin duda además, gracias al Singer Basket, vía de escape física y distracción mental necesarias para no perder la razón.

Por último, y por ello más importante, gracias a mi familia por estar ahí a lo largo y ancho de todo este camino. A mis padres, Laura y Juancar, por vuestra paciencia y por dejarme

espacio para trabajar en lo que sea que haya estado haciendo durante estos años. Y por supuesto, gracias Eli, por haber estado y estar en mi día a día, en los buenos y malos momentos, pero sobre todo en los peores. Gracias por ayudarme y apoyarme siempre de manera incondicional.

Abstract

The presence of sensors that collect data is rapidly growing across various domains. They are used for a plethora of applications, particularly in the context of smart cities. This thesis focuses on fixed, geolocated time series sensor data. Such data can be leveraged to develop forecasting models that estimate future values observed by these sensors at a given time.

This thesis studies the problem of spatio-temporal sensor data forecasting. The objective is to take advantage of multiple time series from neighboring sensors that measure the same variables. By doing so, forecasting models can benefit from a set of measurements across a region at each timestamp, rather than relying solely on the observations of a single sensor over time. Once the model is operational, its performance is typically evaluated using error metrics, such as the Root-Mean-Squared Error (RMSE). While such metrics are important, they neglect vital aspects of the model's usability and adaptability in real-world conditions. This thesis focuses on enhancing these characteristics, referred to as Non-Functional Features (NFFs).

To tackle this problem, we propose a generic three-step methodology. First, we analyze and transform the geolocated time series data through seven predefined phases. The outcome of this step is a set of mesh-grids and relevant metadata. Next, we introduce the generic Spatio-Temporal Forecaster (ST-F) model, whose core component is a spatio-temporal module designed to extract features from the generated mesh-grids. Additionally, the ST-F model can incorporate exogenous modules to enhance its predictive capabilities, such as flat-extraction or recurrent ones. Finally, we devise how to evaluate the model using both conventional error metrics and the proposed NFFs.

To validate our methodology, we apply it to three diverse spatio-temporal forecasting use cases.

1. Solar irradiance: We evaluate how the model adapts to a variable set of sensors and missing sensor observations. We denote these NFFs as flexibility and robustness.

2. Mobility demand: The second use case considers taxi and shared bicycle trips forecasting. Here, we introduce the extensibility NFF, which evaluates how the model benefits from combining different but related forecasted variables. Additionally, we study integrability, i.e., how the model improves the forecasts by incorporating exogenous data thanks to its modular structure.

3. Air quality: In the third use case, the ST-F model combines up to eleven pollutants, besides accomplishing additional NFFs.

Overall, this thesis contributes to developing a methodology for understanding and forecasting spatio-temporal sensor data. Beyond achieving low error metrics, it seeks additional beneficial properties of forecasting models, enhancing their real-world applicability.

Resumen

La presencia de sensores que recogen datos está creciendo rápidamente en distintos dominios. Se utilizan para una gran variedad de aplicaciones, en particular en el contexto de las ciudades inteligentes. Esta tesis se centra en series temporales de datos de sensores fijos y geolocalizados. Estos datos pueden ser aprovechados para desarrollar modelos de predicción que estimen valores futuros observados por esos sensores en un momento determinado.

Esta tesis estudia el problema de la predicción espacio-temporal de datos de sensores. El objetivo es aprovechar múltiples series temporales de sensores cercanos que midan las mismas variables. De este modo, los modelos de predicción se pueden beneficiar de un conjunto de medidas en toda una región en cada instante, en lugar de basarse únicamente en observaciones de un solo sensor a lo largo del tiempo. Una vez que el modelo está operativo, su rendimiento se evalúa típicamente mediante métricas de error, como la raíz del error cuadrático medio. Si bien dichas métricas son importantes, dejan de lado aspectos vitales como la usabilidad y adaptabilidad del modelo en condiciones reales. Esta tesis se centra en mejorar dichas características, referidas como atributos no funcionales.

Para abordar este problema, proponemos una metodología genérica de tres pasos. Primero, analizamos y transformamos los datos geolocalizados de series temporales mediante siete fases predefinidas. El resultado de este paso es un conjunto de rejillas o matrices y metadatos relevantes. A continuación, presentamos el modelo genérico Spatio-Temporal Forecaster (ST-F), cuyo componente central es un módulo espacio-temporal que extrae características de las rejillas generadas. Adicionalmente, el modelo ST-F puede incorporar módulos exógenos para mejorar sus capacidades de predicción, como los de extracción plana o los recurrentes. Finalmente, diseñamos un método para evaluar el modelo utilizando tanto métricas de error convencionales como los atributos no funcionales propuestos.

Para validar nuestra metodología, la aplicamos a tres casos de uso diversos sobre predicción espacio-temporal.

1. Irradiancia solar: Evaluamos cómo el modelo se adapta a un conjunto variable de sensores y a observaciones faltantes de los sensores. Denotamos esos atributos como flexibilidad y robustez.

2. Demanda de movilidad: El segundo caso de uso considera la predicción de viajes de taxi y bicicletas compartidas. Aquí, introducimos el atributo denominado extensibilidad, que evalúa cómo el modelo se beneficia de combinar distintas variables predichas relacionadas entre sí. Adicionalmente, estudiamos la integrabilidad, es decir, cómo el modelo mejora sus predicciones al incorporar datos exógenos gracias a su estructura modular.

3. Calidad del aire: En el tercer caso de uso, el modelo ST-F combina hasta once gases contaminantes, además de conseguir los otros atributos no funcionales.

En conjunto, esta tesis contribuye a desarrollar una metodología para comprender y predecir datos espacio-temporales de sensores. Más allá de conseguir métricas de error bajas, busca lograr propiedades beneficiosas para los modelos de predicción, mejorando su aplicabilidad práctica.

Contents

List of Figures

List of Tables

Acronyms

AEMET	Agencia Estatal de Meteorología
AI	Artificial Intelligence
API	Application Programming Interface
AQI	Air Quality Index
ARIMA	AutoRegressive Integrated Moving Average
ARX	Auto Regressive models with eXogenous inputs
ASO	Almost Stochastic Order
BEN	benzene
BiLSTM	Bidirectional LSTM
CABAHLA	ConvergenciA Big dAta Hpc: de Los sensores a las Aplicaciones
CAIDE	Cloud-based Analysis and Integration for Data Efficiency
CET	Central European Time
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
CO	carbon monoxide
CO_2	carbon dioxide
ConvLSTM	Convolutional LSTM
CSI	Clear-Sky Index
CST	Central Standard Time
CSV	Comma-Separated Values
DEVS	Discrete Event System Specification
DHI	Diffuse Horizontal Irradiance
DL	Deep Learning
DNI	Direct Normal Irradiance
DNN	Deep Neural Network
EBE	ethylbenzene
EC	European Commission
EDA	Exploratory Data Analysis
EEA	European Environment Agency
EU	European Union
FC	Fully Connected
FHV	For-Hire Vehicle

GB	Great Britain
GHI	Global Horizontal Irradiance
GNN	Graph Neural Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HPC	High-Performance Computing
ICT	Information and Communications Technologies
IoT	Internet of Things
k-NN	k-nearest neighbors algorithm
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MBSE	Model-Based Systems Engineering
MBTA	Massachusetts Bay Transportation Authority
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MSLE	Mean Squared Logarithmic Error
n/a	not available
NaN	Not a Number
NASA	National Aeronautics and Space Administration
NFF	Non-Functional Feature
NN	Neural Network
NO	nitrogen monoxide
NO_2	nitrogen dioxide
NO_x	nitrogen oxides
NREL	National Renewable Energy Laboratory
nRMSE	normalized RMSE
NWP	Numerical Weather Prediction
NYC	New York City
O_3	ozone
ODbL	Open Database License
PM	particulate matter
PM_{10}	coarse particulate matter
$PM_{2.5}$	fine particulate matter
PSA	Plataforma Solar de Almería
PV	Photovoltaic
PVGIS	Photovoltaic Geographical Information System

ReLu	REctified Linear Unit
RGB	red, green, and blue
RL	Reinforcement Learning
RMSE	Root-Mean-Squared Error
RNN	Recurrent Neural Network
ROI	Region Of Interest
SARIMA	Seasonal ARIMA
SDG	Sustainable Development Goal
SO_2	sulfur dioxide
ST-AQF	Spatio-Temporal Air Quality Forecaster
ST-F	Spatio-Temporal Forecaster
ST-MDF	Spatio-Temporal Mobility Demand Forecaster
ST-SIF	Spatio-Temporal Solar Irradiance Forecaster
TOL	toluene
UK	United Kingdom
USA	United States of America
WHO	World Health Organization
wRMSE	weighted RMSE

CHAPTER 1

Introduction

The volume and diversity of sensor data continue to increase rapidly, driven by the expanding adoption of Internet of Things (IoT) technologies. Wasicek (2020) expected that over 40 billion IoT devices were connected globally by 2025. However, current projections are closer to 20 billion, with an estimated annual growth rate of nearly 15 % for the next few years[1]. These devices are embedded in various smart city applications, such as:

- climate monitoring (Chen et al., 2019),
- solar energy forecasting (Amaro e Silva & Brito, 2018),
- air quality assessment (Fan et al., 2017), and
- traffic and mobility optimization (Zahid et al., 2020).

Reliable short and mid-term forecasting of sensor data is essential for optimizing energy production, implementing pollution protocols, and managing urban mobility services.

Typically, sensor readings are timestamped, allowing them to be structured as time series data. Additionally, many applications involve networks of geolocated sensors, giving rise to data observations with spatial and temporal dimensions. Consequently, understanding spatial correlations in addition to temporal trends is crucial for effective forecasting. Such spatio-temporal relationships can reveal how sensor observations in a given period or region may influence observations in another based on temporal or geographical proximity.

Once the core characteristics of spatio-temporal sensor data are established, we can explore various predictive modeling approaches. Traditional time series models, such as AutoRegressive Integrated Moving Average (ARIMA) (Aasim et al., 2019),

[1] https://iot-analytics.com/number-connected-iot-devices/ (last accessed: 2025/03/15)

and more recent approaches, including random forest (Huertas Tato & Centeno Brito, 2019), Neural Networks (NNs) (Amato et al., 2020), and Reinforcement Learning (RL) (Walraven et al., 2016), are commonly employed. Nevertheless, all the proposed case studies share a dynamic nature: sensor networks are frequently updated with new devices, geolocations, or additional data variables, and urban expansion can redefine areas of interest. Thus, model adaptability and generalization capabilities are essential for successful deployment in real-world contexts. This thesis aims to identify and provide these Non-Functional Features (NFFs), such as flexibility, robustness, and extensibility, rather than focusing solely on error minimization.

As mentioned above, current works employ a variety of forecasting approaches. Deep Learning (DL), a subset of Machine Learning (ML) that utilizes multi-layered NNs, offers advanced capabilities for learning complex, non-linear functions. These models, referred to as Deep Neural Networks (DNNs), can capture convoluted relationships across spatio-temporal dimensions. Recent DL advances have demonstrated notable success in tasks such as image classification, object segmentation, sequence prediction, and even generative applications. DNNs ability to learn high-level features from vast datasets makes them promising candidates for spatio-temporal forecasting tasks, where both time dependencies and spatial correlations must be considered.

1.1 Objectives

The main objective of this thesis is as follows:

> To design a generic DL-based model for spatio-temporal sensor data forecasting that demonstrates valuable characteristics for real-world deployment and reusability.

This objective is further broken down into the following subgoals:

1. **Beneficial modeling features for spatio-temporal sensor data forecasting**:
   (a) To identify and define key features that enhance forecasting performance and applicability.
   (b) To define evaluation techniques for assessing these features.
2. **Spatio-temporal sensor data analysis and processing**:
   (a) To conceive strategies for analyzing and interpreting spatio-temporal sensor data.
   (b) To harmonize procedures for processing these data.

(c) To devise robust techniques for handling missing data.

3. **DL-based model for spatio-temporal sensor data forecasting**:

   (a) To design a generic model architecture for forecasting spatio-temporal data.

   (b) To equip the model with the beneficial properties previously identified.

4. **Real-world applicability and reusability**:

   (a) To ensure the designed methodology suits a wide range of real-world scenarios.

1.2 Contributions

In this work, we contribute to the field of spatio-temporal sensor data forecasting in several ways. We aim not only to establish a methodology for working with this type of data but also to define a set of desired characteristics that facilitate the real-world deployment of these models. The specific contributions of this thesis are as follows:

⋆ **C1**: Definition of NFFs for modeling spatio-temporal sensor data, including methodologies to evaluate these features.

⋆ **C2**: Development of a methodology for analyzing and transforming spatio-temporal sensor data to enable effective modeling.

⋆ **C3**: Design of a generic, modular model capable of processing and forecasting spatio-temporal sensor data.

⋆ **C4**: Application and evaluation of the methodology, yielding:

- ⋆ **C4.1**: Simultaneous handling of observations from multiple sensors and timestamps, both as inputs and outputs to the model.
- ⋆ **C4.2**: Joint modeling of multiple related variables.
- ⋆ **C4.3**: Lower error levels than the defined baselines.
- ⋆ **C4.4**: Forecast error that outperforms comparable models from the literature.
- ⋆ **C4.5**: Adaptability to different numbers of sensors without modifying the model structure.
- ⋆ **C4.6**: Dynamic estimation of missing sensor observations.
- ⋆ **C4.7**: Integration of exogenous datasets to improve model performance.

Contribution **C4** spans across three forecasting use cases, achieving the following specific contributions in each case:

- Solar irradiance: **C4.1**, **C4.3**, **C4.5**, and **C4.6**.
- Mobility demand: **C4.1**, **C4.2**, **C4.3**, **C4.4**, **C4.5**, **C4.6**, and **C4.7**.
- Air quality: **C4.1**, **C4.2**, **C4.3**, **C4.4**, **C4.5**, **C4.6**, and **C4.7**

1.3 Thesis structure

The remainder of this thesis is organized as follows:

- Chapter 2 presents the current state of the art relevant to this thesis. We identify the limitations of existing work on NFFs and highlight gaps that this research aims to address. In addition to exploring spatio-temporal forecasting models, we outline various data and visualization resources.
- Chapter 3 purpose is three-fold: defining terminology, formalizing the problem, and identifying NFFs. These elements lay the foundation for the methodology in the subsequent chapter.
- Chapter 4 describes our methodology. We begin with spatio-temporal sensor data analysis and transformations for modeling. Next, we introduce our generic modeling framework, the Spatio-Temporal Forecaster (ST-F). Finally, we devise the evaluation methodology for the error metrics and the NFFs that we study.
- Chapter 5 presents the first of three case studies applying our methodology to real-world sensor data. This initial validation focuses on forecasting a single variable using a small number of sensors, scattered across a region. The studied variable is solar irradiance, which provides insights into solar potential and energy production.
- Chapter 6 expands the scope to a larger area and increased number of sensors for mobility demand forecasting. Specifically, this study models both taxi trips and bikeshare rides simultaneously. Additionally, the model incorporates exogenous data to enhance predictions, which is studied as part of the NFFs.
- Chapter 7 presents the third and final case study, which forecasts up to eleven air pollution variables. Following our established methodology, we begin with data analysis and transformation. Second, we proceed with the forecasting approach and baseline models, and conclude with experiments that assess error metrics and NFFs. All three case studies are structured similarly to facilitate readability and consistency.

- Finally, Chapter 8 presents the main conclusions of this thesis and outlines future research directions for further refining our methodology and the studied use cases.

1.4 Dissemination results

This thesis contributes to a larger research project, encompassing four journal articles, three conference papers, three code repositories, two data repositories, and a poster presentation. The specific contributions are detailed below.

Firstly, this thesis is part of the following **research project**:

1. P2018/TCS-4423: ConvergenciA Big dAta Hpc: de Los sensores a las Aplicaciones (CABAHLA) (1st January, 2019 to 30th April, 2023). Funded by the Autonomous Region of Madrid, Spain, the project's primary aim is the integration of High-Performance Computing (HPC) and Big Data. It covers the scope from sensor data acquisition to supercomputer-based processing, with an emphasis on DL modeling.

Our contribution to describing NFF for spatio-temporal sensor data forecasting was published in the following **conference**:

2. **Prado-Rujas**, **I.-I.**, Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2021b). Predicción espacio-temporal: Más allá del error/precisión, In *Congreso Español De Informática 2021 (CEDI'21)*, Málaga, Spain. DOI: 10.5281/zenodo.5530048.

The first use case from this research is published in the following **journal**:

3. **Prado-Rujas**, **I.-I.**, García-Dopico, A., Serrano, E., & Pérez, M. S. (2021). A flexible and robust deep learning-based system for solar irradiance forecasting. *IEEE Access*, *9*, 12348–12361. DOI: 10.1109/ACCESS.2021.3051839.

Due to data distribution restrictions (see Sengupta & Andreas, 2010), we were unable to release the produced data. However, the datasets can be reproduced using the resources provided in the following **repository** (further detailed in Annex A):

4. **Prado-Rujas**, **I.-I.**, García-Dopico, A., Serrano, E., & Pérez, M. S. (2020). *Results, graphs and code for a Deep Learning system for spatio-temporal solar irradiance forecasting* (repository). https://github.com/iipr/solar-irradiance (last accessed: 2025/03/15).

Additionally, we integrated our model into a simulation framework as part of this case study, which is published in the following **journal**:

5. Risco-Martín, J. L., **Prado-Rujas**, **I.-I.**, Campoy, J., Pérez, M. S., & Ol-

coz, K. (2024). Advanced simulation-based predictive modelling for solar irradiance sensor farms. *Journal of Simulation*, 1–18. DOI: 10.1080/17477778.2024.2333775.

Besides that, our research on various retraining techniques using the same framework is published in these **conferences**:

6. Almendras, L., Campoy, J., **Prado-Rujas**, **I.-I.**, Risco-Martin, J. L., Perez, M. S., & Olcoz, K. (2022). A distributed IoT-based simulation framework for solar energy management and forecasting, In *Jornadas SARTECO 2022*, Alicante, Spain. DOI: 10.5281/zenodo.7075818.

7. Campoy, J., **Prado-Rujas**, **I.-I.**, Risco-Martin, J. L., Olcoz, K., & Perez, M. S. (2023). Distributed training and inference of deep learning solar energy forecasting models, In *2023 31st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Naples, Italy. DOI: 10.1109/PDP59025.2023.00035.

The second use case is published in the following **journal**:

8. **Prado-Rujas**, **I.-I.**, Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2023a). Combining heterogeneous data sources for spatio-temporal mobility demand forecasting. *Information Fusion*, *91*, 1–12. DOI: 10.1016/j.inffus.2022.09.028.

Details on the corresponding software are found in Annex A, with its publication in the following **repository**:

9. **Prado-Rujas**, **I.-I.**, Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2021c). *Results, graphs and code for a Deep Learning system for spatio-temporal mobility demand forecasting* (repository). https://github.com/iipr/mobility-demand (last accessed: 2025/03/15)

Additionally, the datasets are available in the following **open-access data repository**:

10. **Prado-Rujas**, **I.-I.**, Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2021a). *Datasets for a Deep Learning system for mobility demand forecasting* (dataset). DOI: 10.5281/zenodo.5166839.

The third use case is published in the following **journal**:

11. **Prado-Rujas**, **I.-I.**, García-Dopico, A., Serrano, E., Córdoba, M. L., & Pérez, M. S. (2024). A multivariable sensor-agnostic framework for spatio-temporal air quality forecasting based on Deep Learning. *Engineering Applications of Artificial Intelligence*, *127*, 107271. DOI: 10.1016/j.engappai.2023.107271.

The associated software is discussed in Annex A and is accessible in the following **repository**:

12. **Prado-Rujas**, **I.-I.**, Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2023c). *Results, graphs and code for a Deep Learning system for spatio-temporal air quality forecasting* (repository). https://github.com/iipr/air-quality (last accessed: 2025/03/15).

Additionally, the datasets for this use case are also made available in the following **open-access data repository**:

13. **Prado-Rujas**, **I.-I.**, Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2023b). *Datasets for a Deep Learning system for spatio-temporal air quality forecasting* (dataset). DOI: 10.5281/zenodo.7764528.

While working on the third use case, we undertook a research internship at the Insight SFI Research Centre for Data Analytics at the University of Galway, Ireland. During this internship, we worked with low-power physical air quality sensors and analyzed a dataset of air quality measurements collected from delivery vans operating in Dublin. We produced the following **poster** to disseminate the results of the internship:

14. Acquier, A., Jordan, E., **Prado-Rujas**, **I.-I.**, Serrano, M., & Smith, B. (2022). *Sensing Air - A citywide citizen science air quality monitoring system* (poster presentation). Youth Climate Assembly 2022, University of Galway, Ireland. (not available online).

Finally, we provided guidance for the following theses:

1. González García, D. (2020). *Predicción de la radiación solar mediante el uso de una red neuronal convolucional* (Bachelor's Thesis). ETSI Informática (Universidad Politécnica de Madrid). (not available online).

2. Heras Aranzana, P. (2020). *Predicción de la radiación mediante el uso de una red neuronal recurrente* (Bachelor's Thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/63345/.

3. García Martínez, M. (2021). *Bike sharing demand forecasting using recurrent neural networks* (Bachelor's Thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/66259/.

4. Campoy Heredero, J. (2023). *Exploring a deep learning based spatio-temporal forecasting model: Testing for scalability and other features* (Master's thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/75899/.

5. Markmann, S. (2023). *Application of graph neural networks for urban mo-*

bility demand forecasting (Master's thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/75995/.

6. Moya Iratxeta, K. (2024). *Predicción de la demanda de movilidad espacio-temporal del transporte público mediante transformers* (Master's thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/82617/.

CHAPTER 2

State of the art

This chapter reviews the related work and current state of the art. Section 2.1 begins by discussing the growing use of sensors for developing smart city applications. Next, we explore relevant studies for the three key use cases: solar irradiance (Section 2.2), mobility demand (Section 2.3), and air quality (Section 2.4). For each use case, we first examine data repositories and visualization tools, followed by forecasting methods. Section 2.5 highlights the importance of identifying generalization attributes in forecasting models, drawing on examples from the literature. Lastly, we summarize the key points and related work in Section 2.6.

2.1 Sensors and smart city applications

As detailed in Section 3.1.1, we define *sensor* as a device that acquires measurements of variables. Sensors are instrumental for experimental science. Research and development have enabled them to become widely used for many applications, such as smart cities. This term, *smart city*, has increasingly become a buzzword during the past decade. Nevertheless, the ideas and methods behind it emerged a while ago (the term appears as early as in 1992, see Gibson et al., 1992). In any case, its rising popularity is sustained by the recent unprecedented technological development.

There are different definitions of smart city in the literature. Anthopoulos and Reddick (2016) define it as *innovation - not necessarily but mainly based on Information and Communications Technologies (ICT)- that enhances urban living in terms of people, governance, economy, mobility, environment and living.* Additionally, they discuss that smart cities are a subset of smart governments. The European Commission (EC) states that *a smart city is a place where traditional networks and services are made more efficient with the use of digital solutions for*

the benefit of its inhabitants and business[1]. In this work, we define *smart city* in a broader sense as an urban settlement where data obtained mainly (but not only) from sensors is used to improve the quality of life of citizens and the natural environment. The main actors involved in smart cities are as follows:

- **Orchestrator**: Stakeholders enabling and deploying smart city applications, including the government, industry, and academia.
- **Society**: Main beneficiary of the smart city ecosystem, who can also partake on its applications (e.g., citizen science).
- **Physical environment**: Urban areas where the smart city applications are deployed. It can also include natural environments that benefit from such applications.
- **Technology**: Sensors that collect data together with the hardware, algorithms, and processes that turn it into valuable information or services.

Smart cities provide both opportunities to improve the lives of citizens and challenges to the scientific community. Advances in big data, IoT, data mining, and DL provide the building blocks for this discipline to flourish. There are different facets where smart city applications can play a defining role. In the following, we provide some examples clustered into categories (see Mohanty et al., 2016):

1. **Infrastructure**: Improving the sustainability and efficiency of the urban infrastructure with the aid of technology. As an example, White et al. (2021) propose a digital twin smart city approach for the area of Dublin (Ireland). The target is urban planning through user feedback. The digital twin allows policymakers and citizens to simulate new buildings, green spaces, floodings, crowd flows, etc.
2. **Buildings**: This category directly relates to the previous one but focuses specifically on city buildings. The work of Vázquez-Canteli et al. (2019) is a good example to illustrate this category. They present a simulation environment for the implementation of buildings' control algorithms. Their purpose is the reduction of energy consumption and carbon dioxide (CO_2) emissions. They use the Keras library (Chollet et al., 2015) to develop the control algorithms.
3. **Transportation**: Optimizing transportation systems through real-time monitoring and intelligent traffic and public transport management. One recurrent problem in this field is efficient parking in cities. To tackle this, Grodi et al. (2016) deploy sensors across parking spots. Working together with a central database, they provide real-time information on free parking spots. The

[1] https://commission.europa.eu/eu-regional-and-urban-development/topics/cities-and-urban-development/city-initiatives/smart-cities_en (last accessed: 2025/03/15)

work by Iqbal et al. (2019) is another example that studies this problem. In this case, several DL-based NNs process images sent wirelessly from cameras. They additionally focus on the power consumption of the sensor network.

4. **Energy**: Integrating renewable energy sources into the city's smart grid. City councils embracing these tools can make their cities more energy-efficient and clean. An example is the deployment of sensors to report the fill level of rubbish bins. The URBAN-WASTE EU project (Gruber et al., 2017) precisely does this. In this way, garbage trucks can optimize their routes dynamically, resulting in energy, pollution, and time savings. Sikder et al. (2018) is another example falling into this category. They analyze how different smart lighting systems can reduce power consumption compared to conventional ones. Energy consumption is a recurrent problem in the field, given the vast networks of sensors employed. The work by Marrero et al. (2019) tackles this problem by developing a portable energy-saving system for wireless networks of sensors.

5. **Healthcare**: Introducing technology to enhance health care services, such as telemedicine or health monitoring devices. As an example, Dbouk et al. (2022) study local airborne pollen transport from trees. They apply Computational Fluid Dynamics (CFD) methodologies to assess the risk of exposure of allergic people in a region of France.

6. **Technology**: As mentioned above, technology is the enabler of all these applications, such as 5G networks (Rao & Prasad, 2018). A problem usually faced in the field is network congestion and imbalance. Zhao et al. (2019) design a deep RL-based routing algorithm to mitigate these issues. Their system can connect to several smart services, and they validate it with numerical experiments.

7. **Governance**: Promoting participatory decision-making and improving public administration services. König (2021) discusses several ways to create value from data in the public sector. Some of them are the publication of open data, the optimization of administrative tasks, and the analysis of data for managerial decisions. Furthermore, the author highlights some ethical and privacy issues to look out for in this context. These include obtaining skewed conclusions from data and analyzing sensitive citizen information.

8. **Education**: Embracing technology for the betterment of education as a public service. The recent COVID-19 pandemic has inevitably accelerated smart education. Gros (2016) describes features and challenges that come with it, such as personalization or student engagement.

9. **Citizens**: Involving citizens in the implementation of smart city applications. One could discuss that, in one way or another, this last category is ubiquitous

to all the others.

The use cases that we study belong to a subset of them, namely: solar energy (smart energy), mobility flow (smart transportation), and air pollution (smart healthcare). These three topics directly contribute to several Sustainable Development Goals (SDGs)[2]. Specifically, they align with SDG 7 (affordable and clean energy), SDG 11 (sustainable cities and communities), and SDG 13 (climate action), respectively. Anticipating challenges related to these topics enables policymakers to make informed decisions that positively impact citizens. This thesis tackles these issues by leveraging sensor observations to develop generic predictive models based on DL. In the subsequent sections, we present related work regarding data, visualization, and forecasting.

2.2 Solar energy

Clean energy harnessing and management has become critical for sustainable development. Solar Photovoltaic (PV) systems have become a highly popular source of renewable energy and are currently experiencing the fastest growth rate in the European Union (EU) (*EU Solar Energy Strategy*, 2022). This fact has increased the interest in solar irradiance forecasting, evidenced by the substantial increase in publications on the topic over the past decade (Yang, 2019). This section outlines different open datasets and initiatives to harness solar irradiance (Section 2.2.1). Afterward, we discuss forecasting approaches in Section 2.2.2.

2.2.1 Data and visualization

The solar energy that can be collected at a certain location depends directly on the Global Horizontal Irradiance (GHI). GHI is the total amount of shortwave radiation received by a horizontal surface. It can be expressed in terms of the Diffuse Horizontal Irradiance (DHI) and the Direct Normal Irradiance (DNI), as we will describe in Section 5.1.2. Solar irradiance data can mainly be obtained from ground sensors or satellite observations. Satellites can measure DHI and DNI to estimate GHI on the Earth's surface. Regarding ground sensors, pyranometers and pyrheliometers are typically used to collect GHI and DNI, respectively (Arbizu-Barrena et al., 2017). It is important to note that temporal resolution tends to be much finer for ground sensors (seconds to minutes) compared to satellite images (quarters of an hour to hours). However, satellites offer broader spatial coverage. Since ground measurements are closer to what solar panels receive on the Earth's surface, they provide a more accurate proxy for solar irradiance forecasting. Therefore, the ideal scenario for collecting solar irradiance data involves having a network of sensors, such as pyranometers, distributed across the Region Of Interest (ROI) with sufficient spatial and temporal granularity.

[2] https://sdgs.un.org/goals (last accessed: 2025/03/15)

Various initiatives collect and distribute solar irradiance data. Sheffield Solar, University of Sheffield (2024) provides "historical, real-time and forecasted solar PV output data at national and regional level". Their Application Programming Interface (API) offers access to solar data covering Great Britain (GB) and includes a Python implementation[3]. In this example, the temporal granularity ranges between 5 to 30 minutes. However, their spatial coverage is based on a set of predefined regions of GB, which may be insufficient for fine-grained spatial forecasting.

The Spanish Agencia Estatal de Meteorología (AEMET) provides an API for obtaining meteorological and climatic data, including irradiance values and weather predictions (AEMET, 2024). Users need to request an API key to access this data[4]. The InfoRiego weather network from the Instituto Tecnológico Agrario de Castilla y León is also located in Spain (Instituto Tecnológico Agrario de Castilla y León, 2024). This network comprises 56 meteorological stations equipped with Campbell Skye SP1110 pyranometers that collect GHI data every 30 minutes. Accessing their API also requires an API key.

Natural Resources Canada (2024) offers a dataset for fine-grain temporal analysis. GHI is collected at a high frequency: between 30 and 100 Hz (or up to once every ten milliseconds). They gather these data at two locations equipped with 17 and 24 LI-COR LI-200S pyranometers, respectively. Solar irradiance is available for four types of days based on cloud cover: clear sky, overcast, variable, and very variable. Additionally, the Oahu Solar Measurement Grid dataset from the National Renewable Energy Laboratory (NREL) provides solar data from Oahu, Hawaii (Sengupta & Andreas, 2010). It includes nearly 20 months of measurements from 17 LICOR LI-200 pyranometers, which collected GHI values daily between 5 am and 8 pm UTC-10 at 1 Hz. This dataset provides fine spatial and temporal granularities, and we will study it as a use case in Chapter 5.

Access to solar irradiance data is highly valuable, and effective visualization tools enhance its utility. The EC's Joint Research Centre offers Photovoltaic Geographical Information System (PVGIS): a free and open access online tool that includes an API (Joint Research Centre, 2024). Users can obtain data on photovoltaic performance, solar irradiance, and typical meteorological years. Granularities and sources (satellite or reanalysis) vary based on the desired geolocation. Similarly, the POWER project by the National Aeronautics and Space Administration (NASA) provides solar and meteorological data to support renewable energy efforts (NASA Langley Research Center, 2024). The tool is well documented, providing an API alongside visualization and graphing capabilities. In contrast, many other tools in this domain often require a paid subscription.

[3] https://github.com/SheffieldSolar/PV_Live-API (last accessed: 2025/03/15).
[4] https://opendata.aemet.es/dist/index.html (last accessed: 2025/03/15).

2.2.2 Forecasting

This section discusses several approaches to solar irradiance forecasting.

Classical approaches

The main classical techniques used in the literature are based on Numerical Weather Prediction (NWP) models, satellite images, or statistical methods. For example, Arbizu-Barrena et al. (2017) use the cloud index to forecast solar irradiance with the aid of an NWP model. Differently, Ayet and Tandeo (2018) employ geostationary satellite images to predict solar irradiance.

A different strategy involves using irradiance measurements from PV plants to develop statistical models. Amaro e Silva and Brito (2018) conducted a study employing Auto Regressive models with eXogenous inputs (ARX) models, showing that solar data from neighboring locations can enhance short-term forecasts (up to a few minutes). However, they found that this effect diminishes for further forecast horizons. Chapter 5 delves deeper into this concept using irradiance maps. In addition, we carry out a spatial analysis similar to that of Amaro e Silva and Brito (2018).

ML and DL approaches

Besides the previously mentioned methodologies, ML approaches, particularly DL techniques, are gaining attention in solar irradiance forecasting. The use of NNs for this purpose is already present in the work by Hontoria et al. (2002).

More recently, Alzahrani et al. (2017) employed Long Short-Term Memory (LSTM) networks for high-resolution forecasting (100 Hz) at a single location, achieving an Root-Mean-Squared Error (RMSE) of 0.086. Their dataset included four days with varying weather conditions: clear sky, few clouds, scattered clouds, and overcast. Similarly, Qing and Niu (2018) leveraged the temporal aspect of solar forecasting using LSTMs. Their results are between 18.3 % and 42.9 % more accurate than the Multilayer Perceptron (MLP). However, their time granularity was relatively coarse and the horizon was far, which limits its applicability to PV systems. Their proposal predicts hourly GHI for the next day at a single location using hourly weather data (exogenous inputs) from the given day. Martín Otero (2018) also utilized LSTMs, but focuses on short-term predictions ranging from 1 to 20 minutes. The model was trained on data from several pyranometers located at the Plataforma Solar de Almería (PSA), Spain.

Wang et al. (2018) include PV data besides weather information, feeding them into a Gated Recurrent Unit (GRU) network, which operates similarly to LSTMs. They first analyzed the correlation between the PV power at a given instant and the n previous ones. Then, they used the K-means algorithm to cluster the training data based on features. Subsequently, they trained multiple GRU networks, one

for each cluster, consistently achieving an RMSE of approximately 0.07.

Temporal input window and scalability

When considering the time-dependent elements of solar forecasting, it is important to emphasize the width of the temporal input and output windows. Time resolution also plays a significant role, as higher frequencies introduce greater variability.

The main contributions of Galicia et al. (2019) include the development of an ensemble of various ML models and analyzing the most suitable timestep for two specific use cases. The ensemble predicts the next 20 time instants, sampled every 30 minutes, by dividing the task into as many sub-problems as there are forecast horizons. Furthermore, they focus on the scalability of their system to handle big data, utilizing the MLlib library within the Apache Spark framework. Scalability is a critical feature when dealing with sensor data forecasting problems.

Spatial component

While most studies address the solar forecasting problem primarily through its temporal component, Li et al. (2019) attempt to harness the spatial aspect of solar irradiance forecasting. They forecast 1-hour ahead irradiance for a single target station based on historical data from surrounding sites. To achieve this, they train four echo state networks, one for each season, achieving forecast accuracy improvements that range from 1.8 % in summer to 18.6 % in autumn. Other works that incorporate spatial attributes, such as Arbizu-Barrena et al. (2017) and Ayet and Tandeo (2018), typically rely on satellite images. However, these methods can be prone to errors due to factors like soil albedo, aerosols, and dust.

Novel DL approaches

Current trends in solar energy production require the flexible and robust management of multiple PV data sources simultaneously, an aspect that previous works have not adequately addressed. This gap is the focus of our work. In related fields, experts are already leveraging hybrid DNNs. For example, Huang et al. (2020) employ a combination of Variational Mode Decomposition, Convolutional Neural Networks (CNNs), and GRUs for electricity price forecasting. Some authors also incorporate ideas from LSTMs or CNNs into novel approaches, such as Graph Neural Networks (GNNs) (Simeunović et al., 2021; Yang et al., 2025). Other recent DNNs incorporate LSTM units into convolutional layers, known as Convolutional LSTM (ConvLSTM) networks. The seminal works by Shi et al. (2015) and Agrawal et al. (2019) on precipitation forecasting exemplify this approach. As detailed in Chapter 5, our work utilizes these types of networks, translating solar forecasting from an individual time series prediction task into an image-to-image transformation problem.

2.3 Mobility flow

Sensible distribution and optimization of mobility elements are critical in city planning. There is a growing interest in understanding urban mobility dynamics, as will be discussed in this section. Similar to the previous Section 2.2, Section 2.3.1 will first explore open datasets related to mobility and initiatives to visualize them. Following this, Section 2.3.2 will delve into the topic of modeling these types of data.

2.3.1 Data and visualization

The Empresa Municipal de Transportes de Madrid manages the city's docked electric bicycles and publishes usage data monthly (EMT Madrid, 2023). In 2023, the municipality expanded this service from 3000 to 7500 bikes and 264 to 611 docks. Similarly, Lyft publishes monthly trip data for their docked bikes in various United States of America (USA) cities, such as New York City (NYC) (Lyft, 2024a) and Chicago (Lyft, 2024b). More ambitious platforms aim to provide APIs for multiple bike-related open datasets. For instance, the Open Bike Share Data makes several datasets available under the Open Database License (ODbL) (Bike Share Research contributors, 2021). In a broader context, the Mobility Database compiles a catalog of many mobility databases across over 60 countries (Mobility Data, 2024). These datasets aim to facilitate access to mobility options for travelers and app developers.

Electric scooters are a relatively new shared mobility option widely available in many cities. For example, City of Chicago (2021) provides e-scooter trip data from pilot studies conducted in 2019 and 2020. Additionally, companies such as Lime[5] and Spin[6] claim to provide data upon request.

Taxis are another important component of urban mobility. The NYC Taxi and Limousine Commission has provided trip data since 2009 (NYC Taxi & Limousine Commission, 2024). This dataset covers a much larger area than Lyft bikes (Lyft, 2024a), including boroughs such as Staten Island and parts of Queens. Additionally, data is disaggregated into different taxi categories: yellow, green, For-Hire Vehicles (FHVs), and high-volume FHVs. Similarly, Chicago offers taxi trip data through their data portal (City of Chicago, 2024), and since 2018, they have also incorporated rideshare companies (FHVs). Furthermore, there are ongoing efforts to study visualization techniques. For example, using mobile phone data, Ma et al. (2015) evaluated several visualization methods in Shenzhen, China.

[5] https://www.li.me/legal/research (last accessed: 2025/03/15).
[6] https://web.spin.pm/data-requests (last accessed: 2025/03/15).

2.3.2 Forecasting

According to Xie et al. (2020), urban mobility flow forecasting can be divided into three subfields: crowd flow (human mobility), traffic flow (vehicle congestion), and public transport flow (taxi, bicycle, bus, metro, etc.). The methods used to address these tasks can be categorized into statistics-based, machine learning-based, deep learning-based, and reinforcement learning-based approaches. Our study from Chapter 6 focuses on forecasting public transit flow using DL-based methods. However, since the prediction of human mobility and traffic flow are closely related, these topics are also discussed in this section.

Statistics-based methods

Early works on mobility demand approached it as a time series forecasting task. Li et al. (2012) used ARIMA to forecast taxi pick-ups in specific hotspots from Global Positioning System (GPS) trajectories. Moreira-Matias et al. (2012) also employed ARIMA, combining it with time-varying Poisson averaged models to create an ensemble approach. Seasonal ARIMA (SARIMA), an extension of ARIMA, was utilized by Zhang et al. (2011) to predict short-term traffic flow.

Other methods rely on probability distributions to predict future mobility demand. For example, Ma et al. (2013) proposed a taxi scheduling system responding to real-time requests from pedestrians simulated using a Poisson distribution. Similarly, Deri et al. (2016) estimated taxi trajectories in NYC, emphasizing job parallelization.

Miao et al. (2016) tackled the issue of taxi dispatching and the *problem of imbalance*, where taxis cluster in certain areas, neglecting others. Their framework combined historical and real-time GPS and occupancy data with a receding horizon control approach. Taxi dispatching and imbalance are closely related to crowd flow management. Ma et al. (2015) conducted temporal correlation analysis to address crowd flow. These methods have been effective for many time series forecasting tasks. However, they fall short in capturing the spatial dependencies inherent to urban mobility flow.

Machine learning-based methods

Many ML-based algorithms are employed in the field, similar to statistical methods. The early work of Li et al. (2012) utilized a simple Bayesian network for predicting taxi demand. More recently, Roos et al. (2017) also employed Bayesian networks, albeit for metro passenger flow forecasting. They outperformed the historical average and handled missing data using a structural expectation-maximization algorithm.

Markov random fields can capture dependencies that Bayesian networks may not, as demonstrated by Hoang et al. (2016). In contrast, Habtemichael and Cetin

(2016) trained a k-nearest neighbors algorithm (k-NN) for traffic flow forecasting. They conducted thorough experiments on 36 datasets from the United Kingdom (UK) and USA, achieving better results than parametric models. Lippi et al. (2013) compared a SARIMA model, which includes a Kalman filter, with a support vector machine using a radial basis function kernel for traffic flow prediction, obtaining similar results. Although the SARIMA model scored better on average, the authors suggested that the support vector machine approach could be a better compromise between accuracy and computational cost.

ML approaches have significantly boosted research progress in the field. However, the availability of vast amounts of heterogeneous data from various sources and sensors limits the use of standard ML-based methods, which are increasingly being relegated to baselines in favor of more advanced DL-based approaches.

Deep learning-based methods

Recent work on mobility flow prediction has increasingly shifted towards DL-based methods. For example, Polson and Sokolov (2017) developed a spatio-temporal DNN for traffic flow prediction, studying its performance during special events like football games. Similarly, Zhang, Zheng, et al. (2016) presented a DL-based spatio-temporal model and tested it on several datasets. Their model incorporated a component to capture global factors, such as the day of the week. As the authors highlight, the quality and quantity of spatio-temporal data is crucial.

The technique known as *transfer learning* addresses data scarcity when training DL models, where an existing model serves as a base for a related task. For example, Wang et al. (2019) applied this idea for bike flow prediction. They used data from Washington, D.C.; Chicago; and NYC as source and target regions. Tian et al. (2021) followed a similar approach, integrating multiple source cities for traffic flow tasks.

Some novel DNNs are also used for related tasks. For example, Ma et al. (2018) employed a convolutional Bidirectional LSTM (BiLSTM) for metro traveler prediction. Another example is the ConvLSTM. As discussed in Section 2.2.2, this architecture combines the LSTM ability to remember long-term dependencies with the CNN's capacity to find spatial correlations. In Chapter 6, we leverage the ConvLSTM network, translating mobility demand forecasting from an individual time series prediction task into an image-to-image transformation problem. Specifically, we focus on taxi and bicycle trips, the related work of which we discuss further hereafter.

Deep learning-based methods for taxi demand prediction

Taxi demand is a well studied problem in the field. For example, Xu et al. (2017) developed LSTM networks alongside mixture density networks for predicting taxi demand in NYC zones using historical demand data and additional information,

such as weather, time, and taxi drop-offs. The city was divided into numerous small regions, all of which served as input to a single model that predicted the probability distribution of taxi demand. Thus, the model was bound to a specific number of areas. Their results are compared to the proposed approach in the relevant use case (Chapter 6).

Similarly, one of the architectures proposed by Rodrigues et al. (2019) utilized LSTMs for taxi demand prediction, with the novel integration of textual information fused with the LSTM output. Yao et al. (2018) go one step beyond, capturing spatial dependencies among nearby regions using convolutional layers over several timesteps. The result is then fed into an LSTM network and concatenated with the output of a so-called semantic view.

The recent work by Liu, Wu, et al. (2020) employed a context-aware attention mechanism to integrate three different predictions: an instant spatio-temporal module (using one-dimensional convolutions and GRU cells), a short-term module covering four days, and a long-term periodic module spanning two weeks. Their experiments demonstrated improved error when progressively combining these modules. We further examine the results of their work in Chapter 6.

Deep learning-based methods for bicycle demand prediction

Bicycle demand modeling is also increasingly adopting DL strategies. Jiang (2022) provided a comprehensive survey of works in this area that utilize DL. The author categorized different prediction problems based on whether the data are treated as time series, graph, or grid formats. Additionally, the survey classified the evaluated works according to the use of exogenous data (e.g., weather, points of interest, and calendar information), the software frameworks employed, and the data availability.

Chai et al. (2018) modeled the problem as a bike-sharing system graph, tested for Lyft bikes in Chicago and NYC. They used multi-graph convolutions with an input window comprising the last six hours to predict bike flow for the next hour. Chapter 6 discusses their results in further detail. Lin et al. (2018) also predicted station-level hourly bike demand by means of GNNs in NYC. In addition to incorporating recurrent information, they explored how to capture pairwise correlations between stations to enhance prediction accuracy. Another example of GNNs applied to bike demand forecasting is the work of Li et al. (2024). In this case, the authors combine two spatio-temporal convolutional blocks with information extracted from points of interest.

In contrast, Li et al. (2021) addressed the problem using a grid-based data structure and examined four use cases, including Chicago. They capture temporal aspects with three independent modules (closeness, period, and trend), which are subsequently fused. Similar to our approach, they employ a ConvLSTM-based

model. However, they do not incorporate weather nor calendar information, and their temporal and spatial granularity is coarser (2×2 kilometers and one hour, respectively). Moreover, none of these works integrate different mobility services into a single model, a key aspect addressed in Chapter 6.

Reinforcement learning-based methods

RL methodologies are also gaining attention in mobility flow forecasting. Jiang et al. (2018) utilized these techniques to optimize the number of metro passengers during rush hours. They simulated their effect on the Shanghai metro line, reducing the number of passengers stranded on the platform. Similarly, Khaidem et al. (2020) employed RL to predict human mobility in three real cities and one synthesized city. Walraven et al. (2016) used RL to reduce traffic congestion while taking into account future traffic predictions. They employed Q-learning to establish the maximum driving speed allowed on highways. This work exemplifies how RL-based methods can take advantage of NNs and, more broadly, DL.

In his thesis, Escribá Pina (2021) conducted a systematic review of RL applications in smart cities. The key use cases the author identifies are the management of resources, telecommunication networks, personal assistants, and urban mobility. He also developed two agents: one for the pick-up and drop-off of passengers and another for efficient waste collection. Another example of RL is the work by Wei et al. (2018), where a deep RL-based model was created to optimize traffic light operation, aiming to reduce waiting times. Additionally, they raised an important question: how can a RL reward function be designed to be fair for all actors involved in mobility (pedestrians, bikes, scooters, cars, etc.)?

2.4 Air pollution

Air quality plays a vital role in both environmental sustainability and human health, as it directly impacts ecosystems and our well-being. Section 2.4.1 provides an overview of tools and initiatives for collecting, analyzing, and visualizing air pollution data. This research briefly examines potential use cases to evaluate the developed framework. Following that, Section 2.4.2 explores several studies that utilize these resources to model air quality.

2.4.1 Data and visualization

Air quality has become a central issue for health institutions and governmental agencies alike (European Environment Agency et al., 2020; World Health Organization, 2021). As a result, extensive efforts are underway to boost air quality research. A foundational step in this process involves establishing infrastructure capable of monitoring various pollutants across multiple locations over time. The collected data can then be stored in repositories for further analysis. Regional (Ayuntamiento de Madrid, 2024; Gobierno de Canarias, 2023), national (Ministerio

para la Transición Ecológica y el Reto Demográfico, 2023; Ministry of Environment, Taiwan, 2024), and global initiatives (European Environment Agency, 2024a; OpenAQ, 2024) are already contributing to this effort. Notably, the nonprofit organization OpenAQ (2024) offers an API that provides open access to global air quality data for seven key pollutants. To maximize the utility of these repositories and support effective data curation, they should include essential details such as: temporal coverage, sensor geolocation, pollutants measured, measurement units, measurement methods, measurement equipment, and approaches to handling missing data.

Thereupon, these data can be transformed into valuable information to better understand pollution dynamics. For instance, the World Health Organization (2022c) offers national-level maps showing fine particulate matter ($PM_{2.5}$) concentrations over several years. Other initiatives aim for more detailed spatial granularity, collecting data at the sensor level. European Environment Agency (2024b), for example, provides a web application that allows users to visualize contaminants detected by sensors across Europe. Additionally, (European Environment Agency, 2024d) focuses on $PM_{2.5}$, and (European Environment Agency, 2024c) covers more pollutants and provides an Air Quality Index (AQI).

Since the outbreak of the COVID-19 pandemic, many people's habits have changed, leading to noticeable impacts on pollution trends. In response, the European Environment Agency (EEA) has maintained an interactive visualization tool that displays weekly and monthly concentrations of $PM_{2.5}$, coarse particulate matter (PM_{10}), and nitrogen dioxide (NO_2) in European cities (European Environment Agency, 2023). Numerous studies have examined the impact of COVID-19 on air pollution, with some analyzing satellite images (Dutheil et al., 2020). In contrast, Berman and Ebisu (2020) used sensor data from the USA to explore this correlation. Additionally, Venter et al. (2020b) combined satellite images with ground sensor data to investigate this phenomenon on a global scale, offering a visualization tool (Venter et al., 2020a). Furthermore, other works have studied how air pollution favors the spread of coronaviruses. As examples, the work of Fattorini and Regoli (2020) focuses on Italy and that of Travaglio et al. (2021) on England.

2.4.2 Forecasting

Once air quality data is available and analyzed, the next challenge is modeling it to mitigate air pollution. This section discusses some classical and novel approaches to air quality forecasting.

Classical approaches

According to Mao et al. (2021), air quality forecasting models can be broadly classified into physical and statistical models. While physical models are based on well-established classical methods (Holmes & Morawska, 2006; Wang et al., 2001),

they are often more computationally intensive than statistical models (Zlatev & Dimov, 2006). Due to these computational demands, along with advancements in Artificial Intelligence (AI) (e.g., Fairfax et al., 2023; Lopez-Gomez et al., 2023; Roy & Bhaduri, 2023), statistical models are increasingly favored in recent research, including our study. The remainder of this section will explore several AI-based statistical models for air quality forecasting and their particularities.

Deep learning-based methods

DNNs are widely recognized as a versatile tool for addressing various forecasting tasks, including air pollution prediction. As a first example, Huang and Kuo (2018) utilize a combination of several one-dimensional convolutions and an LSTM network to predict $PM_{2.5}$ levels an hour in advance. They focus on Beijing, addressing the problem from a smart city perspective. Their model also integrates wind speed and rainfall data. Similarly, Wen et al. (2019) predict $PM_{2.5}$ levels both locally and globally in Beijing and across China. In this case, they use a hybrid approach that combines CNNs, LSTMs, and k-NN. Their model incorporates meteorological data and other physical parameters as well.

In another study, Qi et al. (2019) pursue the same goal but introduce a graph structure, employing regional graph convolutions for their predictions. A more recent example of graph-based approaches for $PM_{2.5}$ forecasting is the work of Ouyang et al. (2023). Meanwhile, Zhou et al. (2019) focus on a finer spatial resolution to forecast $PM_{2.5}$, PM_{10} and nitrogen oxides (NO_x) levels in Taipei, Taiwan, up to four hours ahead. To do so, they develop a Deep Multi-output LSTM neural network, which consists of an MLP with LSTM neurons. Similarly, Drewil and Al-Bahadili (2022) concentrate on predicting $PM_{2.5}$, PM_{10} and NO_x using LSTM methods. Nevertheless, they extend their study to include carbon monoxide (CO) predictions, employing a genetic algorithm to optimize model hyperparameters for next-day forecasts. Finally, Devasekhar and Natarajan (2023) propose a method to forecast air quality by combining supervised ML with multi-agent systems. However, they do not explore the use of meteorological variables.

Multiple sensors and spatial resolution

Several studies have explored the challenge of forecasting air quality across multiple locations. For instance, Mao et al. (2021) developed a framework for predicting air quality over the next 24 hours. It comprises a neural network with a temporal sliding LSTM extended model. They trained individual models for 12 cities, encompassing 65 stations that collect $PM_{2.5}$ data. The model incorporates meteorological data, as well as seasonal and monthly information encoded as one-hot vectors. Their study also examines the impact of long-term prediction horizons (up to 72 hours) on model performance. Their findings indicate that the optimal time lag increases with the forecast horizon.

In contrast, Liu et al. (2022) approach the problem by combining temporal predictions of an AQI with spatial inference for unknown locations. In their model, the AQI is determined by the highest concentration of the available pollutants, that is, ozone (O_3), $PM_{2.5}$, PM_{10}, CO, sulfur dioxide (SO_2), and NO_2. They also include meteorological variables and spatial attributes like population density, road networks, and elevation. Their results indicate that longer forecast horizons generally lead to decreased model performance across all seasons. We further analyze this behavior in Chapter 7.

Additionally, de Medrano et al. (2021) focus on the city of Madrid, Spain, which is the same location as our air quality use case. They forecast NO_2, O_3, $PM_{2.5}$, and PM_{10} for the next 48 hours. They develop a chain of models that incorporate exogenous factors, such as weather and anthropogenic features. However, each air pollutant and station is modeled separately, with the predictions later combined using Bayesian estimation. We will compare some of their results to the proposed framework in Chapter 7.

Achieving a fine spatial resolution in air quality forecasting often requires advanced interpolation techniques. Meteosim (2021) addresses this challenge by developing three platforms tailored to different air pollution-related tasks: (1) gas dispersion simulation, (2) meteorological forecasting, and (3) air quality and dust forecasting for up to 72 hours. Additionally, they create tools to monitor threshold exceedances and identify sources of unpleasant odors. Their platforms achieve a spatial resolution of 1 km for meteorological predictions and 250 meters for air quality forecasts.

2.5 Non-Functional Features

The previous sections discussed related work that focuses on measuring spatio-temporal sensor data, as well as work that uses such data to build forecasting models. In this context, comparing related models is often based primarily on error metrics. While this approach is straightforward and fair when comparing models with similar specifications (such as training datasets and forecast horizons), it overlooks other important features beyond error metrics.

In software development, *non-functional requirements* refer to attributes or properties that a software system exhibits, beyond its core functionality (Glinz, 2007; Pressman, 2005). Such characteristics are also relevant when dealing with spatio-temporal sensor data, and we refer to them as NFFs. Despite their importance, these attributes are often overlooked in the literature. Examples of such characteristics include portability, scalability, and time complexity. For instance, Marrero et al. (2019) examine the portability of a wireless sensor network, enabling its deployment across various locations. Galicia et al. (2019) focus on the scalability property of their ensemble model for time series forecasting. Related to this, Deri et al.

(2016) experiment with a parallelization framework to reduce computation time when estimating taxi trajectories. These generalization capabilities are crucial for real-world, dynamic scenarios.

Chapter 3 will delve deeper into the concept of NFFs. To introduce the topic further, however, we provide additional examples below. Section 2.4.2 discusses related work in the field of air quality forecasting, some of which exhibit advantageous properties beyond their primary functionality. Specifically, we seek to answer the following questions:

- **Multivariable**: Can the model be extended to consider multiple related variables measured by the sensors?
- **Extensive data**: Is the model built using extensive real-world data?
- **Exogenous data**: Can the model naturally integrate additional input data beyond the sensor-provided data to improve its forecasts?
- **Sensor-agnostic**: Can the model operate despite changes in the set of sensors? A sensor-agnostic model offers:
  - **Flexibility**: The ability to work with a varying set of sensors (in terms of both number and geographic distribution).
  - **Robustness**: The capacity to recover from sensor failures and continue providing reliable forecasts.

Table 2.1 summarizes the relationship between these features and the related work discussed in Section 2.4.2. As we will explore further in Chapter 7, our air quality forecasting model meets all these criteria.

Table 2.1: NFFs in related work on air quality forecasting.

Reference	multiple pollutants	extensive data	exogenous data	sensor-agnostic
Huang and Kuo (2018)	×	∼	×	×
Qi et al. (2019)	×	×	×	×
Wen et al. (2019)	×	×	∼	×
Zhou et al. (2019)	✓	✓	✓	×
de Medrano et al. (2021)	∼	✓	✓	×
Mao et al. (2021)	×	✓	✓	∼
Meteosim (2021)	×	?	?	∼
Drewil and Al-Bahadili (2022)	✓	∼	×	×
Liu et al. (2022)	∼	×	∼	✓
Devasekhar and Natarajan (2023)	×	×	✓	×

✓: Covered. ×: Not covered. ∼: Partially covered. ?: Unknown.

2.6 Summary

This chapter presents a summary of the related work relevant to this thesis. First, Section 2.1 highlights the importance of sensors in smart city applications. We then examine spatio-temporal sensor data and state-of-the-art forecasting approaches. Such data is often scarce, particularly when targeting fine spatial and temporal resolutions. Consequently, we identify and compile valuable data repositories and tools for each of the three use cases studied in Sections 2.2.1, 2.3.1, and 2.4.1, respectively. These resources act as proxies for spatio-temporal sensor data modeling, which we review in each case (Sections 2.2.2, 2.3.2, and 2.4.2). Lastly, we explore NFFs identified in related work, highlighting gaps that this thesis seeks to address with its contributions.

Sensors, communication networks, and algorithms are the main ingredients for developing smart city initiatives. Section 2.1 introduces this topic and its key players: orchestrators, societies, physical environments, and technologies. It also provides several examples organized into categories. Table 2.2 compiles these examples along with their fields of application. The use cases studied in this thesis cover energy, transportation, and healthcare, aligning with the SDGs 7, 11, and 13.

Table 2.2: Related work regarding smart city applications (Section 2.1).

Topic	**Reference**
Infrastructure	White et al. (2021)
Buildings	Vázquez-Canteli et al. (2019)
Transportation	Grodi et al. (2016)
	Iqbal et al. (2019)
Energy	Gruber et al. (2017)
	Sikder et al. (2018)
	Marrero et al. (2019)
Healthcare	Dbouk et al. (2022)
Technology	Rao and Prasad (2018)
	Zhao et al. (2019)
Governance	König (2021)
Education	Gros (2016)

Table 2.3 compiles all the relevant open datasets covered in this chapter, drawn from Sections 2.2.1, 2.3.1, and 2.4.1. The first use case we examine is solar irradiance forecasting. Section 2.2.1 provides several data resources for this field, including the specific dataset by Sengupta and Andreas (2010) that we use in Chapter 5. While the number of sensors and the covered ROI are relatively small (see the upper section of Table 2.3), these aspects are explored further in the other two use cases. We selected this dataset for our first use case due to its fine spatial and temporal

resolution. The second use case we investigate is mobility demand in urban areas. Section 2.3.1 collects several open datasets related to this topic, gathered in the middle section of Table 2.3. For this use case, we select a scenario with a large ROI and a significant number of sensors, incorporating both taxi and bicycle ride data (City of Chicago, 2024; Lyft, 2024b). The third use case we explore is air quality forecasting. Section 2.4.1 presents several open datasets related to this topic, summarized in the lower section of Table 2.3. Each dataset collects different pollutants, typically including $PM_{2.5}$, PM_{10}, O_3, NO_2, NO_x, and CO. Our study uses a decade's worth of data from Ayuntamiento de Madrid (2024). Although the ROI and number of sensors are smaller than in the second use case, we touch upon up to eleven pollution variables.

Building on the work related to solar irradiance data and visualization, Section 2.2.2 discusses various forecasting approaches (see Table 2.4). Amaro e Silva and Brito (2018) inspired us to investigate the relationship between forecasting error and the relative positions of the sensors. Additionally, Shi et al. (2015) and Agrawal et al. (2019) are among the pioneering studies that introduced ConvLSTMs, which are a key component of our forecasting model. As detailed in Chapter 5, our forecasting approach provides some NFFs while achieving low error levels. In particular, we examine the flexibility and robustness of our method, aspects that are not addressed by the works listed in Table 2.4.

Section 2.3.2 provides a comprehensive overview of research focused on enhancing urban mobility (see Table 2.5). Our approach falls within the category of forecasting public transit flow using DL-based methods. Few works listed in Table 2.5 propose frameworks for integrating heterogeneous data sources into a unified structure for mobility forecasting. Moreover, none of these studies explore flexibility or robustness NFFs, focusing mostly on error metrics. In Chapter 6, we aim to address these gaps by contributing to both dimensions, beyond exceeding baseline metrics.

Section 2.4.2 reviews the diverse modeling approaches for air quality forecasting (see Table 2.6). As with the previous use case, we compare our approach to related work, focusing on both error metrics and NFFs, including the work by de Medrano et al. (2021) (see Chapter 7). Table 2.6 compiles these works, primarily focusing on minimizing the error. Given the range of pollutants, a versatile forecasting tool capable of simultaneously addressing multiple pollutants is crucial. However, most studies either focus on a few pollutants or address them separately.

Finally, Section 2.5 provides an overview of how other works approach NFFs. As discussed, we find that comparing models solely based on error metrics is insufficient. While some studies aim to achieve these beneficial properties, only a few successfully do. For example, Table 2.1 highlights related work on air quality and the specific features they achieve. This thesis develops a forecasting approach targeting these NFFs and evaluates its performance on the discussed use cases.

Table 2.3: Open datasets related to spatio-temporal sensor measurements from the three studied use cases, discussed in Section 2.2.1, Section 2.3.1, and Section 2.4.1. The datasets we use are highlighted in bold.

Use case		Reference	Location	Temporal ganularity	Number of sensors	Area of ROI [km^2]
Solar irradiance		Sheffield Solar, University of Sheffield (2024)	GB	5-30 min	n/a	243 610
		AEMET (2024)	Spain	1 h	811	505 990
		Instituto Tecnológico Agrario de Castilla y León (2024)	Castilla y León, Spain	30 min	56	94 225
		Natural Resources Canada (2024)	Varennes & Alderville, Canada	≥10 ms (30-100 Hz)	17 & 24	0.25 & 0.1
		Sengupta and Andreas (2010)	Oahu, Hawaii, USA	1 s	17	1
		Joint Research Centre (2024)	worldwide	1 h	n/a	worldwide
		NASA Langley Research Center (2024)	worldwide	1 h	n/a	worldwide
Mobility demand	Taxi	NYC Taxi & Limousine Commission (2024)	NYC, USA	1 s	263	790
		City of Chicago (2024)	Chicago, USA	15 min	801	630
	Bicycle	EMT Madrid (2023)	Madrid, Spain	1 s	264-611	150
		Lyft (2024a)	NYC, USA	1 s	1500	250
		Lyft (2024b)	Chicago, USA	1 s	800	630
		Bike Share Research contributors (2021)	Multiple	n/a	n/a	n/a
	Other	City of Chicago (2021)	Chicago, USA	1 h	77	630
		Mobility Data (2024)	Multiple	n/a	n/a	n/a
Air quality		Gobierno de Canarias (2023)	Canary Islands, Spain	1 h	58	7500
		Ayuntamiento de Madrid (2024)	Madrid, Spain	1 h	24	340
		Ministerio para la Transición Ecológica y el Reto Demográfico (2023)	Spain	1 h	∼600	505 990
		Ministry of Environment, Taiwan (2024)	Taiwan	1 h	77	36 197
		European Environment Agency (2024a)	Europe	1 h	7524	6.5 million
		OpenAQ (2024)	worldwide	1 h	n/a	worldwide

Table 2.4: Related work regarding solar irradiance forecasting (Section 2.2.2).

Reference	Summary
Hontoria et al. (2002)	Early example of NN for the irradiance forecasting case.
Shi et al. (2015)	ConvLSTM for precipitation forecasting with radar images.
Arbizu-Barrena et al. (2017)	Satellite images, pyranometers, and pyrheliometers.
Alzahrani et al. (2017)	LSTM for high frequency forecasting (four days).
Ayet and Tandeo (2018)	Geostationary satellite images.
Amaro e Silva and Brito (2018)	ARX models with data from Sengupta and Andreas (2010).
Martín Otero (2018)	LSTM for short-term forecasts from ground sensors.
Galicia et al. (2019)	ML ensemble looking for the best timestep and scalability.
Li et al. (2019)	Echo state networks with data from neighbor stations.
Agrawal et al. (2019)	ConvLSTM for precipitation forecasting with radar images.
Huang et al. (2020)	CNN with GRU for electricity price forecasting.
Simeunović et al. (2021)	GNNs combined with LSTMs, CNNs, and transformers.
Yang et al. (2025)	GNN, temporal CNN and attention for PV power.

Table 2.5: Related work regarding mobility demand forecasting (Section 2.3.2).

Methods	Crowd flow	Traffic flow		Public transit flow
Statistics-based	Ma et al. (2015)	Zhang et al. (2011) Deri et al. (2016)		Li et al. (2012) Moreira-Matias et al. (2012) Ma et al. (2013)
ML-based	Hoang et al. (2016)	Habtemichael and Cetin (2016) Lippi et al. (2013)		Zhang, Pan, et al. (2016) Roos et al. (2017) Boufidis et al. (2020) Qian et al. (2020)
DL-based	Zhang, Zheng, et al. (2016) Wang et al. (2019)	Polson and Sokolov (2017)	Taxi	Xu et al. (2017) Yao et al. (2018) Rodrigues et al. (2019) Liu, Wu, et al. (2020)
			Bike	Chai et al. (2018) Li et al. (2021) Jiang (2022) Li et al. (2024)
RL-based	Khaidem et al. (2020)	Walraven et al. (2016) Wei et al. (2018)		Jiang et al. (2018) Escribá Pina (2021)

Table 2.6: Related work regarding air quality forecasting (Section 2.4.2).

Reference	Summary
Huang and Kuo (2018)	CNNs and LSTMs for $PM_{2.5}$ (1 hour).
Qi et al. (2019)	Regional graph convolutions and LSTM for $PM_{2.5}$.
Wen et al. (2019)	CNNs, LSTMs and k-NN for $PM_{2.5}$ with exogenous data.
Zhou et al. (2019)	MLP and LSTM for $PM_{2.5}$, PM_{10}, and NO_x (4 hours).
De Medrano et al. (2021)	Ensemble for NO_2, O_3, $PM_{2.5}$, and PM_{10} (48 hours).
Mao et al. (2021)	Extended LSTMs for $PM_{2.5}$ in multiple cities (72 hours).
Meteosim (2021)	Air quality with fine spatial resolution (72 hours).
Drewil and Al-Bahadili (2022)	LSTMs for $PM_{2.5}$, PM_{10}, NO_x, and CO (24 hours).
Liu et al. (2022)	AQI prediction with spatial inference and exogenous data.
Devasekhar and Natarajan (2023)	Combination of supervised ML and multi-agent systems.
Ouyang et al. (2023)	Local and global spatio-temporal GNNs to forecast $PM_{2.5}$.

CHAPTER 3

Background, problem, and Non-Functional Features

This chapter establishes the foundations and terminology necessary to formally define the forecasting problem under study. First, Section 3.1 defines variables, sensors, and key terms related to spatio-temporal aspects and time series forecasting. Next, Section 3.2 formulates the core problem: spatio-temporal forecasting of multivariable sensor observations. Section 3.3 introduces the NFFs, i.e., the beneficial properties we aim to achieve in the models addressing this research problem. Finally, Section 3.4 summarizes the chapter.

3.1 Background and formalization

The problem studied in this work has a generic fixed sensor at its core, which collects physical values continuously. The objective is to forecast its future values for different horizons, locations, and variables. Therefore, we denote the matter as: **spatio-temporal sensor data modeling**. This section builds the formalization required to pose the problem subsequently. Initially, we introduce terminology about variables and sensors in Section 3.1.1. Afterward, we describe the temporal and spatial aspects of the problem in Section 3.1.2 and Section 3.1.3, respectively. Subsequently, we present some terminology related to time series forecasting in Section 3.1.4. Finally, we state the problem in Section 3.2 and introduce the NFFs in Section 3.3.

3.1.1 Variables and sensors

First, consider a set of physical *variables* of interest $\mathcal{V}$. Each variable $v \in \mathcal{V}$ can take discrete or continuous values. *Discrete variables* can only take a countable set of values $x \in \mathbb{N}$ (i.e., they are separated). Differently, for *continuous variables*, the values $x \in \mathbb{R}$ can be arbitrarily close to each other. The number of trees per

square meter and temperature are examples of discrete and continuous variables, respectively.

The variables $v \in \mathcal{V}$ can be observed and recorded over time using a set of sensors $\mathcal{S}$. Throughout this work, the term *sensor* refers to a device that comprises one or more measurement instruments for different variables. Therefore, when referring to a sensor $s \in \mathcal{S}$, it may record one or more variables $v \in \mathcal{V}$. We denote the *number of variables and sensors* as n_v and n_s respectively, i.e., $n_v \equiv |\,\mathcal{V}\,|$ and $n_s \equiv |\,\mathcal{S}\,|$.

3.1.2 Temporal aspects

All observations recorded by sensors $s \in \mathcal{S}$ must have an associated *timestamp* t, which corresponds to a time interval (e.g., t =[12:00, 13:00)). The left endpoint of the time interval is used as a tag to refer to the timestamp (12:00 in the previous example). The *temporal coverage* of v refers to the set of timestamps in which sensors observe v, and we denote it as $\mathcal{T}_v$. The set $\mathcal{T}_v$ may include periods of inactivity or malfunction, i.e., some $t \in \mathcal{T}_v$ where sensor readings are empty. The *temporal granularity or resolution* of v refers to the time elapsed between consecutive observed data points of v. We denote it as $\tau_{s,v}$, and it is fixed for a given sensor s that records v. Note that the same timestamp may refer to different time intervals based on $\tau_{s,v}$. For instance, t =12:00 refers to the interval [12:00, 13:00) when $\tau_{s,v}$ =1 h, but to [12:00, 12:15) when $\tau_{s,v}$=15 min. In practice, we homogenize the temporal coverage $\mathcal{T}$ and the temporal granularity τ among variables and sensors for a given experiment. Therefore, we have that $\mathcal{T} = \{\, t_i \mid 1 \leq i \leq n_t \,\}$ with $t_{i+1} - t_i = \tau$, for all $1 \leq i < n_t$ and $n_t \equiv |\,\mathcal{T}\,| > 0$. We discuss how to accomplish this conversion in Section 4.1.2.

3.1.3 Spatial aspects

As discussed in Section 3.1.1, each $v \in \mathcal{V}$ is observed by several sensors $s \in \mathcal{S}_v \subset \mathcal{S}$. Sensors are placed at different *geolocations* $\mathcal{G}_v$, which correspond to either latitude-longitude pairs or disjoint bounded regions. Therefore, a one-to-one correspondence exists between sensors in $\mathcal{S}_v$ and the geolocations in $\mathcal{G}_v$. The *spatial coverage* $\mathcal{A}$ refers to a rectangular area that encompasses all geolocations $\mathcal{G} \equiv \{\, g \mid g \in \mathcal{G}_v,\, v \in \mathcal{V} \,\}$ within the ROI of the experiment.

With this setting, we can define the set of data points recorded by the instruments as follows:

$$X_v \equiv \left\{\, x^t_{s,v} \mid t \in \mathcal{T},\, s \in \mathcal{S}_v \,\right\}, \text{ where } v \in \mathcal{V}.$$

Consequently, the complete set of observations is:

$$X \equiv \bigcup_{v \in \mathcal{V}} X_v\,, \text{ where } |\,X\,| = n_t \cdot \sum_{v \in \mathcal{V}} |\,\mathcal{S}_v\,| \leq n_t \cdot n_s \cdot n_v.$$

3.1.4 Time series forecasting

Time series are sequences of observations recorded over time. Time series forecasting tackles the problem of predicting future values of time series. In this section, we introduce notation regarding time series forecasting. We denote by n_x the *number of input timestamps*, i.e., the width of the temporal input window. Similarly, we denote by n_y the *number of horizons*. The *shift (or leap)* l refers to the number of timestamps between the last observation of the input window and the first one of the output window. The parameters n_x, n_y, and l must be greater than zero (see Figure 3.1). These three parameters will be displayed as a subscript of the model to ease the reading or if there is ambiguity:

$$\text{MODEL}_{n_x,l,n_y}\,.$$

The term *horizon* h refers to the time duration leading up to the future predicted timestamps, expressed as:

$$h_i = (i+l)\cdot\tau\,,\ \text{where } i = 0,\ldots,n_y-1.$$

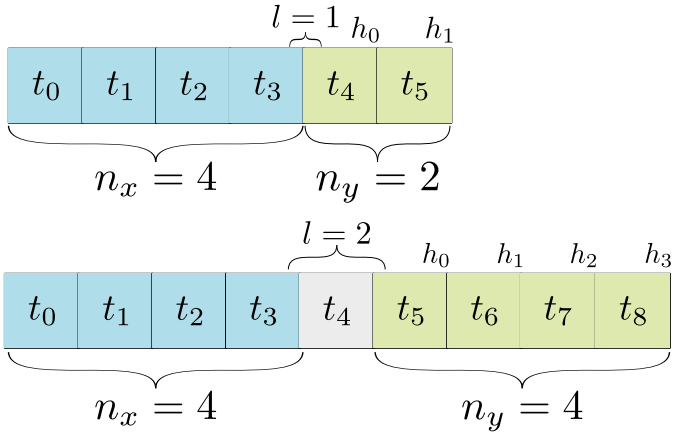


Figure 3.1: Time series forecasting notation.

3.2 Problem formulation

As is usually the case when developing forecasting models, $\mathcal{T}$ is split into three disjoint subsets:

$$\mathcal{T} = \mathcal{T}_{\text{train}} \sqcup \mathcal{T}_{\text{val}} \sqcup \mathcal{T}_{\text{test}}\,,$$

i.e., timestamps of the training, validation, and testing sets. We denote the observations corresponding to each of these datasets as X_{train}, X_{val}, $X_{\text{test}} \subset X$. Forecasting models are trained by inspecting only observations within X_{train}, and their worth is evaluated in X_{test}. During the training step, results are monitored in the set X_{val} to prevent *overfitting* (i.e., the model cannot generalize because it fits the training data too closely).

Considering the terminology introduced in Section 3.1, we pose the problem under study as follows:

Problem formulation:

Design a function $f : \mathbb{R}^{n_x \cdot n_s \cdot n_v} \longrightarrow \mathbb{R}^{n_y \cdot n_s \cdot n_v}$ that forecasts observations of $\mathcal{S}$ on a time window that starts on $t_{\gamma+l}$ of size n_y, given observations from $\mathcal{T}_{\text{train}}$ and a time window that ends on t_γ of size n_x, for every $t_\gamma \in \mathcal{T}_{\text{test}}$, $v \in \mathcal{V}$ and $s \in \mathcal{S}_v$.

In other words:

$$
\begin{aligned}
&\text{Design } f : \mathbb{R}^{n_x \cdot n_s \cdot n_v} \longrightarrow \mathbb{R}^{n_y \cdot n_s \cdot n_v} \text{ to forecast} \\
&\qquad \mathcal{Y}^{t_\gamma} \equiv \left\{ y^t_{s,v} \mid t = t_{\gamma+l}, \ldots, t_{\gamma+l+n_y-1},\ v \in \mathcal{V},\ s \in \mathcal{S}_v \right\} \quad \forall\, t_\gamma \in \mathcal{T}_{\text{test}} \\
&\text{given } \mathcal{X}^{t_\gamma} \equiv \left\{ x^t_{s,v} \mid t = t_{\gamma-(n_x-1)}, \ldots, t_\gamma,\ v \in \mathcal{V},\ s \in \mathcal{S}_v \right\}, \\
&\text{and } X_{\text{train}} \equiv \left\{ x^t_{s,v} \mid t \in \mathcal{T}_{\text{train}},\ v \in \mathcal{V},\ s \in \mathcal{S}_v \right\}, \\
&\text{where } \mathcal{Y}^{t_\gamma},\ \mathcal{X}^{t_\gamma},\ X_{\text{train}} \subset X.
\end{aligned}
$$

With this definition, we can define forecast horizons in terms of timestamps:

$$h_i = t_{\gamma+l+i} - t_\gamma \,, \text{ where } i = 0, \ldots, n_y - 1.$$

Therefore, we can also express the timestamps of the predictions as:

$$t_{\gamma+l+i} = t_\gamma + h_i \,, \text{ where } i = 0, \ldots, n_y - 1.$$

Figure 3.2 depicts the problem visually. Once the problem is laid, we outline the desirable properties required for deploying the model in real-world scenarios in the next section.

3.3 Non-Functional Features

Regression problems arise when looking for relationships between a dependent variable and one or more independent variables. The overall performance of models that tackle such problems is usually measured in terms of how close the predictions are to the recorded true values of the dependent variable. This measure of closeness is realized by calculating an error metric, such as the RMSE or the Mean Absolute Error (MAE). In general terms, we denote this characteristic of the model as the *error metric minimization*, or *error metric* in short. In this way, we can grade models simply by calculating and comparing an error metric over the same dataset.

Error metrics offer an objective manner to evaluate certain aspects of the performance of regression models. Admittedly, they are the main way in which models

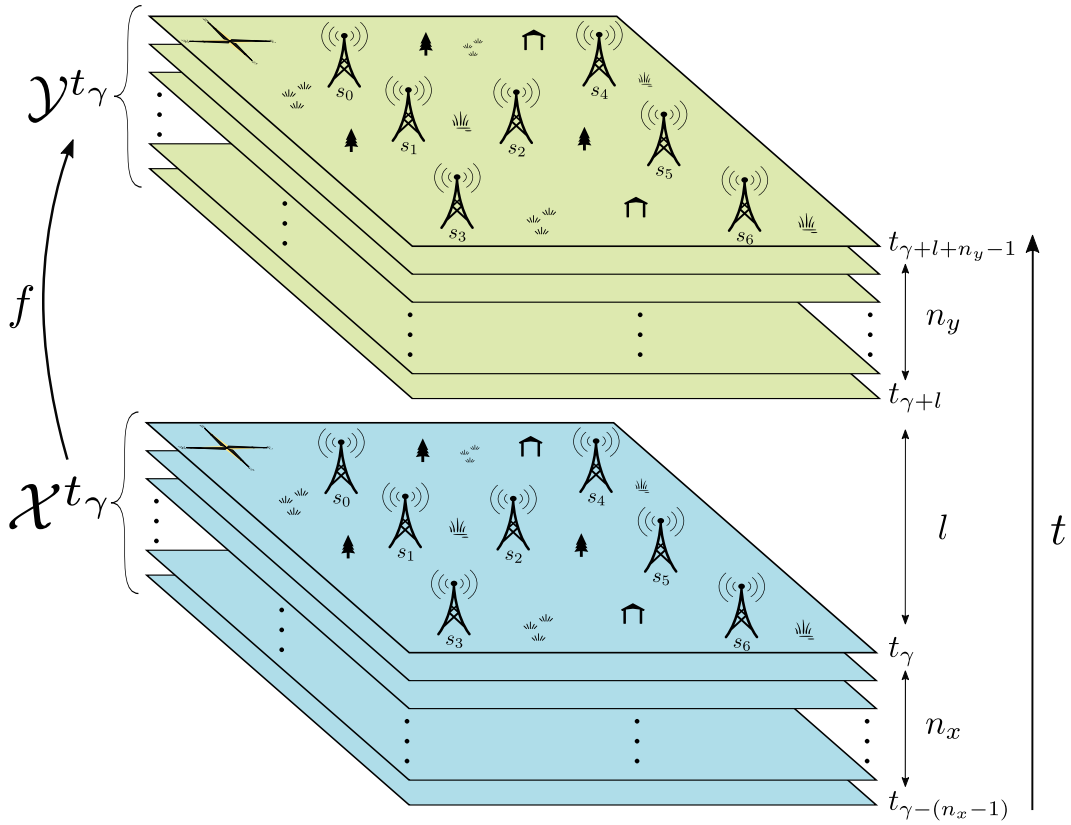


Figure 3.2: The problem: Spatio-temporal sensor data modeling for $v \in \mathcal{V}$.

are evaluated in the state-of-the-art. Nevertheless, they neglect some important aspects of the model under certain scenarios. We denote such characteristics as *Non-Functional Features (NFFs)*. These features can derive from the model itself or from a forecasting framework in which the model is encased. In contrast to functional features, such as error metrics, NFFs address whether the model is resilient enough to realize its purpose.

In real-world environments, models are likely to face adverse situations and should be able to adapt and recover, to some extent. Large forecasting models that achieve low error metrics are valuable, though this strength does not necessarily reflect their adaptability to real-world scenarios. In fact, it can work against them if, for example, they are too big to scale. Similarly, the computational time required by complex models can be a disadvantage, particularly when estimating missing observations. Therefore, it is important to seek a balance in the trade-off between quality and performance. Ideally, models should achieve relevant NFFs while also maintaining low error metrics compared to the literature.

The following subsections describe some NFFs that are relevant to the studied problem. We initially discussed these characteristics in a previous work (Prado-Rujas, Serrano, et al., 2021b). Chapters 5, 6, and 7 practically explores these

characteristics, and Section 8.2 discusses some additional ones.

3.3.1 Flexibility

Consider the problem described in Section 3.2. While the framework is operational, some of the sensors may become obsolete and be removed from the network. Similarly, some additional ones may be installed as long as they are located within the spatial coverage $\mathcal{A}$. Under this scenario, *flexibility* is defined as the ability of the forecasting framework to accept a different number of inputs and outputs without changing its inner structure. This implies that the set $\mathcal{S}$ (and therefore $\mathcal{G}$) can change dynamically as time progresses. Figure 3.3 depicts the idea behind this NFF.

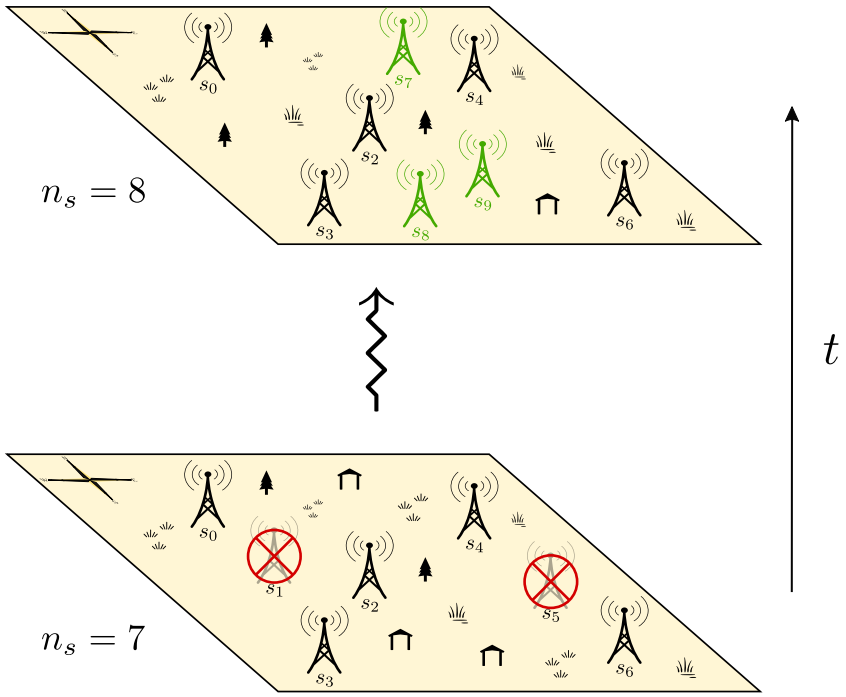


Figure 3.3: Graphical description of the flexibility NFF.

3.3.2 Robustness

Once the sensors are operational, they will start streaming a continuous flow of measurements into a central server. Unavoidably, some of those measurements will be invalid or empty due to hardware breakdowns or other external factors. In this vein, *robustness* is defined as the ability to recover from sensor failure and continue producing reliable predictions. This implies that even if some $x_{s,v}^t \in X$ are unknown for some $t \in \mathcal{T}$, $v \in \mathcal{V}$, $s \in \mathcal{S}_v$, the framework should still be able to operate. Figure 3.4 shows an example of robustness, filling missing observations on $\mathcal{X}^{t_\gamma}$. The model f uses neighboring observations to estimate s_3 and previous observations for s_6 in the input, providing forecasts for all sensors on $\mathcal{Y}^{t_\gamma}$.

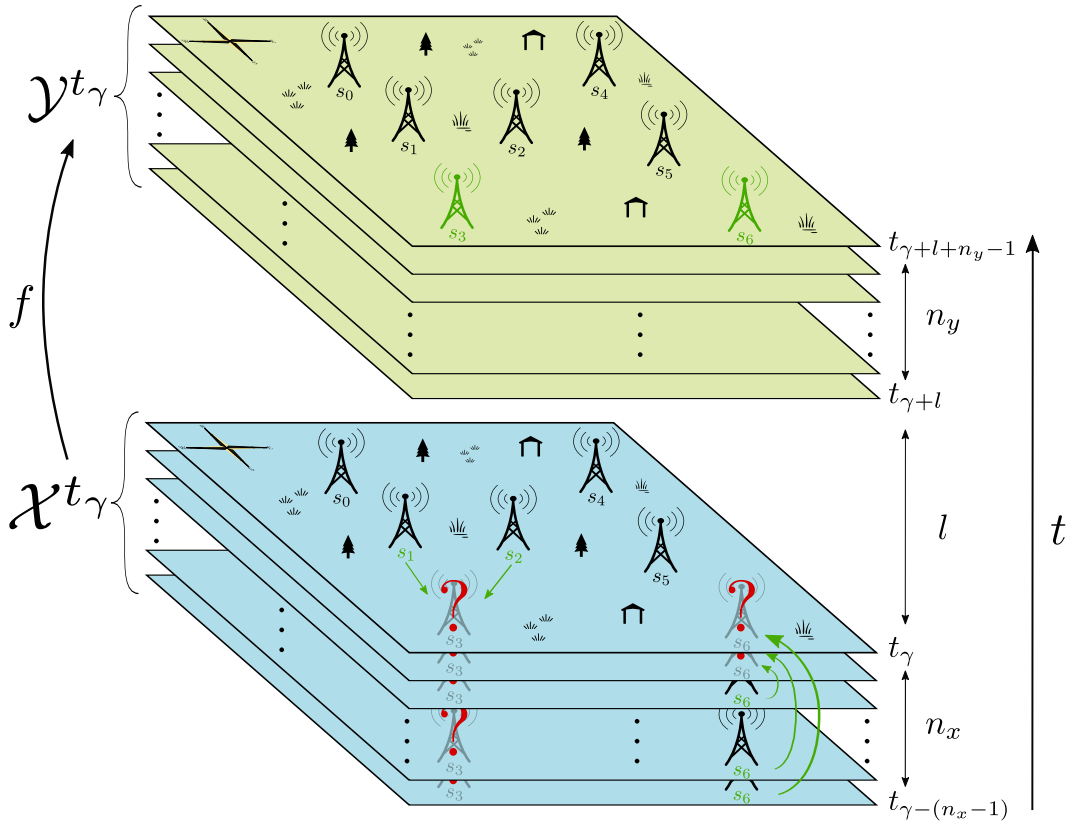


Figure 3.4: Graphical description of the robustness NFF.

3.3.3 Extensibility

Generally, this feature measures the capability of a model to be effortlessly extended with new functionalities. For the studied problem, the *extensibility* of the framework refers to the feasibility of incorporating new variables into $\mathcal{V}$. This extension should be applicable both at the input and output levels of the model. Figure 3.5 shows an example of this NFF, where the model f is extended with two new variables, i.e., $\mathcal{V} = \{v_0, v_1, v_2\}$.

3.3.4 Integrability

This NFF refers to the capability of the framework to incorporate new subsystems. In the studied context, *integrability* refers to the feasibility of integrating additional data. Specifically, such data must:

- be of a different nature than that of $\mathcal{V}$,
- be obtained without the intervention of the sensors $\mathcal{S}$,
- serve just as input to the framework, and

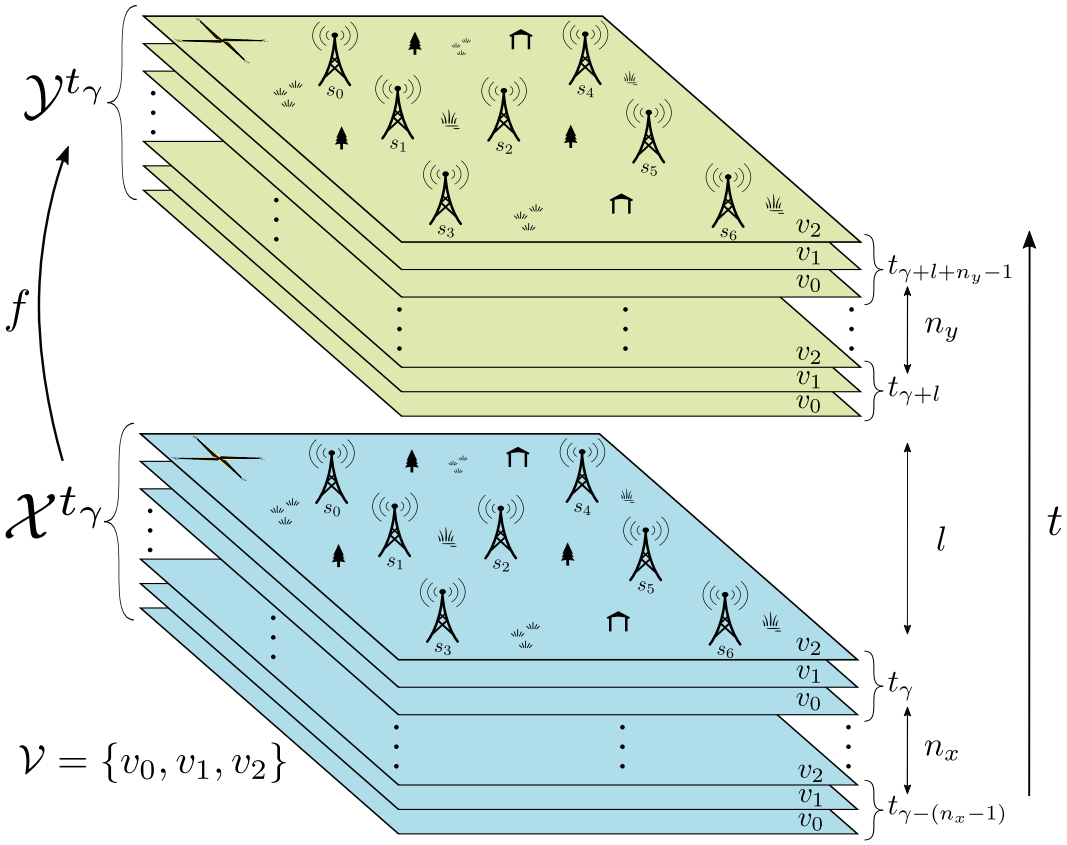


Figure 3.5: Graphical description of the extensibility NFF.

- contribute to improving the forecasts.

In other words, the input of the forecasting framework may be enriched with $p \geq 0$ exogenous datasets $\{X_1, \ldots, X_p\}$ that do not overlap with the dataset being forecasted, i.e.:

$$X_i \cap \mathcal{Y}^{t_\gamma} = \varnothing, \ \forall\, t_\gamma \in \mathcal{T}_{\text{test}}, \ X_i \in \{X_1, \ldots, X_p\}\,.$$

Figure 3.6 shows an example where the model f integrates exogenous datasets to its input.

3.4 Summary

This chapter lays the groundwork for presenting the problem of spatio-temporal forecasting of multivariable sensor observations. First, we define essential concepts, such as variables, sensors, and their spatio-temporal attributes. Section 3.1 additionally distinguishes between discrete and continuous variables, temporal aspects like timestamps and granularity, and spatial elements like geolocations and the mesh-grid structure. We also elaborate on time series forecasting terminology and its relevance to sensor data modeling, facilitating the subsequent problem

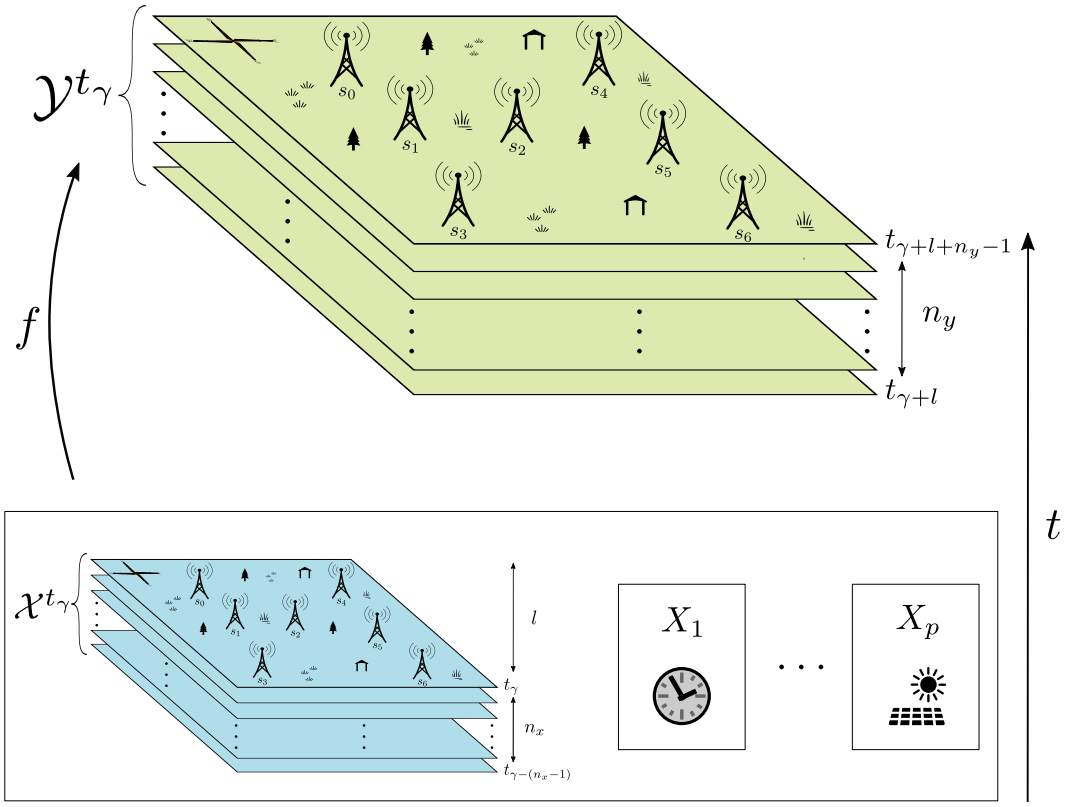


Figure 3.6: Graphical description of the integrability NFF.

formulation.

Section 3.2 defines the problem formally: designing a predictive function that forecasts sensor observations on several horizons based on historical data. The temporal dataset is split into training, validation, and testing subsets. This procedure allows to train models, evaluate their performance, and avoid overfitting. The mathematical formulation and visual representation illustrate the relationship between input data, forecast horizons, and the resulting predictions.

Lastly, Section 3.3 introduces NFFs, which go beyond conventional error metrics to address real-world model applicability. NFFs are essential to this thesis, and we discuss four of the most meaningful ones in this chapter:

- **flexibility**, which refers to adapting to changes in sensor configurations,
- **robustness**, which ensures reliable forecasts despite missing or invalid data,
- **extensibility**, which facilitates incorporating new variables, and
- **integrability**, which enables the use of exogenous datasets to enhance predictions.

These characteristics highlight the importance of developing models that are not only accurate, but also versatile and adaptable to dynamic environments. However, it is important to consider the trade-off between the quality of NFFs and their performance. For example, incorporating more exogenous datasets into the model inputs generally improves predictive capabilities, but also slows down computation. As we will see throughout the experiments, the real challenge lies in finding the right balance between NFFs and traditional error-based metrics.

CHAPTER 4

Methodology

This chapter presents the developed methodology for spatio-temporal sensor data modeling from a data-agnostic point of view. The core idea revolves around mesh-grids and how they foster the pursued NFFs. Section 4.1 presents the data analysis and transformation processes. After that, we describe the general model and how we evaluate it in Section 4.2 and Section 4.3, respectively. Finally, Section 4.4 summarizes the chapter.

4.1 Data analysis and transformation

This section presents the different phases involved in understanding and transforming the data. These phases serve as an initial step before developing forecasting models. However, they may be repeated if new data becomes available or if some characteristics of the problem vary. As a result, these processes are not independent but rather entwine and provide feedback to each other, as Figure 4.1 conveys. The main objective of these phases is to convert raw sensor data into higher-dimensional arrays that retain only the relevant information and metadata. The final dataset obtained will comprise stacked mesh-grids, as we shall see.

4.1.1 Phase 1: Exploratory Data Analysis

Raw sensor data are typically stored in two-dimensional archives, such as Comma-Separated Values (CSV) files. Rows correspond to individual observations, while columns represent attributes of those observations. As described in Section 3.1.2, all sensor observations must have an associated timestamp $t \in \mathcal{T}$. Additionally, as discussed in Section 3.1.3, each observation is retrieved from a sensor $s \in \mathcal{S}_v$ located at $g \in \mathcal{G}_v$ for the variable $v \in \mathcal{V}$. Along with the actual measured value, those are the minimum required attributes of every observation: t, s, and v. They may come as columns of the tabular raw data or be inferred from a predefined convention (e.g., the file name). Furthermore, the units of each variable

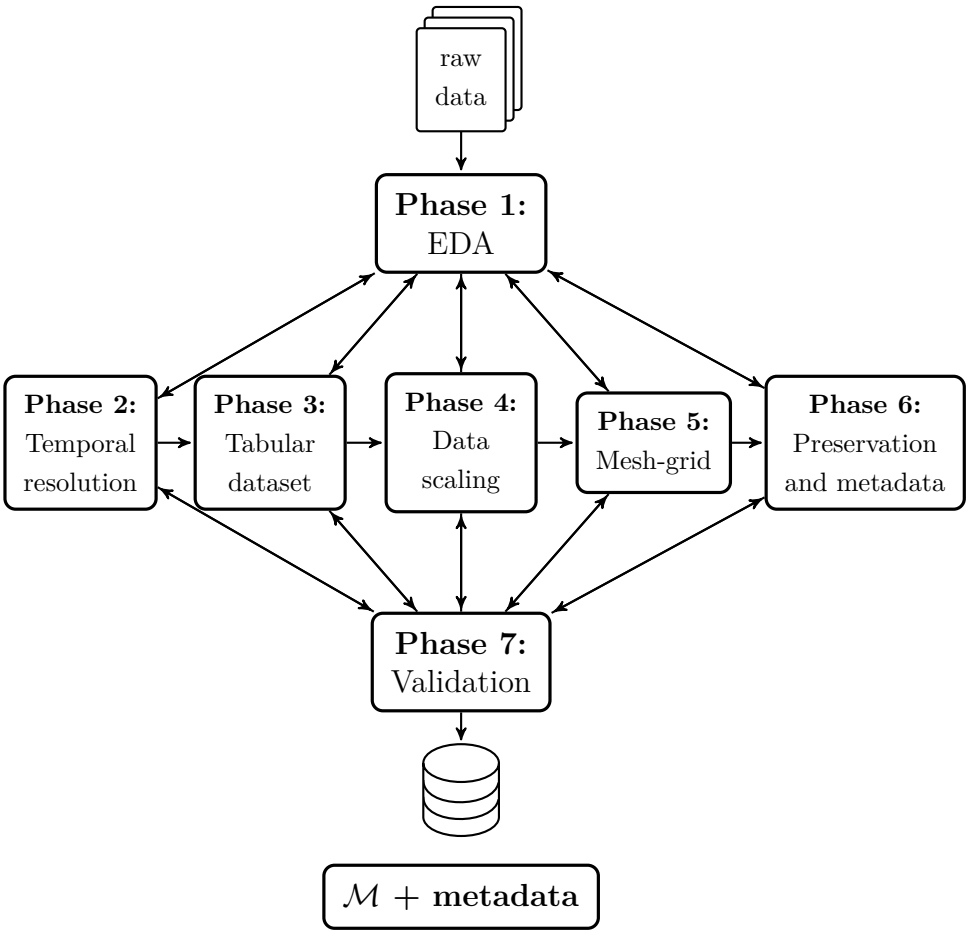


Figure 4.1: Data analysis and transformation phases.

and sensor are also required.

Depending on the use case, raw sensor data may be stored in a single or multiple files. For instance, they might be split by: (1) periods (months, years, seasons, etc.); (2) sensors; (3) variables; or (4) a combination of the above. Because of this variability, fully automating this phase is usually challenging and often requires some degree of craftsmanship. After this initial phase, we should be able to answer the following questions:

1. **Sensors**: Which sensors are we considering? I.e., what is the set $\mathcal{S}$?
2. **Variables**: Which variables are we considering? I.e., what is the set $\mathcal{V}$?
3. **Variable units**: What are the units of every variable and sensor?
4. **Variables observed by every sensor**: Which variables does each sensor record? I.e., what is the set $\mathcal{S}_v$ for every $v \in \mathcal{V}$?
5. **Geolocation of every sensor**: What is the geolocation associated to each

sensor? I.e., what is the set $\mathcal{G}_v$ corresponding to each $\mathcal{S}_v$ for every $v \in \mathcal{V}$?

6. **Spatial coverage**: What is the region that the sensors cover and its area?
7. **Temporal granularity**: What is the temporal granularity of each variable and sensor? I.e, what is $\tau_{s,v}$ for every $s \in \mathcal{S}_v$ and $v \in \mathcal{V}$?
8. **Temporal coverage**: Which timestamps are we considering? I.e., what is the set $\mathcal{T}$?
9. **Missing observations**: Which observations were not recorded from the temporal coverage?
10. **Main statistics**: What are the main statistics of every variable and sensor?

Table 4.1 summarizes how to answer these questions during the Exploratory Data Analysis (EDA) phase. The purpose of this phase is to gain an initial understanding of the available data. Based on its results, we may decide to restrict the problem to a subset of $\mathcal{S}$, $\mathcal{V}$, or $\mathcal{T}$. This could be necessary if some sensors or variables are not well covered either spatially or temporally. Furthermore, we may be able to undertake some data imputation based on our findings already in this phase. Section 4.1.5 additionally discusses methods for estimating sensor values for the entire spatial coverage.

4.1.2 Phase 2: Temporal resolution

In question 5 of Section 4.1.1, we identified the temporal resolution $\tau_{s,v}$ of every variable $v \in \mathcal{V}$ and sensor $s \in \mathcal{S}_v$. In this phase, we define a unique and shared temporal resolution τ. Afterward, we shall convert all observations and their corresponding timestamps into τ. We assume that all temporal datasets correspond to a common time zone and format. For example, if X employs Central European Time (CET) but an exogenous dataset uses Central Standard Time (CST), we must first homogenize them.

The defined temporal resolution τ may be finer or coarser than each $\tau_{s,v}$. For the sake of simplicity, we assume that the coarser one is a multiple of the other for some $m \in \mathbb{N}$, e.g.:

- If $\tau_{s,v} = 1$ s and $\tau = 1$ min then $\tau_{s,v} < \tau$ and $\tau = m \cdot \tau_{s,v}$ with $m = 60$.
- If $\tau_{s,v} = 1$ h and $\tau = 15$ min then $\tau_{s,v} > \tau$ and $\tau_{s,v} = m \cdot \tau$ with $m = 4$.

Converting X into a finer temporal resolution without additional sensor observations is inherently artificial. However, it can be beneficial for certain experiments, provided we do it realistically. For example, if $\tau_{s,v}$ is 15 minutes but the granularity of an exogenous dataset X_1 is hourly, we can interpolate the three missing timestamps by forward-filling.

Table 4.1: Steps of the EDA phase of the methodology. The table summarizes the objectives of this phase and the corresponding tasks required to achieve them.

What?	Why?	How?
1. $\mathcal{S}$	To define the set of sensors considered.	Inspect raw data metadata to determine sensor characteristics and identifiers. This step results in the definition of $\mathcal{S}$.
2. $\mathcal{V}$	To define the set of variables considered.	Inspect raw data metadata to derive available variables. Variables may be transformed if necessary. This step results in the definition of $\mathcal{V}$.
3. Units	To homogenize units across sensors.	Inspect raw data metadata for all sensors. Visualization and literature review can help infer missing unit information. If sensors collect variable observations in different units, homogenize them for consistency. This step is fundamental for result interpretation and is used in Sections 4.1.3 and 4.1.6.
4. $\mathcal{S}_v$	To define the set of variables observed by each sensor.	Inspect raw data to infer sensor coverage of the available variables. This step results in the definition of $\mathcal{S}_v$ for every $v \in \mathcal{V}$.
5. $\mathcal{G}_v$	To define the set of geolocations of each sensor.	Plot a scatter graph of $\mathcal{G}_v$ for every $v \in \mathcal{V}$, checking for over- or under-representation of variables. Compute the distance between all pairs of sensors and their convex hull to aid the meshgrid definition in Section 4.1.5.
6. $\mathcal{A}$	To define the spatial coverage based on $\mathcal{G}_v$.	Derive the total area covered by the sensors. This step supports the meshgrid definition in Section 4.1.5.
7. $\tau_{s,v}$	To homogenize temporal granularity.	Inspect raw data to determine $\tau_{s,v}$ for every $s \in \mathcal{S}_v$ and $v \in \mathcal{V}$. This step supports the definition of τ in Section 4.1.2.
8. $\mathcal{T}$	To define the temporal coverage.	Inspect raw data to detect defective sensors and other artifacts, such as daylight saving time shifts. This step results in the definition of $\mathcal{T}$.
9. Missing observations	To detect defective sensors or missing periods.	Plot time versus the number of available observations for every $v \in \mathcal{V}$. Sometimes, raw data includes quality codes representing the observation integrity.
10. Statistics	To understand differences between sensors and variables and for data scaling.	Calculate several statistics and boxplots for every $s \in \mathcal{S}_v$ and $v \in \mathcal{V}$. Check for outliers (e.g., extreme or unrealistic values), and flag, correct, or remove them. This step also supports data scaling in Section 4.1.4.

To accomplish the conversion, we must consider which temporal resolution is coarser and whether the variable is continuous or discrete. Based on these cases, Table 4.2 summarizes the steps to carry out.

Table 4.2: Data conversion from $\tau_{s,v}$ into a common temporal resolution τ.

Finer vs. coarser	**Discrete vs. continuous**	**Conversion (for each $v \in \mathcal{V}$ and $s \in \mathcal{S}_v$)**
$\tau_{s,v} < \tau$ ($\tau = m \cdot \tau_{s,v}$)	Discrete	Sum or concatenate every m consecutive observations (rows).
	Continuous	Compute the mean every m consecutive observations (rows).
$\tau_{s,v} > \tau$ ($\tau_{s,v} = m \cdot \tau$)	Discrete	Randomly split each discrete observation onto m consecutive bins.
	Continuous	Estimate $m-1$ intermediate observations using interpolation (e.g., linear, nearest-neighbor, etc.).

4.1.3 Phase 3: Tabular dataset

At this stage, we have analyzed the data, defined the relevant set of sensors and variables, and decided on a common temporal resolution. In this phase, we combine all sensor observations into a tabular structure. The outcome is a two-dimensional array where each row is an observation and columns comprise:

- a timestamp $t \in \mathcal{T}$;
- a sensor $s \in \mathcal{S}_v$, which corresponds to $g \in \mathcal{G}_v$;
- a variable $v \in \mathcal{V}$; and
- an observed value, which could be empty as discussed in Section 4.1.1.

The result shall resemble Table 4.3. We use this structure in Section 4.1.5 to build the mesh-grid dataset.

The steps to undertake in this phase are as follows:

1. Convert all observations to the units decided in Section 4.1.1.
2. Discard unnecessary observations from X, i.e., the ones where $t \notin \mathcal{T}$, or $s \notin \mathcal{S}_v$, or $v \notin \mathcal{V}$.
3. Discard unnecessary attributes, keeping t, s, v, and the observed value.
4. Identify and unify empty observations that have different formats or codes,

Table 4.3: Tabular dataset obtained in the third phase of the methodology.

index	timestamp	sensor	variable	value			
1	t_1	s_1	v_1	$x^{t_1}_{s_1,v_1}$	s_1, v_1 $\forall t \in \mathcal{T}$	v_1 $\forall t \in \mathcal{T}$ $\forall s \in \mathcal{S}$	$\forall t \in \mathcal{T}$ $\forall s \in \mathcal{S}$ $\forall v \in \mathcal{V}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
n_t	t_{n_t}	s_1	v_1	$x^{t_{n_t}}_{s_1,v_1}$			
$n_t + 1$	t_1	s_2	v_1	$x^{t_1}_{s_2,v_1}$			
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
$n_t \cdot n_s$	t_{n_t}	s_{n_s}	v_1	$x^{t_{n_t}}_{s_{n_s},v_1}$			
$n_t \cdot n_s + 1$	t_1	s_1	v_2	$x^{t_1}_{s_1,v_2}$			
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
$n_t \cdot n_s \cdot n_v$	t_{n_t}	s_{n_s}	v_{n_v}	$x^{t_{n_t}}_{s_{n_s},v_{n_v}}$			

The first stride traverses all timestamps for $s_1 \in \mathcal{S}$ and $v_1 \in \mathcal{V}$, the second one traverses all timestamps and sensors for $v_1 \in \mathcal{V}$, and the third one traverses all variables, sensors, and timestamps. Therefore, there are $n_t \cdot n_s \cdot n_v$ rows. Depending on the sets $\mathcal{V}$ and $\mathcal{S}$, the number of rows may be smaller in practice.

such as Not a Number (NaN) or -9999.

5. Identify and discard outliers with the aid of the statistics computed on Section 4.1.1.

4.1.4 Phase 4: Data scaling

The literature widely discusses the benefits of scaling numerical data when developing models (e.g., Aggarwal, 2018). NNs are no different; therefore, all input data tends to be scaled. The most common scaling techniques are *normalization* and *standardization*. Their equations applied to a set of observations A are presented on Eq. 4.1 and Eq. 4.2, respectively:

$$\widehat{x} = \frac{x - \min(A)}{\max(A) - \min(A)} \in [0, 1], \quad \forall x \in A, \tag{4.1}$$

$$\widehat{x} = \frac{x - \mu(A)}{\sigma(A)}, \quad \forall x \in A, \tag{4.2}$$

where $\min(A)$ and $\max(A)$ refer to the minimum and maximum observations of the set A, respectively; $\mu(A)$ refers to the mean value of A; and $\sigma(A)$ refers to the standard deviation of A.

The normalization equation presented on Eq. 4.1 is know as *min-max normalization*. The formula can be generalized to any interval $[a, b] \subset \mathbb{R}$ as follows:

$$\widehat{x} = a + (b - a) \cdot \frac{x - \min(A)}{\max(A) - \min(A)} \in [a, b], \quad \forall x \in A. \tag{4.3}$$

Both in Eq. 4.1 and Eq. 4.3, the idea is to linearly interpolate the data into a closed interval. In contrast, the idea behind standardization (Eq. 4.2) is to transform the data so that it is drawn from a normal distribution with null mean and standard deviation of 1. The main difference is that, while standardization homogenizes the data, normalization squeezes them into a closed interval. Figure 4.2 shows an example of this.

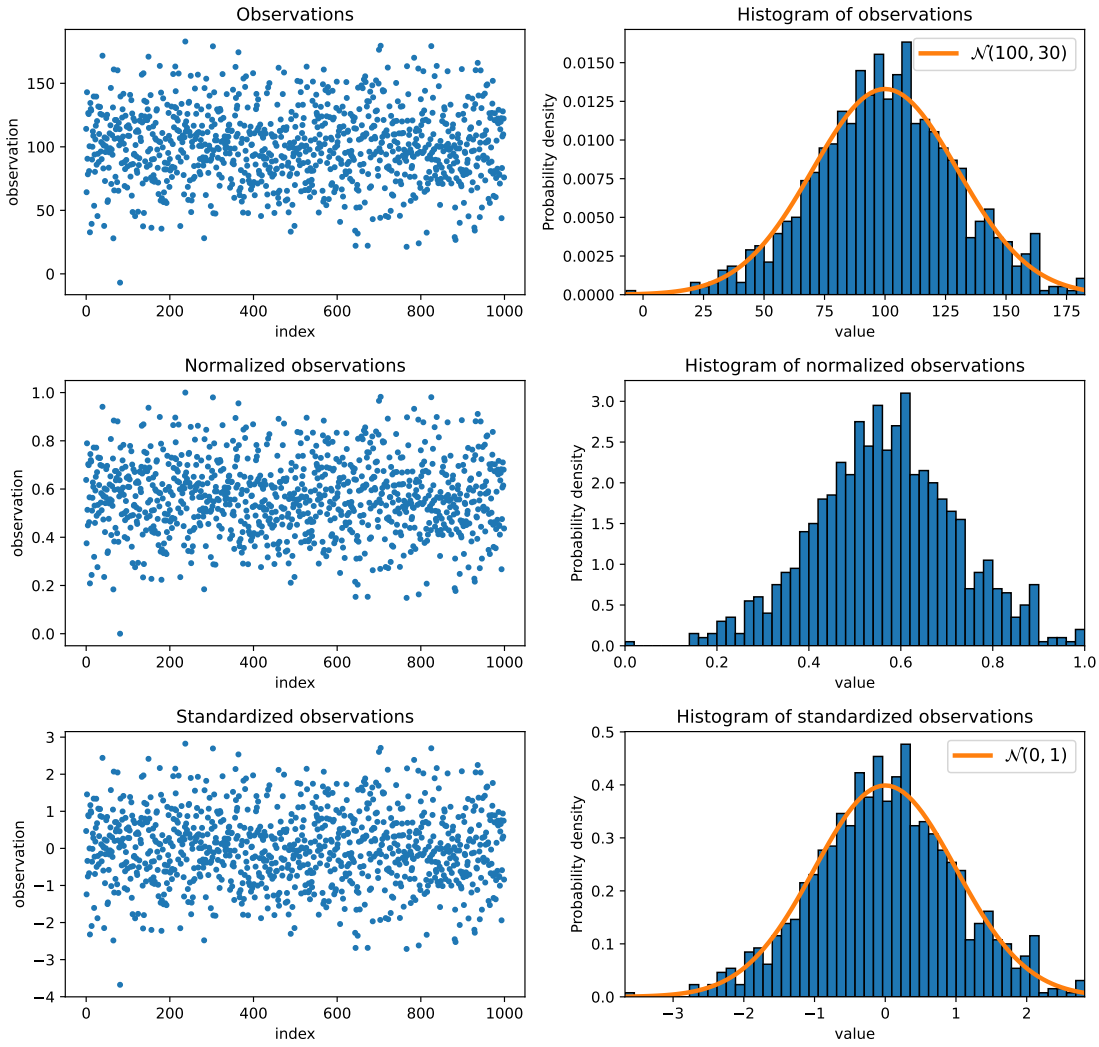


Figure 4.2: Data scaling example. One thousand observations are randomly drawn from a normal distribution $\mathcal{N}(100, 30)$ (top), normalized as per Eq. 4.1 (middle), and standardized as per Eq. 4.2 (bottom). If we ignore the axes values, data have the same shape in the three cases.

In Section 4.1.1, we calculated several statistics of every sensor-variable combination (i.e., of X_v as defined in Section 3.1.3). Therefore, equations 4.1 and 4.2 must be applied to every set $X_v \subset X$. Depending on the use case, it might be better to use one scaling technique or the other. For instance, when some observations

differ greatly from their mean value, their standardized peer belongs to the tails of the normal distribution. If those observations are not identified as outliers, this phenomenon could distort the forecasting model. In any case, the choice of scaling technique shall be included in the hyperparameter tuning step.

The set X only includes numerical data. However, the exogenous datasets $\{X_1, \ldots, X_p\}$ may consider categorical values. In that case, we may use the *one-hot encoding* scaling technique. To this aim, we first extend the dataset with as many null columns as categories there are. Then, we select only the observations that belong to a specific category and set their value to 1. Alternatively, we may use *ordinal encoding* if the categorical variable can be ordered. In this case, we map categories to integers ranging from zero to the number of categories minus 1. Examples of such variables are continuous numerical values binned into intervals or grades scored by students on an A to F scale.

4.1.5 Phase 5: The mesh-grid

A *mesh-grid* (or *grid* in short) $\mathcal{M}_v^t$ is a two-dimensional array based on a regular partition of the spatial coverage $\mathcal{A}$. The elements of $\mathcal{M}_v^t$ are observations of the variable $v \in \mathcal{V}$ on the timestamp $t \in \mathcal{T}$. The *shape* of $\mathcal{M}_v^t$ is the pair formed by its number of rows n_r and columns n_c, which we sometimes denote by $n_r \times n_c$. The *spatial granularity or resolution* $\sigma \equiv (\sigma_\mathsf{y}, \sigma_\mathsf{x})$ of $\mathcal{M}_v^t$ is the pair obtained by dividing the height of $\mathcal{A}$ by n_r and its width by n_c. It can also be seen as the distance between the centers of consecutive elements of $\mathcal{M}_v^t$ along the latitude and longitude.

At this stage of the methodology, we have scaled and organized the set X in a tabular structure, as shown in Table 4.3. In this phase, we convert those two-dimensional data into a four-dimensional set of mesh-grids. We generally refer to these multi-dimensional arrays as *tensors*, which are a generalization of scalars, vectors, and matrices to higher dimensions. The *rank of a tensor* is the number of axes or dimensions it has. The *shape* refers to the number of elements of each of its axes, as in the case of grids. The transformation into mesh-grids happens for every $t \in \mathcal{T}$ and $v \in \mathcal{V}$. Consequently, we define the produced dataset as:

$$\mathcal{M} \equiv \bigcup_{v \in \mathcal{V}} \mathcal{M}_v\,, \text{ where } \mathcal{M}_v \equiv \left\{ \mathcal{M}_v^t \mid t \in \mathcal{T} \right\}.$$

Practically, the set $\mathcal{M}$ is a rank-4 tensor of shape (n_t, n_r, n_c, n_v). Figure 4.3 depicts the forecasting problem when considering mesh-grids instead of sensor locations (see Figure 3.2 for that). With the introduction of mesh-grids, the sets $\mathcal{Y}^{t_\gamma}$ and $\mathcal{X}^{t_\gamma}$ for $t_\gamma \in \mathcal{T}_{\text{test}}$ that we defined in Section 3.2 become:

$$\mathcal{Y}^{t_\gamma} \equiv \left\{ \mathcal{M}_v^t \mid t = t_{\gamma+l},\, \ldots,\, t_{\gamma+l+n_y-1},\, v \in \mathcal{V} \right\},$$
$$\mathcal{X}^{t_\gamma} \equiv \left\{ \mathcal{M}_v^t \mid t = t_{\gamma-(n_x-1)},\, \ldots,\, t_\gamma,\, v \in \mathcal{V} \right\}.$$

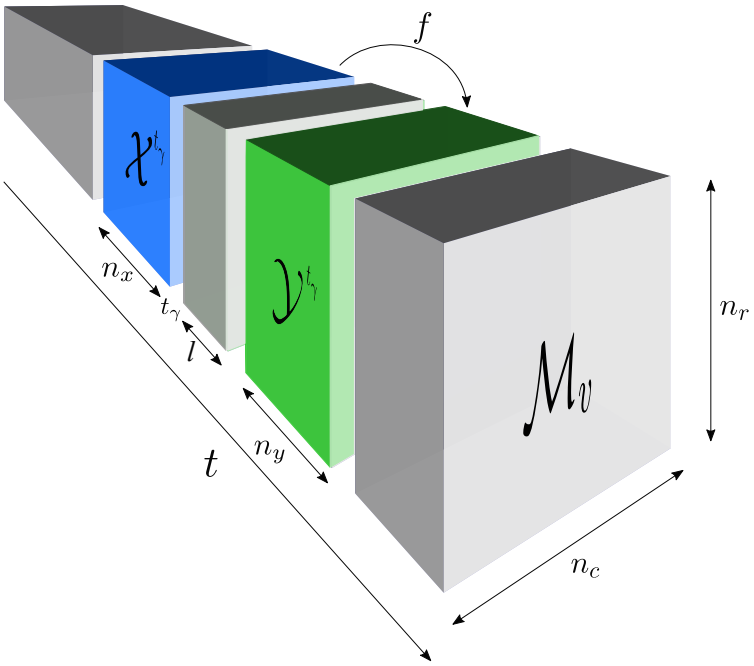


Figure 4.3: The problem: Spatio-temporal sensor data modeling using mesh-grids for a varible $v \in \mathcal{V}$.

Two aspects must be determined to carry out the transformation: the spatial resolution for the mesh-grid placement and the method to convert sensors observations into grids. In both cases, the number of sensors n_s and their geolocations $g \in \mathcal{G}$ play a major role. As discussed in Section 3.1.3, the set $\mathcal{G}$ corresponds to either latitude-longitude pairs or disjoint bounded regions. In the later case, we identify the regions by the coordinates of their centroids. The distances between geolocations condition spatial resolution: the closer they are, the finer resolution we can achieve. However, if they are too close, their observations become redundant. Therefore, geolocations must be dense enough so that we can build the mesh-grids realistically. In addition, the relative positions of the sensors restrict our alternatives when building the mesh-grids, as we shall soon see. The nature of the variable is another aspect that determines the conversion method, as we shall also explore later on.

Phase 5.1: Spatial resolution and grid placement

We must choose the parameters n_r and n_c as a compromise between the number of sensors n_s and the desired spatial resolution $\sigma \equiv (\sigma_\mathsf{y}, \sigma_\mathsf{x})$. On the one hand, it can be tempting to fix a very small σ to obtain an exhaustive mesh-grid. However, on the other hand, n_s is a limiting factor of σ for producing realistic mesh-grids. This is because in the conversion from $\left\{ x^t_{s,v} \mid s \in \mathcal{S} \right\}$ to $\mathcal{M}^t_v$ the number of data values goes from n_s to $n_r \cdot n_c$, which are somewhat synthetic. Therefore, we must consider the ratio $n_s/(n_r \cdot n_c)$ on each particular use case. Once the parameters n_r

and n_c are determined, the position of the mesh grid remains to be fixed. We can do this by first calculating the minimum and maximum latitudes and longitudes of $\mathcal{G}$:

$$\check{\mathsf{y}} \equiv \min\{\mathsf{y}_i\} - \varepsilon,\ \hat{\mathsf{y}} \equiv \max\{\mathsf{y}_i\} + \varepsilon,$$
$$\check{\mathsf{x}} \equiv \min\{\mathsf{x}_i\} - \varepsilon,\ \hat{\mathsf{x}} \equiv \max\{\mathsf{x}_i\} + \varepsilon,$$

where $(\mathsf{y}_i, \mathsf{x}_i) \in \mathcal{G}$[1] and $i \in \{0, \ldots, n_s - 1\}$. We introduce a small offset $\varepsilon > 0$ to these four parameters to prevent sensors from lying within the grid border. From there, we obtain the spatial resolution as follows:

$$\sigma_{\mathsf{y}} \equiv \frac{\hat{\mathsf{y}} - \check{\mathsf{y}}}{n_r}, \quad \sigma_{\mathsf{x}} \equiv \frac{\hat{\mathsf{x}} - \check{\mathsf{x}}}{n_c}.$$

Finally, we denote the centers of the elements of the mesh-grid as $\mathcal{G}'$, which we calculate as:

$$\mathcal{G}' \equiv \{\, (\mathsf{y}_i, \mathsf{x}_i) \mid \mathsf{y}_i \in \mathsf{Y}\,,\ \mathsf{x}_i \in \mathsf{X}\,\}\,, \text{where}$$
$$\mathsf{Y} \equiv \left\{ \check{\mathsf{y}} + \sigma_{\mathsf{y}} \cdot \left(\frac{1}{2} + i\right) \mid i = 0, \ldots, n_r - 1 \right\} \text{ and}$$
$$\mathsf{X} \equiv \left\{ \check{\mathsf{x}} + \sigma_{\mathsf{x}} \cdot \left(\frac{1}{2} + i\right) \mid i = 0, \ldots, n_c - 1 \right\}.$$

Figure 4.4 depicts the terminology of this conversion graphically.

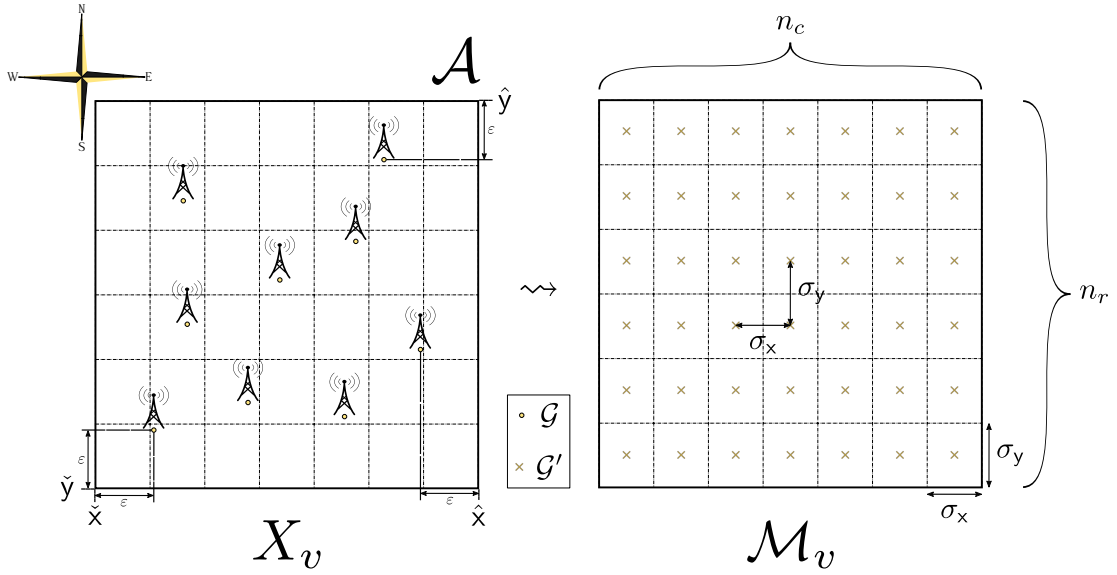


Figure 4.4: Conversion from sensor obervations into the mesh-grid.

[1]Note that we refer to coordinates as (y, x) and not as (x, y) to align with the usual (latitude, longitude) convention. This choice also corresponds to (row, column) when dealing with mesh-grids.

Phase 5.2: From sensors to mesh-grids

In this phase, we describe a method for converting sensors' observations into mesh-grids. Although this process is devised for the dataset X, it could also be applied to any of the exogenous datasets $\{X_1, \ldots, X_p\}$. The approach to obtain the mesh-grids depends on the nature of the considered variable (see Section 3.1.1):

Continuous variables: Intuitively, observations of a continuous variable tend to vary smoothly over space and time, rather than changing abruptly within a short time (e.g., seconds or minutes) or distance (e.g., meters). In such cases, interpolation methods can be applied over either $\mathcal{A}$ or $\mathcal{T}$, i.e., spatial or temporal interpolation. Two simple yet effective interpolation methods are:

- Nearest-neighbor interpolation: This method assigns the value of the nearest observation. Therefore, it preserves the original range of values without generating new intermediate magnitudes.
- Linear interpolation: This method generates new intermediate values, but can only provide estimates for locations within the convex hull of the available points $\mathcal{G}$.

This limitation of the linear method can be problematic in some scenarios since we must devise estimations outside the convex hull. On the other hand, nearest-neighbor interpolation provides an additional benefit that the linear method does not: it is easily reversible by applying the same approach, provided that each region of the mesh-grid contains at most one element of $\mathcal{G}$. Nevertheless, we should always choose the interpolation method and its specific configuration based on the problem under study.

Discrete variables: In this case, the considered variable is still numerical, but it can only take values $x \in \mathbb{N}$. Therefore, the assumption of continuity is not fulfilled. In this case, we first aggregate sensor observations for each grid element and timestamp, summing the data points from sensors within every grid element. This step can be skipped if there is at most one element of $\mathcal{G}$ per grid element. Ideally, the ratio between the number of sensors and grid elements should not be too small, ensuring fair sensor coverage. However, in practice, this may not always be the case. Therefore, we must decide whether to interpolate as before, based on the sparseness of the resulting set of mesh-grids. Interpolation here has a different meaning since the computed grid values cannot be interpreted as direct estimates of actual sensor observations. Instead, they represent an approximation of the underlying discrete distribution of the variable over the grid.

4.1.6 Phase 6: Preservation and metadata

Once the set of mesh-grids $\mathcal{M}$ is ready, we store it for subsequent training and evaluation of the model. It is essential to use a robust framework that allows for

efficient storage of high-dimensional data, such as HDF5, NetCDF, or Parquet. Preserving all relevant metadata is of uttermost importance for reproducibility reasons. Metadata can be stored separately (e.g., in a JSON or YAML file) or embedded if the format allows (e.g., HDF5, NetCDF, or Parquet). In either case, it should always be accessible and kept up to date. The following are some recommended attributes to store as metadata:

- **General metadata**:
  - Dataset name.
  - Brief description of the dataset.
  - Variables that the dataset includes.
  - Author.
  - License.
  - Creation date.
- **Sensor metadata**:
  - Sensor type.
  - Sensor identifier.
  - Variables observed by each sensor.
  - Measurement units for each observed variable.
- **Time-related metadata**:
  - Temporal coverage $\mathcal{T}$.
  - Temporal resolution τ.
  - Time zone.
- **Spatial metadata**:
  - Spatial coverage $\mathcal{A}$.
  - Spatial resolution σ.
  - Geolocation of each sensor.
  - Geolocation of the mesh-grid elements $\mathcal{G}'$.
- **Processing metadata**:

- Scaling method.
- Interpolation method (if applicable).
- Missing data and imputation methods (if applicable).
- Software libraries and versions used.
- Other relevant aspects that arised during analysis.

4.1.7 Phase 7: Validation

Careful verification of the transformed data is fundamental at every phase, as Figure 4.1 suggests. Since phase 1 (EDA) is an initial analysis step rather than a transformation requiring validation, we exclude it from this phase. Such validations may include:

- **Phase 2 (Temporal resolution)**:
  - Ensure timestamps are uniformly spaced at intervals of τ.
  - Identify missing periods for discarding or for later estimation.
  - When converting to a finer temporal resolution, validate estimation consistency.
- **Phase 3 (Tabular dataset)**:
  - Check that the number of entries in the table is correct.
  - Ensure that the timestamps are as expected.
  - Validate geolocation consistency.
- **Phase 4 (Data scaling)**:
  - Verify variable ranges correspond to the expected ones.
  - Check variable distributions, based on the scaling method.
  - Ensure that original observations can be recovered through inverse scaling.
- **Phase 5 (The mesh-grid)**:
  - Validate variable ranges.
  - Check the consistency of raw sensor observations with nearby grid values.
  - Visualize individual mesh-grids.

    - Create animated visualizations of mesh-grids over time.

- **Phase 6 (Preservation and metadata)**:
    - Verify data integrity and keep regular backups.
    - Ensure metadata is complete and up to date.

4.2 Forecasting model

In Section 3.2, we formulated the spatio-temporal forecasting problem. One way to tackle this problem is to train independent models for each variable, sensor, and forecast horizon. However, this approach neglects the correlations between contiguous variables, sensors, and horizons. Arguably, a single model combining all these dimensions could achieve better results. The forecasted set $\mathcal{Y}^{t_\gamma}$ when $t_\gamma \in \mathcal{T}_{\text{test}}$ considers multiple timestamps, geolocations, and variables. This fact highlights the need for a model that can work with temporal and spatial information arranged as a single structure. The set of mesh-grids $\mathcal{M}$ we build in Section 4.1 aims to bridge this gap.

This section describes the proposed modeling framework and how it works with $\mathcal{M}$. We generally denote it as ST-F, which we later adapt to each use case. Section 4.2.1 describes the main component of the model: the spatio-temporal module. Afterward, Section 4.2.2 explains how the model can incorporate additional modules that work with exogenous data. The results of the different modules are combined in Section 4.2.3 to obtain the final prediction. The subsequent Section 4.3 will address how this model ought to fulfill the NFFs.

4.2.1 Spatio-temporal module

This module is the main one of the forecasting model and works with the set of mesh-grids $\mathcal{M}$. As we have discussed, the tensor $\mathcal{M}$ captures the temporal and spatial aspects of the sensor observations. Recurrent Neural Networks (RNNs) are a kind of NN that work with time-dependent data. Compared to the MLP, their main difference is that this network iterates over a time window of width n_x on each inference step. The primary issue with this kind of network is the vanishing gradient problem (Bengio et al., 1994). This pitfall makes the predictions too dependent on the last values of the sequence, which in turn makes the network forget the first ones. LSTM networks are a type of RNNs which have proved to behave better in this sense and to capture both long and short-term relations. Despite their benefits, LSTMs do not capture spatial correlations and are better suited for classical time series forecasting.

Traditionally, spatial correlations with NN have been addressed using CNNs. Thanks to their application to image classification problems, they became well-

known, particularly with the work by Krizhevsky et al. (2012). CNNs work with rank-3 tensors, such as red, green, and blue (RGB) images. Therefore, they can be applied to geolocated data but do not account for time. The ConvLSTM has been employed in recent years to bridge this gap (e.g., Agrawal et al., 2019; Shi et al., 2015). This kind of NN layer is obtained by substituting matrix multiplications at each gate of the LSTM with the convolutional operator. In this way, it combines the LSTM ability to find temporal correspondences with the extraction of two-dimensional (spatial) features of the convolutional operator.

The spatio-temporal module of the ST-F starts by feeding batches from the rank-4 tensor $\mathcal{M}$ to the ConvLSTM layer. There, it iterates over the temporal dimension of size n_x and produces a rank-3 tensor. The specific shape of this tensor depends on parameters of the ConvLSTM (number of filters, kernel size, strides, etc.). This tensor is then shrunk by a max-pooling layer and flattened for use in Section 4.2.3.

4.2.2 Exogenous modules

Exogenous data can improve the predictions of the ST-F model, as discussed in Section 3.3.4. We can incorporate exogenous modules corresponding to each dataset besides the main spatio-temporal module to achieve this goal. Each module extracts information from an exogenous dataset, later combined in Section 4.2.3. In this way, the ST-F model incorporates information from exogenous datasets in a natural way through independent modules.

The exogenous modules perform different operations based on the nature of the datasets:

- **Flat-extraction modules**: If the dataset is timestamp-independent, it can be fed to a fully-connected layer.
- **Recurrent modules**: If the dataset is a time series, the module can feed them to a recurrent layer, such as an LSTM.
- **Spatio-temporal modules**: If exogenous data is dense enough to build a mesh-grid, we can follow the same approach as for the main module (see Section 4.2.1).

4.2.3 Concatenation and prediction

Once the spatio-temporal and the additional modules have produced their feature vectors, the ST-F concatenates them. Thanks to this operation, we can build different variations of the ST-F model based on which modules are active. Afterward, the network passes that vector by several fully-connected layers and reshapes it into a rank-4 tensor. The next step is a deconvolution, also known as a transposed convolution and initially described by Zeiler et al. (2010) (see also

Dumoulin & Visin, 2018). This NN operation establishes the same connectivity patterns as the convolutional operator but in the opposite direction. The final step is to perform the inverse conversion of the one that generated the mesh-grids. This results in $n_y \cdot n_s \cdot n_v$ predictions for every $t_\gamma \in \mathcal{T}_{\text{test}}$. Figure 4.5 shows the complete generic model, where each input branch corresponds to a module.

4.3 Evaluation

This section goes over the different aspects of the model that we evaluate. First, Section 4.3.1 details how we calculate the error metric for every variable, sensor, and horizon. Afterward, we address NFFs: flexibility (Section 4.3.2), robustness (Section 4.3.3), extensibility (Section 4.3.4), and integrability (Section 4.3.5).

4.3.1 Error metrics

Even if the main focus of this work is the NFFs, we must first address the error of the model. The goal is to forecast:

$$\mathcal{Y}^{t_\gamma} = \left\{ y^t_{s,v} \mid t = t_\gamma + h_0, \ldots, t_\gamma + h_{n_y-1},\, v \in \mathcal{V},\, s \in \mathcal{S}_v \right\}$$

when $t_\gamma \in \mathcal{T}_{\text{test}}$, as the problem formulation holds (Section 3.2). Therefore, we consider the evaluation metric as a function of the prediction horizon, sensor, and variable. Some of the most common error metrics for regression problems are:

- *Mean Squared Error (MSE)*,
- *Root-Mean-Squared Error (RMSE)*,
- *normalized RMSE (nRMSE)*,
- *Mean Absolute Error (MAE)*, and
- *Mean Squared Logarithmic Error (MSLE)*.

We have adapted them for our problem as follows:

$$\text{MSE}\,(h,\, s,\, v) = \frac{1}{|\,\mathcal{T}_{\text{test}}\,|} \sum_{t_\gamma \in \mathcal{T}_{\text{test}}} \left(y^{t_\gamma+h}_{s,v} - p^{t_\gamma+h}_{s,v} \right)^2, \tag{4.4}$$

$$\text{RMSE}\,(h,\, s,\, v) = \sqrt{\text{MSE}\,(h,\, s,\, v)}, \tag{4.5}$$

$$\text{nRMSE}\,(h,\, s,\, v) = \frac{\text{RMSE}\,(h,\, s,\, v)}{\frac{1}{|\,\mathcal{T}_{\text{test}}\,|} \sum_{t_\gamma \in \mathcal{T}_{\text{test}}} y^{t_\gamma}_{s,v}}, \tag{4.6}$$

$$\text{MAE}\,(h,\, s,\, v) = \frac{1}{|\,\mathcal{T}_{\text{test}}\,|} \sum_{t_\gamma \in \mathcal{T}_{\text{test}}} |y^{t_\gamma+h}_{s,v} - p^{t_\gamma+h}_{s,v}|, \tag{4.7}$$

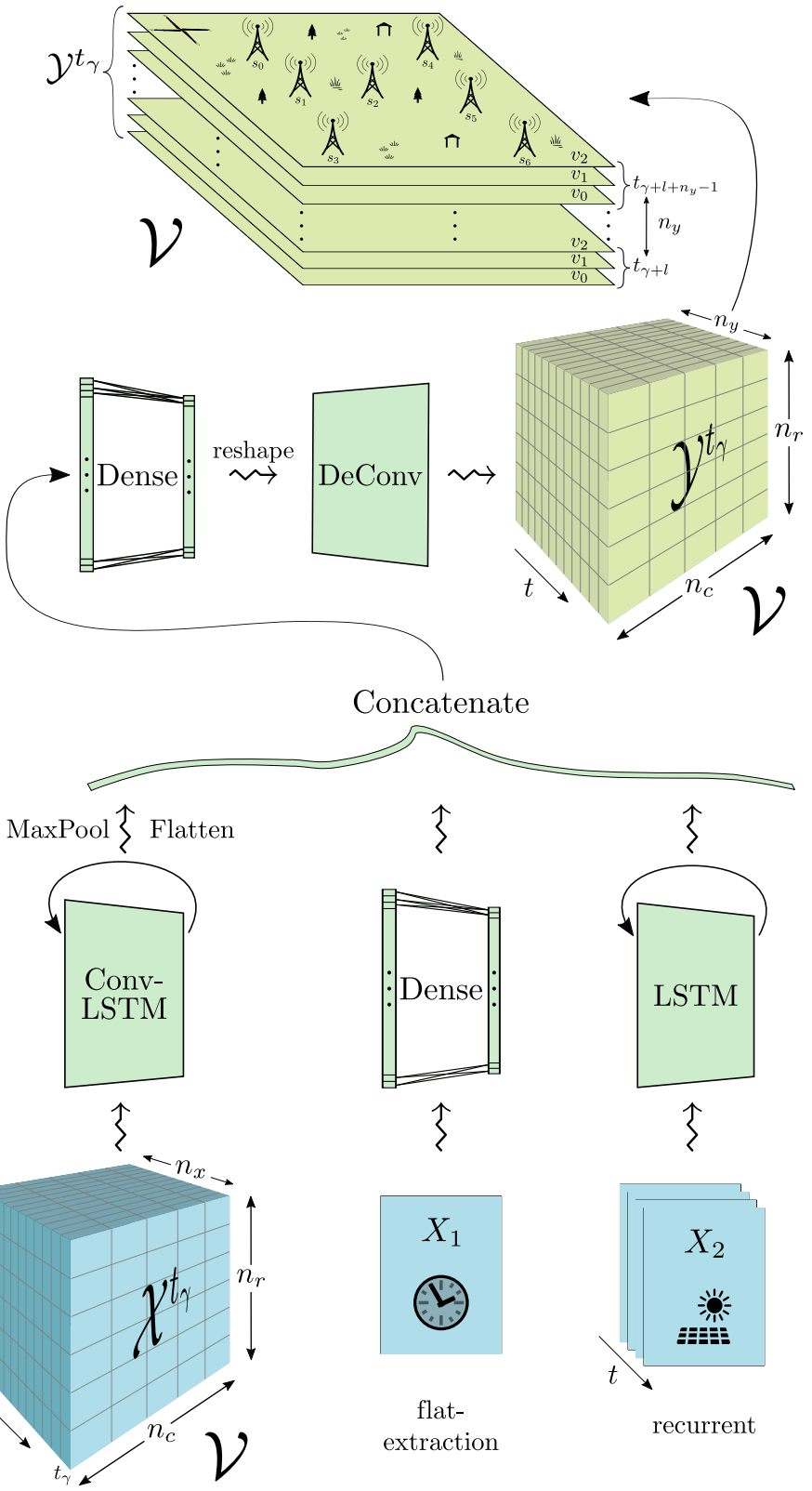


Figure 4.5: Proposed ST-F model. Flow moves upward, from the input $\{\mathcal{X}^{t_\gamma}, X_1, X_2\}$ to the output $\mathcal{Y}^{t_\gamma}$.

$$\text{MSLE}\,(h,\,s,\,v) = \frac{1}{|\,\mathcal{T}_{\text{test}}\,|}\sum_{t_\gamma \in \mathcal{T}_{\text{test}}} \left(\log\left(y_{s,v}^{t_\gamma+h}+1\right) - \log\left(p_{s,v}^{t_\gamma+h}+1\right)\right)^2, \tag{4.8}$$

where $h \in \left\{h_0, \ldots, h_{n_y-1}\right\}$ represents the forecast horizon, $s \in \mathcal{S}_v$ is a sensor that observes the variable $v \in \mathcal{V}$, $y_{s,v}^{t_\gamma+h} \in \mathcal{Y}^{t_\gamma}$ is a data observation (true value), and $p_{s,v}^{t_\gamma+h}$ represents the predicted value for the same timestamp, sensor and variable. These metrics provide a rank-3 tensor based on the horizon, sensor, and variable. To summarize these values, we calculate the median or mean across sensors for each horizon and variable. Afterward, we compute the mean across horizons, yielding one error value per variable for comparison purposes.

When it comes to comparing the error of two models, we use the *skill* criterion (Amaro e Silva & Brito, 2018; Yang, 2019). It indicates the relative improvement over a reference model as a percentage:

$$\text{Skill} \equiv \left(1 - \frac{\text{error}_{\text{proposed}}}{\text{error}_{\text{reference}}}\right) \cdot 100 \tag{4.9}$$

The error metric can be any of the previous ones. Positive skill values indicate a forecast improvement, while negative values imply a worsening. Zero skill indicates that the proposed model obtains the same error metric as the reference.

The model's skill strongly depends on the choice of hyperparameters at different levels:

- **Training**: optimizer, loss function, learning rate, batch size, and regularization techniques (e.g., dropout).
- **Mesh-grid**: interpolation and scaling methods, spatial resolution, etc.
- **Model architecture**: number of layers, neurons per layer, activation functions, attention methods, ablation studies, etc.
- **Spatio-temporal aspects**: number of filters and kernel size of convolutions, pooling size, temporal window width, etc.

Finding the best combination of hyperparameters through brute-force search can be a time-consuming task, especially when dealing with high-dimensional spatio-temporal data. Instead, this analysis should be guided by predefined heuristics or automated ML tools. Given the computational cost of training DL-based models that work with fine-grained mesh-grids, tools like Graphics Processing Units (GPUs) and strategies such as distributed training become necessary.

4.3.2 Flexibility

We described the flexibility NFF in Section 3.3.1 as a property of models that can accept a different number of inputs and outputs without changing its inner

structure. Therefore, to address whether a model is flexible, we must answer the question: can the model operate if the set $\mathcal{S}$ evolves over time? This property also implies that the set $\mathcal{G}$ can change as long as new geolocations lie within $\mathcal{A}$. Since the ST-F framework relies on the mesh-grids, it is detached from the number and distribution of the sensors. Hence, our model is inherently flexible, as seen in the use cases.

4.3.3 Robustness

A robust model produces reliable predictions even when a subset of the sensors fails to obtain some observations (Section 3.3.2). Note that a robust model is flexible by definition because it can operate even when the number of inputs changes. However, the opposite is not necessary: if a model can accept a variable number of inputs and outputs, it does not imply that it can recover from sensor failure.

We evaluate this NFF by calculating how the error degrades on $\mathcal{T}_{\text{test}}$ when an increasing number of sensors fail. Practically, the experiment follows these steps:

(1) Fix a variable v, observed by a maximum of n_s sensors.

(2) For each number of faulty sensors $i \in \{1, ..., \lceil n_s/2 \rceil\}$:

  (I) For each repetition $j \in \{1, ..., r\}$:

    (i) Choose i random sensors from the n_s available ones.

    (ii) For each $t_\gamma \in \mathcal{T}_{\text{test}}$, calculate the error without those sensors:

      (a) Load the $n_s - i$ available sensors for timestamp t_γ.

      (b) Build the grid based on observations from those sensors.

      (c) Feed it to the model, obtaining a prediction p.

      (d) Calculate the partial error of p with respect to the corresponding true value y that includes n_s sensors.

      (e) Go to step (ii) until all $t_\gamma \in \mathcal{T}_{\text{test}}$ are completed.

    (iii) Calculate the total error for repetition j with i sensors.

    (iv) Go to step (I) until r repetitions are completed.

  (II) Go to step (2) the $\lceil n_s/2 \rceil$ cases are completed.

(3) Finish evaluation.

The evaluation algorithm runs for any variable of interest v, simulating the failure of up to half of the available sensors. We randomly choose the failing sensors and

run the experiment for r repetitions. This results on a rank-4 tensor of error values, whose shape is $(\lceil n_s/2 \rceil, r, n_y, n_s)$. The final step is to compare the errors obtained to the original ones when the model has all the sensors available. We use the skill criterion for this, defined in Eq. 4.9. To summarize the resulting rank-4 tensor, we average the skill across repetitions and then calculate the median across the sensors.

4.3.4 Extensibility

The extensibility NFF refers to the feasibility of incorporating new variables into $\mathcal{V}$, as Section 3.3.3 discussed. This feature requires a refinement or tuning of the model. Therefore, we seek the ease of extending the model with new variables. In the case of the ST-F model, the spatio-temporal module deals with the tensor $\mathcal{M}$, whose last dimension corresponds to variables. Some of the use cases show how this fact favors extensibility.

4.3.5 Integrability

Section 3.3.4 defines integrability as the model's capability to integrate exogenous data. Section 4.2.2 describes how the ST-F model achieves this using independent modules. In practice, we additionally study what effect the activation of such modules entails on the error metric. This type of analysis is commonly referred to as an *ablation study*, where components are removed to determine their contribution to the overall system's performance. To find the best combination, we train and evaluate the model using all combinations of modules.

4.4 Summary

This chapter presents the methodology to address the problem described in Section 3.2 while incorporating the NFFs. In the first stage, we inspect the spatio-temporal data and decide how to build the mesh-grid based on seven phases (see Section 4.1):

- **Phase 1: EDA.** Understanding variables, sensors, geolocations, temporal granularity, and missing observations.
- **Phase 2: Temporal resolution.** Homogenizing temporal granularity across all time-dependent datasets.
- **Phase 3: Tabular dataset.** Building a comprehensive table that collects observations for all timestamps, sensors, and variables.
- **Phase 4: Data scaling.** Scaling all datasets that the ST-F model take as input.
- **Phase 5: Mesh-grid.** Building the tensor $\mathcal{M}$ based on the nature of the

variables.

- **Phase 6: Preservation and metadata.** Storing $\mathcal{M}$ together with all relevant metadata.
- **Phase 7: Validation.** Ensuring data transformations are applied consistently on every phase.

Afterward, we describe the proposed ST-F model in Section 4.2, which is based on different modules and their combination:

- **Spatio-temporal module**: Main module that is based on the application of the ConvLSTM to the set $\mathcal{M}$.
- **Exogenous modules**: Based on the kind of exogenous data, the additional modules can be recurrent, flat-extraction, or spatio-temporal.
- **Concatenation and prediction**: Combine the features extracted by each module and deconvolute them to obtain the final prediction.

In the final Section 4.3, we describe how to evaluate error metrics and NFFs of the model: flexibility, robustness, extensibility, and integrability. Figure 4.6 summarizes the complete methodology described in this chapter. In subsequent chapters, we apply our methodology to three forecasting use cases: solar irradiance, mobility demand, and air quality.

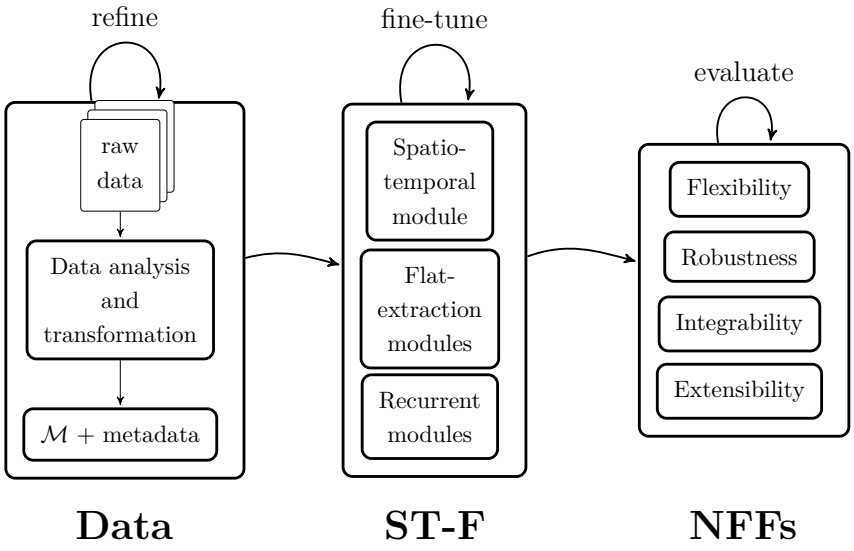


Figure 4.6: Summary of the complete methodology.

CHAPTER 5

First use case: Solar irradiance

The first use case that we study is solar irradiance forecasting. We focus on just one variable and no exogenous data. In addition to error metrics, we study flexibility and robustness NFFs. This chapter is based on our work (Prado-Rujas, García-Dopico, Serrano, & Pérez, 2021). In addition, we have gathered supplementary material and code in a repository (Prado-Rujas et al., 2020).

This chapter is organized as follows. Section 5.1 introduces the topic and presents a related theoretical background. Afterward, Section 5.2 applies our 7-phase methodology (see Section 4.1) to a solar irradiance dataset. Section 5.3 details the baselines and our proposal, the Spatio-Temporal Solar Irradiance Forecaster (ST-SIF). We describe our experiments in Section 5.4, and we discuss the results in Section 5.5.

5.1 Introduction and background

This section introduces the topic of solar irradiance forecasting. Section 5.1.1 explains the relevance of the topic and provides a summary of our contributions. Afterward, Section 5.1.2 presents a theoretical background of the subject.

5.1.1 Introduction

Over the last decade, solar energy has become decisive in drifting away from fossil fuels. The use of thermal and photovoltaic solar energy is steadily increasing in both industry and households (Zaręba et al., 2022). This fact has boosted research in solar irradiance forecasting. Obtaining reliable fine-grain energy predictions is crucial for developing many renewable energy systems. An example is optimizing the layout and orientation of the solar panels of a PV plant, which can significantly impact the overall amount of energy produced during its lifetime. As the number of PV plants grows vastly and they are integrated into the electric grid, anticipating the amount of energy produced is needed to avoid overloads and foresee energy

shortage. In addition, accurate forecasts are needed for effectively managing batteries that store the energy produced.

As discussed in Section 2.2, the literature explores many different methods for solar irradiance forecasting. However, this task inherently presents many angles of complexity. On the one hand, time granularity directly impacts the variability of the measurements. For example, a finer temporal granularity τ translates into spikes that look like noise. In addition, the input width n_x that we consider significantly affects the forecast performance. On the other hand, forecast horizons must be established based on the purpose of the predictions. It is also noteworthy that solar irradiance received in a particular area is highly correlated with the irradiance recorded nearby. This fact can provide valuable information to predictive models.

Despite the extensive literature on the topic, challenges still need to be tackled. Firstly, the model's predictive capabilities should be ranked based on a standard and robust metric, due to the numerous dimensions of this problem. The forecast skill is widely accepted, becoming more prevalent in the field as noted by Yang (2019). However, current works should also address flexibility and robustness features.

In this chapter, we apply our methodology to solar irradiance forecasting. For this, we develop the ST-SIF, a DL-based model to predict short-term standardized GHI or Clear-Sky Index (CSI) with no exogenous inputs. Therefore, $n_v = 1$, i.e., the model only works with one variable. In addition, we study the error, flexibility, and robustness (as described in Section 3.3). None of the published works cover all these characteristics to the best of our knowledge, as discussed in Section 2.2. As will be seen, the ST-SIF model contributes along the following lines:

- Harnessing of the spatial and temporal aspects of the solar irradiance forecasting task. The model works simultaneously with several nearby sensors as input and output. Furthermore, it considers multiple previous and future timestamps by leveraging recurrent layers.
- Flexibility in the interface while still providing competitive error figures. The number of sensors can vary without changing the model's structure, thanks to solar irradiance mesh-grids.
- Recovery of missing irradiance observations by filling them based on nearby stations or previous records, making it robust to sensor failure.

The purpose of this framework is to forecast solar irradiance for up to an hour ahead reliably. Such a system can be deployed in PV plants or smart cities, where flexibility and robustness are central. The proposed framework can play a fundamental role in making critical decisions, such as:

- Location of a PV plant for maximizing profitability.

- Layout of the solar panels on a PV plant.
- Optimization of energy accumulation and distribution.
- Avoiding network overloads and power cuts.
- Electricity price forecasting.
- Planning of solar panels maintenance.

5.1.2 Theoretical background

Global Horizontal Irradiance (GHI) is the total amount of shortwave radiation received by a horizontal surface, expressed in $\mathrm{W/m^2}$. It can be formulated in terms of the Diffuse Horizontal Irradiance (DHI) and the Direct Normal Irradiance (DNI) as:

$$\mathrm{GHI} \equiv \mathrm{DHI} + \mathrm{DNI} \cdot \cos(z), \tag{5.1}$$

where z is the zenith angle of the sun. Let $\mathrm{GHI}(t) = x_s^t$ be the measured GHI at timestamp t by a fixed sensor s. Also, let $\mathrm{GHI}_{cs}(t)$ be the GHI for a clear sky at the same timestamp t and location as s. We can interpret GHI_{cs} as the theoretical maximum GHI. The *Clear-Sky Index (CSI)* can be expressed as:

$$\mathrm{CSI}(t) \equiv \frac{\mathrm{GHI}(t)}{\mathrm{GHI}_{cs}(t)}\,, \tag{5.2}$$

CSI provides a natural way to normalize solar irradiance data, obtaining values above 0 and (mostly) below 1. Furthermore, CSI removes seasonal trends from the GHI measurement caused by the sun's position in the sky.

The *persistence model* is the most basic time series forecasting approach. As such, it is used as a baseline model for solar irradiance forecasting. It is based on the assumption that the forecasted variable will remain unchanged on the forecast horizon h. If $\mathrm{pers}(t+h)$ denotes the forecasted value for the horizon h, the persistence model is defined as:

$$\mathrm{pers}(t+h) \equiv \mathrm{GHI}(t) \tag{5.3}$$

Smart persistence refers to a persistence model on the CSI:

$$\mathrm{smart}(t+h) \equiv \mathrm{GHI}(t) \cdot \frac{\mathrm{GHI}_{cs}(t+h)}{\mathrm{GHI}_{cs}(t)} = \mathrm{CSI}(t) \cdot \mathrm{GHI}_{cs}(t+h) \tag{5.4}$$

Similarly to the persistence model, its prediction is based on the current measured value. However, it also accounts for the expected variation based on the clear-sky irradiance. A limitation of smart persistence is that it assumes positive values, which is usually not the case when data is standardized. Therefore, we can generalize Eq. 5.4 to allow negative values as follows:

$$\mathrm{smart}(t+h) \equiv \mathrm{GHI}(t) + |\,\mathrm{GHI}(t)\,| \cdot \left(\frac{\mathrm{GHI}_{cs}(t+h)}{\mathrm{GHI}_{cs}(t)} - 1\right), \tag{5.5}$$

which indeed provides the same expression as Eq. 5.4 when GHI(t) is positive.

A standard metric must be established when comparing the goodness of several models. It is insufficient to rely solely on common metrics like RMSE or MAE to fully capture intrinsic aspects of solar forecasting, such as location, timestep, and forecast horizon (Yang, 2019). The skill criterion defined in Eq. 4.9 is widely accepted as a reference metric in solar irradiance forecasting. As discussed in Section 4.3.1, it indicates the relative improvement over a reference model as a percentage. This way, we can compare models tailored for different ROIs and temporal granularities against a common and simple model. Typically, RMSE or MAE are used as error metrics for the skill (see Eq. 4.9). The simple or smart persistence model is usually the reference model. In the latter case, the skill metric is denoted as *smart skill*. Depending on various factors, any of the two reference models can be better than the other one. Our analysis found that smart persistence outperformed simple persistence for horizons beyond 30 minutes. Eventually, simple persistence was used as the reference model since the differences in skill were never above 4.5 %.

5.2 Data analysis and transformation

In this section, we apply the 7-phase data analysis and transformation methodology defined in Section 4.1 to an irradiance dataset. For our experiments, we use the Solar Measurement Grid dataset of the Hawaiian island of Oahu provided by NREL (Sengupta & Andreas, 2010).

5.2.1 Phase 1: Exploratory Data Analysis

NREL's Solar Measurement Grid located at Oahu (Sengupta & Andreas, 2010) is comprised of 17 LICOR LI-200 pyranometers. These sensors collect GHI, among other variables. The region is typically characterized by scattered clouds, with strong winds blowing mainly from the northeast. These two conditions translate into high solar irradiance oscillations. Previous works have analyzed this topic in detail (Hinkelman, 2013). In the following, we answer the EDA matters that we laid on Section 4.1.1:

1. Sensors

The set $\mathcal{S}$ is comprised of 17 sensors, whose identifiers are depicted in Figure 5.1.

2. Variables

As mentioned in Section 5.1.1, the set $\mathcal{V}$ only comprises one variable: GHI. Some experiments work with GHI and some others with CSI, which are both continuous. As discussed in Section 5.1.2, CSI is derived from GHI and GHI_{cs}.

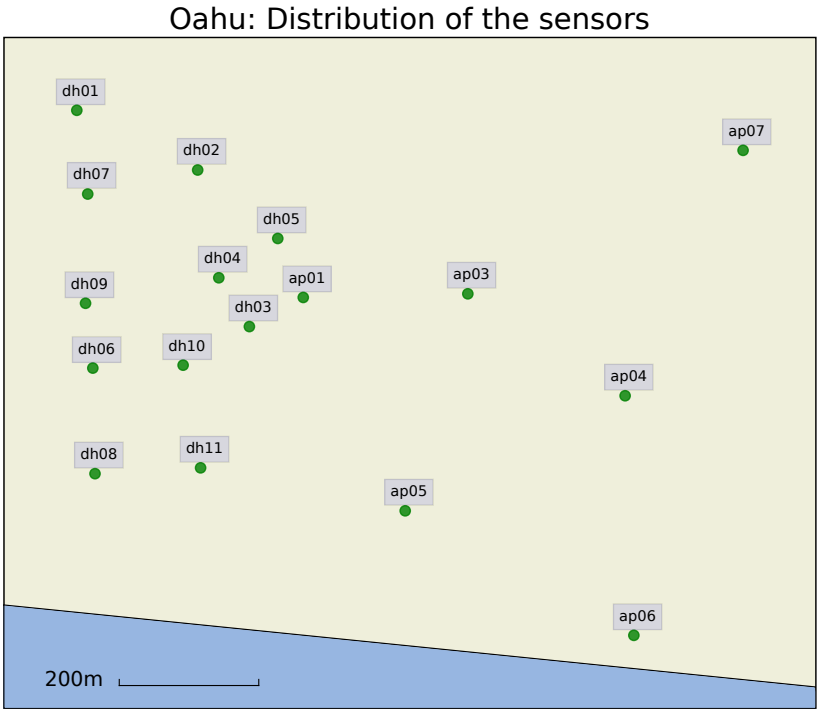


Figure 5.1: Location of the 17 pyranometers at Oahu, Hawaii.

3. Variable units

GHI is measured in $\mathrm{W/m^2}$. CSI is unitless.

4. Variables observed by every sensor

All sensors measure GHI. In this chapter, we work with a single variable.

5. Geolocation of every sensor

We show the spatial distribution of the 17 sensors in Figure 5.1. The median distance between sensors is 380 m. The specific geolocations (longitude and latitude) can be found in our repository (Prado-Rujas et al., 2020).

6. Spatial coverage

The sensors cover an area of roughly 1 $\mathrm{km^2}$, as Figure 5.1 shows.

7. Temporal granularity

All sensors collect measurements at 1 Hz. Therefore, $\tau_{s,v} = 1$ s for every $s \in \mathcal{S}$ and $v = \mathrm{GHI}$.

8. Temporal coverage

The data were measured from March 18$^{\text{th}}$, 2010 to October 31$^{\text{st}}$, 2011, spanning nearly 20 months. GHI values were measured daily between 5 am and 8 pm (UTC-10).

9. Missing observations

Considering the temporal coverage, the dataset comprises 593 days. With 15 hours daily, this accounts for almost 9000 hours. Since $\tau_{s,v} = 1$ s, every sensor measured over 32 million irradiance values. Missing observations are represented as -99 999 in the raw data. Most sensors have a small number of missing values (see Table 5.1). However, over 180 days were empty for *ap03*, which we will fix in Section 5.2.3.

10. Main statistics

Table 5.1 summarizes the main statistics of the raw observations from the 17 pyranometers. Even if the sensors are close by, there are differences in their values, as can be seen in the table. As mentioned earlier, solar irradiance presents high variability in this region. This fact is supported by the similarity between the mean and standard deviation and that maximum values are nearly three times higher than the 75$^{\text{th}}$ percentile. In addition, Figure 5.2 depicts the mean GHI across all sensors. The red line corresponds to the mean over the 593 days; the other three are sample days with different conditions. The top figure corresponds to $\tau = 1$ s and the bottom to $\tau = 1$ min. We further discuss the temporal resolution conversion in the next phase (Section 5.2.2).

5.2.2 Phase 2: Temporal resolution

In Section 5.2.1, we identified that $\tau_{s,v} = 1$ s for every $s \in \mathcal{S}$ and $v =$ GHI. This temporal granularity is too fine for forecasting solar irradiance due to its small variation every second. For example, see the values for March 18$^{\text{th}}$ 2010 in Figure 5.2a. Therefore, we convert it to a coarser value: $\tau = 1$ min. To do so, we average every 60 consecutive observations, as described in Section 4.1.2. This conversion drastically decreases the number of observations of every sensor to over half a million. All the models considered in this chapter work with $\tau = 1$ min.

5.2.3 Phase 3: Tabular dataset

At this point, we have performed an EDA and converted the data into a standard temporal resolution. In this phase, we combine all sensor observations into a tabular structure as discussed in Section 4.1.3. This way, we can index observations using the timestamp, the sensor identifier, and the variable. The result resembles Table 4.3. For this use case, $n_v = 1$ variable, $n_s = 17$ sensors, and $n_t = 900\,\text{timestamps/day} \cdot 593\,\text{days} = 533\,700$ timestamps. This phase yields

Table 5.1: Summary of the pyranometers raw data (March 2010 – October 2011).

Sensor	NA	mean	std	min	25 %	50 %	75 %	max
ap01	0.1	369.8	347.9	-0.4	41.1	279.3	623.6	1637.1
ap03	31.5	364.3	350.2	0.0	31.8	270.4	618.5	1729.7
ap04	0.0	370.8	348.8	-0.4	40.9	280.6	625.4	1673.7
ap05	0.0	374.4	355.1	0.0	40.5	279.9	630.1	1684.5
ap06	0.0	376.3	355.9	-0.4	40.6	282.6	634.3	1688.9
ap07	0.1	375.8	357.3	-0.4	38.2	280.1	634.6	1743.3
dh01	0.3	370.1	349.5	-0.4	37.3	278.1	626.0	1711.8
dh02	0.2	371.3	351.7	-0.7	37.4	277.5	627.9	1712.6
dh03	0.2	377.1	363.1	-0.5	36.8	275.4	636.2	1776.9
dh04	0.2	370.5	350.4	-0.4	40.4	277.4	624.9	1700.4
dh05	0.2	374.4	354.8	-0.4	40.2	279.4	631.9	1679.1
dh06	0.2	375.1	354.7	-0.4	39.8	281.1	633.5	1695.7
dh07	0.2	376.5	356.3	-0.4	40.9	281.3	635.8	1695.8
dh08	0.2	371.3	351.5	-0.4	40.3	277.7	625.9	1690.0
dh09	0.2	375.3	363.0	-0.5	36.3	271.8	634.1	1774.3
dh10	0.2	374.8	353.1	-0.4	41.5	282.4	632.4	1683.5
dh11	0.2	371.5	351.1	-0.4	39.3	279.1	626.2	1671.9

Abbreviations are as follows. NA: percentage of not available observations, min: minimum, max: maximum, std: standard deviation, X %: X^{th} percentile. Except for NA, units are W/m^2.

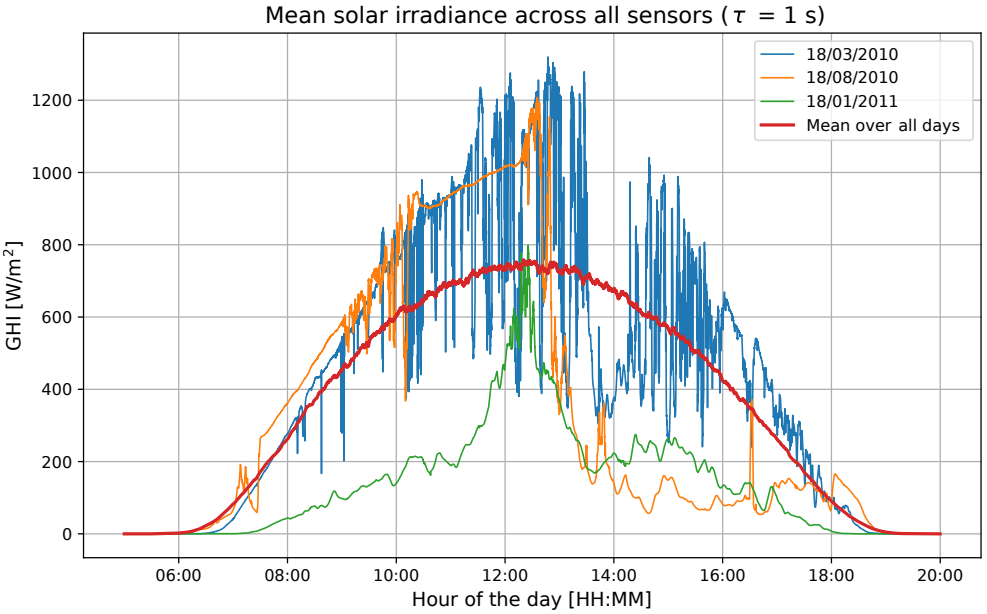
a table of over 9 million GHI observations.

In Section 5.2.1, we mentioned that a small number of observations is missing, except for *ap03* (see Table 5.1). Instead of discarding this sensor, we estimated missing observations. We filled in missing values by taking the non-empty closest value in time from the 30 preceding seconds of the same sensor. If the previous procedure failed to find a value, we took the non-empty measurement from the nearest sensor at the same timestamp.

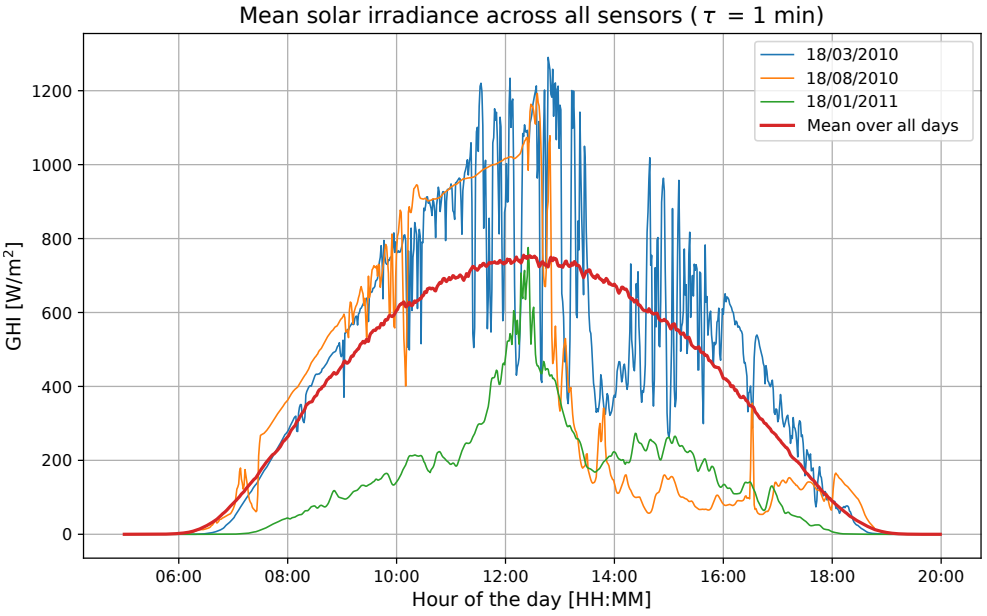
5.2.4 Phase 4: Data scaling

As discussed in Section 4.1.4, data are often scaled when working with NNs. This is because they tend to perform better when modeling values near zero. Therefore, we standardized GHI measurements following Eq. 4.2. To apply the formula, we computed each sensor's mean and standard deviation to account for calibration differences. Note that standardized GHI is unitless.

CSI provides a natural way to normalize GHI, as anticipated on Section 5.1.2. Thus, we built a CSI dataset equivalent to the one of Section 5.2.3. We used the Haurwitz clear-sky model to obtain the CSI (Haurwitz, 1945, 1946), which is implemented in the pvlib Python library (Holmgren et al., 2018). The code to reproduce these calculations is in the repository (Prado-Rujas et al., 2020). The



(a) Mean GHI with $\tau = 1$ s.



(b) Mean GHI with $\tau = 1$ min.

Figure 5.2: Mean GHI measured across all sensors. The red line shows the overall mean across all sensors for all days. The blue, orange, and green lines represent the mean GHI across sensors for different days with varying irradiance conditions.

CSI is of interest for some practical applications, mainly because it removes the inherent seasonality of solar irradiance, easing the prediction task. As will be seen in Section 5.4, the models that work with CSI obtain slightly better results than for standardized GHI.

5.2.5 Phase 5: The mesh-grid

Time and space are two key components to consider when forecasting solar irradiance. Whereas most studies focus on the former, the spatial aspect should also be exploited, as discussed in Section 5.1.1. Location is particularly influential for short-term forecasting, as will be studied in Section 5.4.3. In this section, we utilize phase 5 of our methodology (Section 4.1.5) to leverage the spatial aspect of solar irradiance forecasting. We initially fix the spatial resolution of the grid based on the sensors' geolocations and relative distances. Afterward, we build the solar irradiance mesh-grids for the spatio-temporal forecasting model.

Phase 5.1: Spatial resolution and grid placement

Choosing the spatial resolution of the mesh-grid $\sigma = (\sigma_y, \sigma_x)$ is not straightforward. We must consider the spatial coverage and the distances between sensors for this. First, we fix the spatial coverage by overlaying a rectangular area over the sensors. As described in Section 4.1.5, we add an offset ε to the sensors' minimum and maximum latitudes and longitudes to avoid leaving sensors on the border. Since the median distance between sensors is 380 m, we will choose σ below that value.

We consider two different grid shapes (n_r, n_c) in the experiments and study their differences: (8, 8) and (10, 10). For the first case, we choose an offset of 0.0006^o (about 65 m) to adjust as much as possible to the ROI. Then, we calculate the mesh-grid position as a regular partition of the spatial coverage, obtaining σ =(116, 134) meters. For the 10 by 10 case, we fix the offset to 0.001^o (about 110 m). Therefore, this yields a finer spatial granularity of σ =(102, 116) meters. We choose these shapes so that the mesh-grids represent real recorded values (about 26.6 % and 17 %, respectively).

Phase 5.2: From sensors to mesh-grids

In this phase, we explain how we build the solar irradiance mesh-grids. We applied the procedure to the standardized GHI values, but it could also be used for CSI. In the previous section, we fixed σ and the geolocations of the (8, 8) and (10, 10) mesh-grid elements. In this phase, we applied two-dimensional nearest-neighbor interpolation on the 17 sensor observations to build the grid for every timestamp. Figure 5.3 shows an example of a mesh-grid built in this way. There, it can also be seen that the method yields grids with pronounced steps, which resemble the shadows that clouds cast over the ground. Another benefit of the nearest-neighbor method is its computational simplicity, which will be discussed later. As with the rest of the section, Prado-Rujas et al. (2020) gathers the code to reproduce this transformation.

This phase is vital for the flexibility and robustness NFFs that we will study later. Including new sensors improves the quality of the solar irradiance mesh-grids,

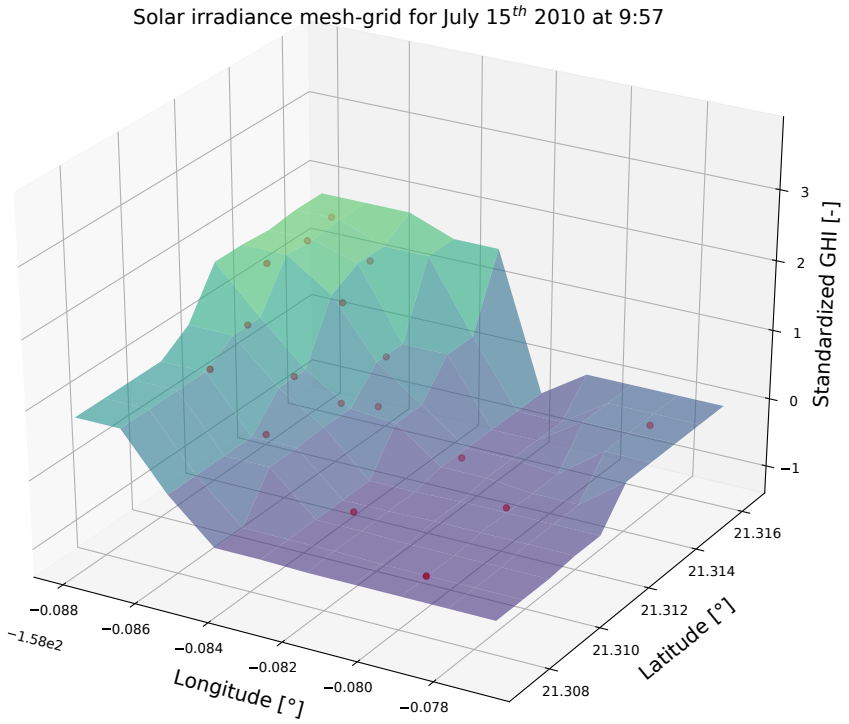


Figure 5.3: Solar irradiance mesh-grid for a specific timestamp. Red dots represent standardized GHI observations from pyranometers, and the surface depicts the interpolated mesh-grid.

which, in turn, should enhance the predictive error. The only requirement new sensors must meet is to be located within the spatial coverage $\mathcal{A}$. As for robustness, the interpolation operation fills the gaps produced by failing sensors. Besides, a model that works with mesh-grids does not need to be reframed when the set $\mathcal{S}$ changes. Section 5.4 further studies these ideas.

5.2.6 Phase 6: Preservation and metadata

The format and structure into which we preserve the data are essential to ease experimentation and reproducibility (see Section 4.1.6). We use HDF5 due to its hierarchical nature and the convenience of keeping metadata next to the data. Furthermore, it allows for different data compression levels and sliced data loading. We produced the following datasets:

- **GHI with filled missing values**: This accounts for 593 days of 900 observations from 17 sensors.
- **Standardized GHI**: This corresponds to the same observations, standardized as described in Section 5.2.4.
- **CSI**: This corresponds to the CSI, obtained as described in Sections 5.1.2

and 5.2.4.

- **Standardized GHI 8×8 mesh-grids**: This accounts for 593 days of 900 mesh-grids of shape (8, 8).
- **Standardized GHI 10×10 mesh-grids**: This accounts for 593 days of 900 mesh-grids of shape (10, 10).

In addition, we keep as metadata: GHI units, sensor identifiers, sensor geolocations, sensor statistics, temporal coverage, temporal resolution, spatial coverage, spatial resolution, data source, etc.

NREL's Solar Measurement Grid data (Sengupta & Andreas, 2010) are subject to distribution restrictions. Therefore, we published a Jupyter Notebook in our repository (Prado-Rujas et al., 2020) to reproduce the transformations of the raw data into the solar irradiance mesh-grids.

5.2.7 Phase 7: Validation

It is also essential to carry out data validation on every phase at different levels, as discussed in Section 4.1. The Jupyter Notebook published in our repository (Prado-Rujas et al., 2020) gathers some of these verifications. Initially, we checked that the raw data were well formed and included all timestamps. We also computed distances between sensors and the main statistics of their measured values. Furthermore, we carried out an analysis to depict differences between simple and smart persistence (Section 5.1.2) and came up with the generalized smart persistence (Eq. 5.5). As for the CSI calculation, we considered different clear-sky models from the literature and their implementations. On every phase, we plot different graphs to validate our data transformations. Daily mesh-grids animations were particularly helpful in visualizing solar irradiance behavior (see Prado-Rujas et al., 2020).

5.3 Forecasting model

In this section, we present the developed solar forecasting models. Initially, Section 5.3.1 describes baseline models that do not consider sensor geolocations. Afterward, Section 5.3.2 presents our ST-SIF model.

5.3.1 Baselines

Before delving into models that work with solar mesh-grids, we developed several NNs baselines that do not consider sensor locations. They all combine observations from the 17 sensors as input and forecast irradiance for the same 17 locations. The models provide forecasts for several temporal horizons, as we shall see in Section 5.4. After training, their RMSE is evaluated as described by Eq. 4.5

(see Section 4.3).

The most basic baseline is the MLP or Fully Connected (FC) network, in which every neuron is connected to all neurons from the previous layer. Although they do not work with time series directly, this kind of data can be reshaped and fed to this network (as pointed out by Alzahrani et al., 2017). Instead, RNNs work naturally with such data. The network iterates over a temporal window of fixed width n_x used as input for each inference step. The input for each inference step consists of a fixed-width temporal window of size n_x. The main issue with this kind of network is the vanishing gradient problem, as discussed in Section 4.2.1. LSTMs are a type of RNNs that can capture both long and short-term relations. BiLSTMs extend the latter by performing both a forward and a backward pass on each time window.

The specific parameters of the baseline models are as follows:

- **FC**: The network takes a tensor of shape (n_x, n_s) values, flattens it, and passes it through 4 dense layers of 68, 136, 68, and $n_y \cdot n_s$ neurons. Finally, the output is reshaped to have (n_y, n_s) values. The activation function for the dense layers is the REctified Linear Unit (ReLu), except for the last one, which is linear.

- **RNN**: The model takes a tensor of shape (n_x, n_s) values and passes it through a simple RNN layer of 16 neurons and a dense layer of $n_y \cdot n_s$ neurons. Finally, the output is reshaped to have (n_y, n_s) values. Both activation functions are linear in this case.

- **LSTM**: The network takes a tensor of shape (n_x, n_s) values and passes it through two LSTM layers of 16 and 34 neurons, respectively. The first of the two LSTMs returns the intermediate n_x sequences so that the second one can iterate over them. Afterward, the tensor is fed to two dense layers of 32 and $n_y \cdot n_s$ neurons to finally reshape it into (n_y, n_s) values. The activation function of the LSTMs is the hyperbolic tangent and linear for the dense ones.

- **BiLSTM**: The model takes a tensor of shape (n_x, n_s) values and passes it through a BiLSTM layer of 128 neurons. Then, the tensor is fed to a dropout and dense layers of 128 and $n_y \cdot n_s$ neurons, respectively. Finally, the tensor is reshaped into (n_y, n_s) values. The dropout rate is 0.5, the LSTM activation is the hyperbolic tangent, and the dense activation is linear.

5.3.2 ST-SIF

Section 5.3.1 presents several baseline models that do not consider the spatial component of solar forecasting. However, they combine sensor observations in the input and output on every timestamp. As discussed in Section 5.1.1, observations

from nearby sensors can improve the model's forecasts. In this section, we describe the ST-SIF model, which is based on Section 4.2.1 of the methodology.

Figure 5.4 shows the end-to-end solar irradiance forecasting procedure, whose steps are as follows:

1. On a certain timestamp t_γ, the $n \leq n_s$ available solar irradiance observations are collected.
2. These values are used to interpolate an irradiance mesh-grid of shape $n_r \times n_c$.
3. This grid is stacked to the $n_x - 1$ last ones to conform $\mathcal{X}^{t_\gamma}$, which is the input to the network and has shape (n_x, n_r, n_c).
4. $\mathcal{X}^{t_\gamma}$ is passed by C ConvLSTM layers.
5. The resulting four-dimensional tensor is flattened, and D dense layers are applied.
6. After reshaping, a tensor of shape (n_y, n_r, n_c) is obtained.
7. The output $\mathcal{Y}^{t_\gamma}$ is obtained by reversing the interpolation, which yields the $n_y \cdot n_s$ forecasted solar irradiance values.

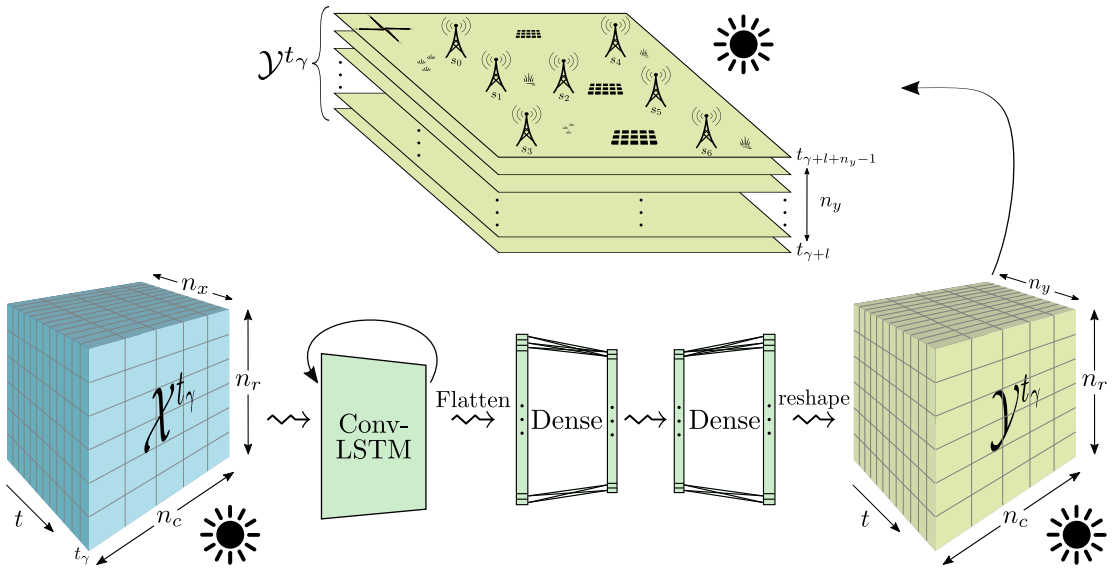


Figure 5.4: Spatio-Temporal Solar Irradiance Forecaster (ST-SIF) model.

In the presented experiments related to the ST-SIF, we used one ConvLSTM layer and two dense ones. The ConvLSTM has four filters, kernel size of $(2, 2)$, and hyperbolic tangent activation. The dense layers have 32 and $n_y \cdot n_r \cdot n_c$ neurons, respectively, and linear activation. It contains 117 248 trainable parameters when configured with $n_x = 10$, $n_r = 10$, $n_c = 10$, and $n_y = 4$. As for the error calculation, we apply Eq. 4.5 for every horizon and sensor. This approach provides RMSE

and skill values comparable to the ones obtained by the models described in Section 5.3.1.

5.4 Experiments and evaluation

This section presents the experiments related to the solar irradiance use case and their results. Initially, we describe the experimental setup and training parameters in Section 5.4.1. Afterward, Section 5.4.2 provides error values for the baseline and ST-SIF models. Then, Section 5.4.3 studies the correlation between sensor location and forecast skill. Finally, we discuss flexibility and present robustness experiments in Section 5.4.4.

5.4.1 Experimental setup

Regarding the implementation, we used the Keras library (Chollet et al., 2015) with the Tensorflow backend (Abadi et al., 2015). Additionally, NumPy, Pandas, and PyTables Python libraries were used for data manipulation and storage. We developed several Python scripts that work together to train and test models with different parameter configurations semi-automatically, available in our repository (Prado-Rujas et al., 2020). The software allows the variation of multiple parameters, such as the target variable, model, n_x, n_y, optimizer, learning rate, loss function, etc. Concretely, we used the following parameters in the experiments:

- **Variables**: Standardized GHI or CSI, so $n_v = 1$ as discussed in Section 5.1.1.
- **Temporal granularity**: $\tau = 1$ minute, as discussed in Section 5.2.2.
- **Models**: The parameters of the baseline and ST-SIF models were already discussed in Sections 5.3.1 and 5.3.2. The repository (Prado-Rujas et al., 2020) gathers Keras model graphs for all baselines and the ST-SIF.
- **Number of input timestamps**: $n_x = 10, 20$, or 60 minutes.
- **Forecast horizons**: $h \in \{1, 11, 31, 61\}$ minutes, so $n_y = 4$.
- **Number of sensors**: $n_s = 17$.
- **Mesh-grid shapes**: 8×8 and 10×10.
- **Optimizers**: After testing the available optimizers in Keras, we eventually chose Adam for the baselines and Adadelta for the ST-SIF.
- **Learning rate**: 0.001 for the baselines and 1 for the ST-SIF.
- **Loss function**: MSE.
- **Batch size**: 32.

- **Number of epocs**: 30.

As described in Section 3.2, we split the temporal coverage into three disjoint sets. We chose the partition into training, validation, and testing to cover different seasons. The testing set comprises about 16 % of the total days, including April 2010, December 2010, and July 2011. We trained the model on the remaining months and used 20 % of each month for validation. As for the hardware, we trained all models on an Ubuntu machine with 12 cores and 24 logical processors.

5.4.2 Error metrics

This section presents the skill of the baselines and ST-SIF models. Even if our main interest lies upon NFFs, models should reach reasonable error levels. As discussed in Section 5.1.2, we use the forecast skill as the error metric to compare different models. All skills are calculated with respect to the persistence model. We first calculate RMSE (Eq. 4.5) and then apply Eq. 4.9 on every horizon and sensor. This yields $n_y \cdot n_s = 4 \cdot 17 = 68$ skills for each model. We calculate the median across sensors to obtain a skill value per horizon and then calculate the mean of those n_y values.

Tables 5.2 and 5.3 present the skills of the baseline models when forecasting CSI and standardized GHI, respectively. Both tables show that the median skills range between 14.4 % and 35.5 %, depending on factors such as horizon and n_x. Furthermore, baselines obtain higher skills for CSI than for standardized GHI, except on the first horizon. Table 5.2 shows that skill tends to improve with further horizons, but the contrary happens for Table 5.3. These differences between variables may be because CSI removes the seasonality of solar irradiance, which could help predict further horizons.

Table 5.4 shows the skill scored by the ST-SIF model when forecasting standardized GHI. As discussed, we studied mesh-grid shapes of 8×8 and 10×10. Their skills are comparable to those obtained by the baselines on Table 5.3 since we calculate the skill at the sensor level. Indeed, skills are akin, and thus the use of mesh-grids does not penalize the skill of the ST-SIF. Regarding the use of mesh-grids, we carried out additional experiments with two and three-dimensional CNNs, which obtained poorer results. We are not showing them here for clarity, but we have gathered their results in our repository (Prado-Rujas et al., 2020).

Figure 5.5 presents the true and predicted standardized GHI for one of the baselines. Specifically, the figure corresponds to the BiLSTM with $n_x = 60$ min, for sensor *dh10* on April 23$^{\text{rd}}$, 2010. The selected date showcases two weather situations: stable irradiance until around 10:30 and considerable variation afterward. Scattered clouds are likely to cause this situation, which are quite common in the area, as mentioned in Section 5.2. The figure shows that each horizon constrains the prediction timestamps. For example, for $h = 1$ min they span from 6:00 to

Table 5.2: Skill of the baseline models when forecasting CSI.

Variable	n_x [min]	Baseline	Median skill [%] across $\mathcal{S}_v$ 1 min	11 min	31 min	61 min	Mean [%]
CSI	10	FC	19.9	21.9	23.8	27.5	23.3
		RNN	23.3	14.6	19.1	23.0	20.0
		LSTM	26.7	21.9	23.4	27.3	24.8
		BiLSTM	26.1	22.8	24.8	28.6	25.6
	20	FC	14.8	22.3	24.5	28.2	22.5
		RNN	24.4	14.4	19.6	23.6	20.5
		LSTM	27.3	24.3	25.6	29.5	26.7
		BiLSTM	26.2	23.9	25.8	29.9	26.5
	60	FC	17.2	24.7	22.5	29.7	23.5
		RNN	24.7	16.8	21.1	22.4	21.2
		LSTM	27.4	27.7	25.4	32.0	28.1
		BiLSTM	25.3	27.1	25.7	32.0	27.5

Skill is computed with respect to the persistence model (see Section 5.1.2). The median skill is calculated across sensors for each forecast horizon. The last column is the mean skill across horizons.

Table 5.3: Skill of the baseline models when forecasting standardized GHI.

Variable	n_x [min]	Baseline	Median skill [%] across $\mathcal{S}_v$ 1 min	11 min	31 min	61 min	Mean [%]
Standardized GHI	10	FC	27.3	18.7	18.9	19.5	21.1
		RNN	35.5	17.7	15.8	14.4	20.9
		LSTM	33.1	19.8	19.3	19.7	23.0
		BiLSTM	33.0	20.1	19.7	20.3	23.3
	20	FC	29.8	19.3	19.0	19.7	22.0
		RNN	35.3	18.5	16.4	14.7	21.2
		LSTM	31.8	20.2	19.7	20.7	23.1
		BiLSTM	32.9	20.2	19.9	21.3	23.6
	60	FC	22.4	19.2	19.9	22.6	21.0
		RNN	35.3	18.4	16.8	16.3	21.7
		LSTM	31.3	20.6	21.9	25.9	24.9
		BiLSTM	31.6	19.8	21.2	25.4	24.5

Skill is computed with respect to the persistence model (see Section 5.1.2). The median skill is calculated across sensors for each forecast horizon. The last column is the mean skill across horizons.

18:59, but for $h = 61$ min from 7:00 to 19:59.

5.4.3 Impact of location on the forecast skill

Tables 5.2, 5.3, and 5.4 summarize the skill scored by each model. However, they do not provide spatial granularity at the sensor level. We obtained boxplots of each model to understand how the horizon and location affect the skill. Figure 5.6 shows

Table 5.4: Skill of the ST-SIF model when forecasting standardized GHI.

Variable	$n_r \times n_c$	n_x [min]	Median skill [%] across $\mathcal{S}_v$ 1 min	11 min	31 min	61 min	Mean [%]
Standardized GHI	8×8	10	31.6	19.6	19.5	19.6	22.6
		20	32.2	20.5	20.4	21.0	23.5
		60	31.4	20.2	20.1	21.2	23.2
	10×10	10	30.1	19.8	20.0	20.2	22.5
		20	31.3	20.3	19.8	19.7	22.8
		60	30.3	19.2	19.3	19.9	22.2

Skill is computed with respect to the persistence model (see Section 5.1.2). The median skill is calculated across sensors for each forecast horizon. The last column is the mean skill across horizons.

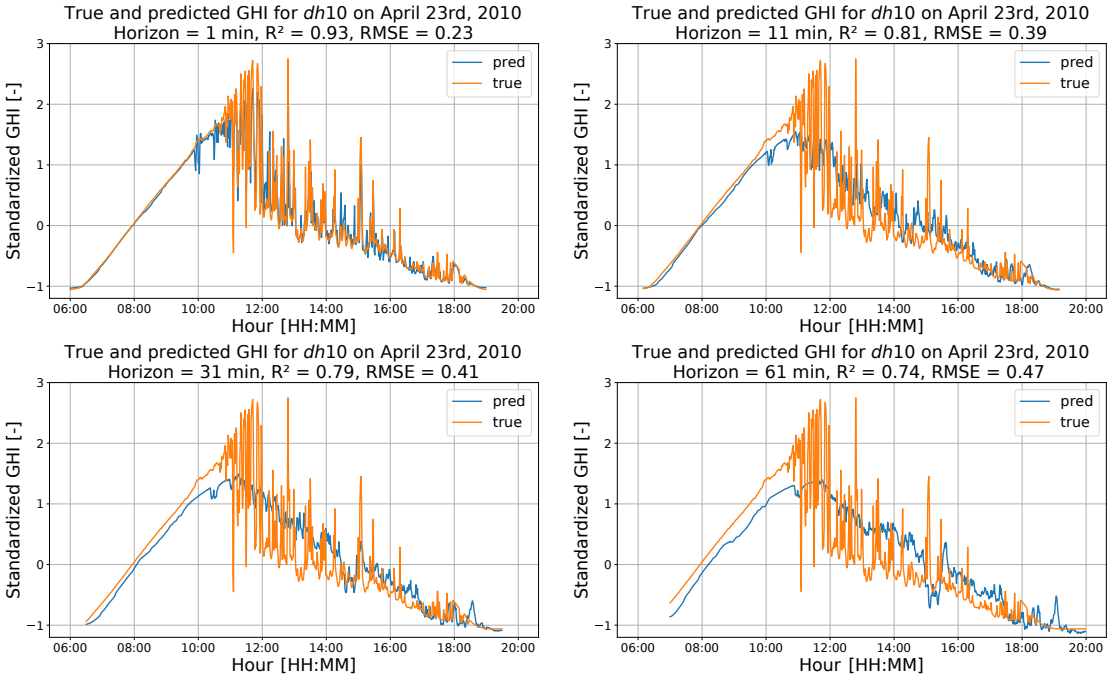
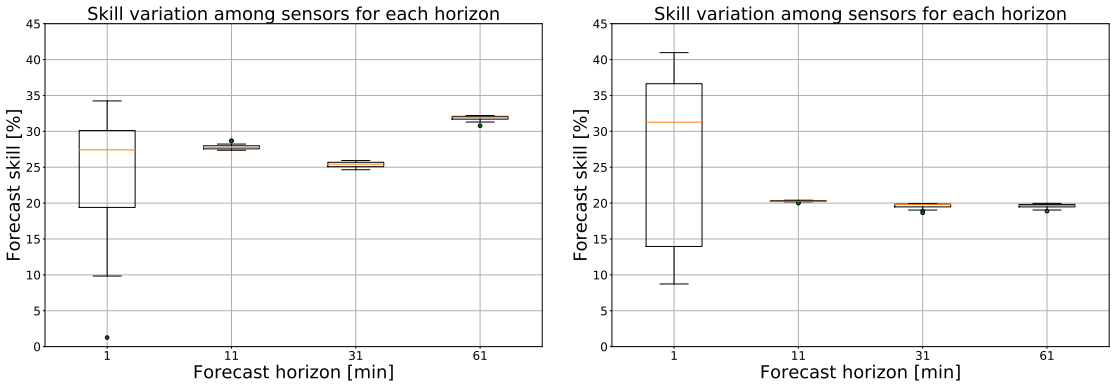


Figure 5.5: True and predicted standardized GHI on sensor *dh10* for the BiLSTM with $n_x = 60$ min. Each plot corresponds to a forecast horizon: 1, 11, 31, and 61 minutes.

two of them (for the rest, see Prado-Rujas et al., 2020): Figure 5.6a corresponds to the baseline LSTM with $n_x = 60$ min, and Figure 5.6b to the ST-SIF with $n_x = 10$ min and grid shape 10×10. These Figures show that skills are very stable for horizons of 11, 31, and 61 minutes. However, there is high variability for 1-minute ahead forecasts, which we investigate in the following.

Inspired by the boxplots and Amaro e Silva and Brito (2018), we analyzed the impact of location on the forecast skill. Figure 5.7 presents the spatial distribution of the skill per horizon for one of the baselines. Specifically, the figure corresponds



(a) LSTM that forecasts CSI with $n_x = 60$ min. **(b)** ST-SIF that forecasts standardized GHI with $n_x = 20$ min and grid shape 10×10.

Figure 5.6: Boxplots showing skill variation among sensors for each forecast horizon.

to the BiLSTM for CSI with $n_x = 60$ min. We obtained the skill contours by nearest-neighbor interpolation. Whereas skill figures are uniform for horizons of 11, 31, and 61 minutes, for 1 minute, it draws significant variability. The grid of sensors lies in an area with predominant winds from the northeast, as described in Section 5.2.1. Therefore, new clouds pushed by the wind tend to quickly cover the sensors from the north and east sides before the rest. This behavior translates into poorer skills for sensors on the north and east sides but benefits those on the west and south sides for short horizons. Specifically, while *ap6* and *ap7* achieve skills of 4.6 % and 2.4 % respectively, the best performers are *dh09* and *dh10* with skills around 32 %, which lie on the west side. This spatial correlation is not present on further horizons, where skill variation is usually below 2 %. These results are consistent with the findings of Amaro e Silva and Brito (2018): further horizons lead to less skill variability among sensors. Furthermore, skills from further forecast horizons are less affected by neighboring sensors, and sensors lying on the west score better skills. The conclusions are similar in the other models, gathered in our repository (Prado-Rujas et al., 2020).

5.4.4 Flexibility and robustness

In a real-world environment, such as a PV plant, a continuous flow of measurements is streamed into a central computing facility. Then, the data is processed, and the model makes the inferences based on them. It is foreseeable that at some stage, PV operators will want to introduce new sensors or remove existing ones. A flexible model can absorb these changes without alterations in its structure, as described in Section 3.3.1. Similarly, some sensors may fail at some point, run out of battery, or, in general, stop working temporarily. In this situation, it is essential to have a system that can recover and continue with the inference. We can expect a temporary decline in prediction quality, but the system will still function until the issue is resolved. This scenario corresponds to the robustness NFF described in

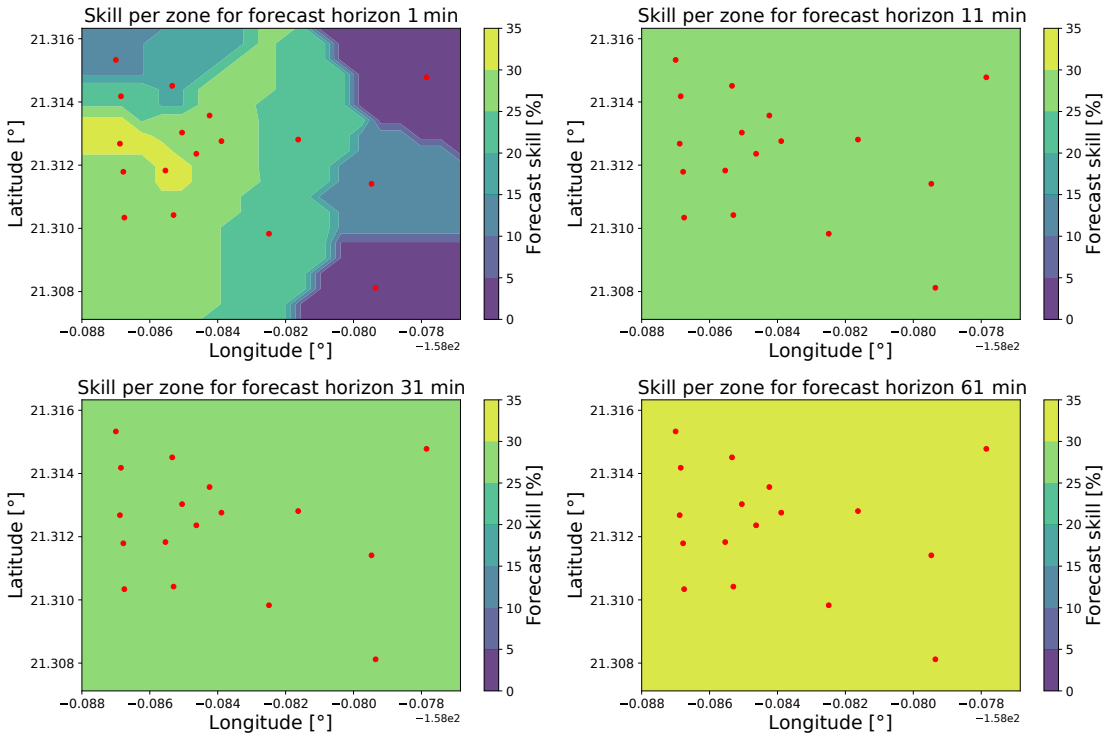


Figure 5.7: Impact of location and horizon on the forecast skill. Skill values correspond to the baseline BiLSTM for CSI with $n_x = 60$ min, obtained by nearest-neighbor interpolation. Each graph corresponds to a different horizon: 1, 11, 31, and 61 minutes. Red dots show the sensor locations.

Section 3.3.2.

If a model can recover missing observations, we can simulate this situation and evaluate the model's robustness. Starting with the tabular standardized solar irradiance dataset, we can randomly remove measurements and analyze the error of the model. Section 3.3.2 describes the general steps to evaluate robustness. Algorithm 1 walks through the process adapted to the solar irradiance use case.

In the following, we explain the robustness test in detail. Firstly, we fix the maximum number of excluded sensors to $n_s - 1 = 16$ in line 1. Then, we set the number of random repetitions for each case in line 2. Therefore, we create a table of $\max_s \cdot$ entries to evaluate the error (line 3). Afterward, we load the model to test, the mesh-grid shape, and the months that belong to the test set. The remaining algorithm is run for ten repetitions for each $s \in \{1, \ldots, \max_s\}$. We randomly pick s sensors from the n_s available ones in line 9. We set the sum of squared errors and the number of predictions to 0 in lines 10 and 11, respectively. We then load batches of sensor observations for each test month and filter the excluded ones in line 15. This step yields the set of uncorrupted observations X. We apply nearest-neighbor interpolation to X, obtaining the mesh-grid $\mathcal{M}$ in line 16. Next,

Algorithm 1: Robustness test procedure.

1: $\max_s \leftarrow n_s - 1$
2: reps $\leftarrow 10$
3: skills $\leftarrow$ table($\max_s \times$ reps)
4: model $\leftarrow$ load_model()
5: grid_shape $\leftarrow$ get_grid_shape(model)
6: months $\leftarrow$ get_test_months()
7: **for** $s \in \{1, \ldots, \max_s\}$ **do**
8: **for** $r \in \{1, \ldots, \text{reps}\}$ **do**
9: excluded $\leftarrow$ random_choice(s, $\{1, \ldots, 17\}$)
10: sse $\leftarrow 0$
11: $n_{\text{preds}} \leftarrow 0$
12: **for** month $\in$ months **do**
13: batches $\leftarrow$ load_batches(month)
14: **for** batch $\in$ batches **do**
15: $X \leftarrow$ filter(batch, excluded)
16: $\mathcal{M} \leftarrow$ interpolate(X, grid_shape)
17: $p \leftarrow$ predict_and_deinterpolate(model, $\mathcal{M}$)
18: $y \leftarrow$ get_ground_truth(batch)
19: sse $\leftarrow$ sse $+ \sum_{\text{batch}} (y - p)^2$
20: $n_{\text{preds}} \leftarrow n_{\text{preds}} + |\,\text{batch}\,|$
21: **end for**
22: **end for**
23: rmse $\leftarrow \sqrt{\frac{1}{n_{\text{preds}}} \cdot \text{sse}}$
24: skills $[s, r] \leftarrow$ compute_skill(rmse)
25: **end for**
26: **end for**

line 17 runs the model and reverses the interpolation, obtaining the prediction p. We then accumulate the squared error based on the ground truth y and the number of predictions n_{preds}. Once all test months are exhausted, Eq. 4.5 yields the RMSE of the simulation. Finally, we compute the skill (Eq. 4.9) in line 24 and store it in the s-th row and r-th column of the skills table.

We performed this robustness test for every model listed in Table 5.4. The test results in $\max_s \cdot r \cdot n_y$ skills, representing the number of excluded sensors, times repetitions, and times horizons. We averaged across repetitions to analyze the skill variation per number of excluded sensors and horizon. Figure 5.8 presents the results for the ST-SIF with $n_x = 60$ min and grid shape 10×10. The figure compares its RMSE when an increasing number of sensors fail to the one scored when all pyranometers function. RMSE worsens below 10 % when almost 25 % of them stop working, and less than 25 % when half of them fail. Figure 5.8 also shows that the three further horizons present more stable skills than those of

1-minute ahead forecasts. Reading the figure from right to left: skill improves as the mesh-grid incorporates more sensors. Therefore, flexibility in the number and distribution of the sensors is also accomplished. The remaining robustness tests derive similar results and are gathered in our repository (Prado-Rujas et al., 2020).

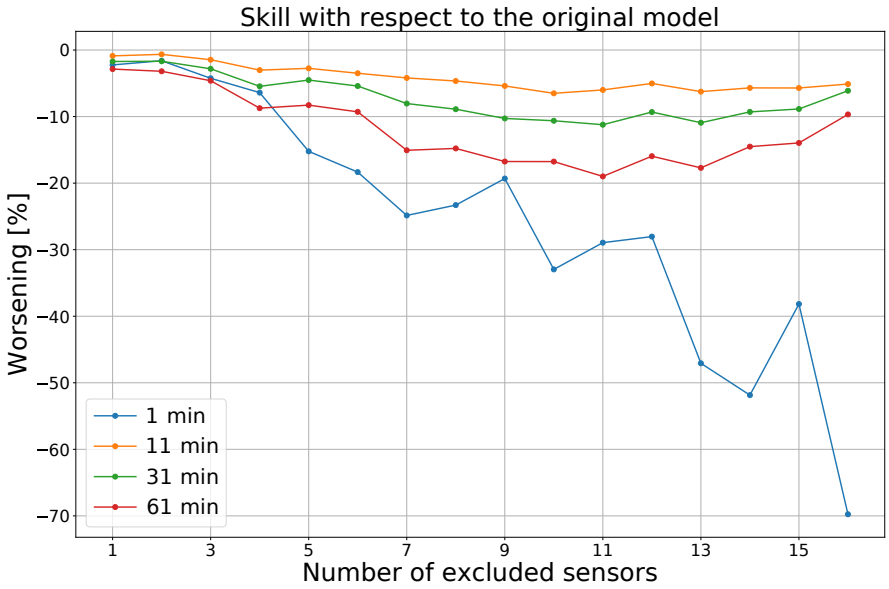


Figure 5.8: Robustness test results for the ST-SIF with $n_x = 60$ min and grid shape 10×10. The plot shows how error degrades when more sensors fail, on every horizon. Each dot corresponds to the average across 10 repetitions with different random sensors excluded.

5.4.5 Integration into a simulation environment

Up to now, we have presented experiments that consume static, historical data in a local computing setup. Yet, we aim to investigate the potential of the ST-SIF in a real-world environment. To bridge this gap, we developed Cloud-based Analysis and Integration for Data Efficiency (CAIDE) (Risco-Martín et al., 2024), a framework for real-time monitoring, management, and forecasting of solar irradiance sensor farms.

CAIDE integrates Model-Based Systems Engineering (MBSE) and Discrete Event System Specification (DEVS) to ensure scalability. This approach builds upon previous studies about the xDEVS formalism (Risco-Martín et al., 2023) and IoT-based solar irradiance sensors (Regueiro Tuñón, 2021). A key advantage of using MBSE and DEVS is that the simulator can progressively become the real control environment of the PV plant. This enables the incremental migration from virtual components to real ones, facilitating conceptualization, design, modeling, and deployment on a unified development framework. As a result, costs are minimized, and confidence in the final system is maximized.

The aim of CAIDE is to achieve a simulation environment capable of processing solar irradiance measurements from farms while integrating additional services, such as forecasting. Following IoT principles, sensors collect solar irradiance observations at the edge layer that are subsequently sent to the server at the fog layer. At this level, PV specialists can analyze those observations and obtain predictions. The cloud layer sustains training tasks of the predictive models and provides authorities with visual reports for decision-making. Furthermore, externalizing the training process aids the scalability of the CAIDE framework when handling larger datasets and models.

For the experiments, we employed the 17 sensors from Oahu (Sengupta & Andreas, 2010) and 18 additional synthetic sensors located in Almería, Spain, generated with PVGIS (Joint Research Centre, 2024). The ST-SIF model predicts GHI using $n_x = 10$ minutes and a grid shape of $(n_r, n_c) = (10, 10)$, for the same forecast horizons (1, 11, 31, and 61 minutes). We conducted all experiments on a Google Cloud Virtual Machine equipped with an NVIDIA Tesla T4 GPU.

To validate the functionality of the CAIDE framework, we performed the following experiments:

1. **Monitoring results**: First, we verify that CAIDE can collect and store sensor observations from both farms.

2. **Outliers detection service**: This service uses the Prophet model (Taylor & Letham, 2017) to detect outliers. Then, those observations are corrected using linear interpolation. As expected, Oahu's data presented significantly more outliers than Almería's, given the smoother trends of the synthetic dataset.

3. **Training service**: This feature allows the training of a solar irradiance model either from scratch or by tuning a pre-trained model with short recent observations. We experimented with the pre-trained ST-SIF model developed in Section 5.4.2 using training periods of 5, 10, 15, 20, and 25 days. The Oahu-based model presented higher errors due to real-world variability, and training time increased linearly with the training set size.

4. **Inference service**: The service launches an inference at a specified timestamp and duration for all forecast horizons. This execution provides estimates of future energy generation, providing valuable information for PV operators.

5. **Reports generation**: This system produces a graphical report, including historical time series plots, skill heatmaps, and outlier detection results. Furthermore, the report can be customized based on the user's needs.

These experiments illustrate the capabilities of CAIDE as a valuable tool for PV plant management and highlight the possibilities of the ST-SIF in real-world

applications. The code used to conduct the experiments is available online (Risco-Martin et al., 2024).

5.5 Summary and discussion of results

The proposed ST-SIF model for solar irradiance forecasting provides some NFFs without neglecting error metrics. Experiments show that our DL-based approach scores similar skills to the four studied baselines for several horizons. In addition, it integrates observations of sensors spread across a ROI into the input and output through mesh-grids. This feature allows the addition or removal of sensors without altering the ST-SIF architecture (flexibility). Furthermore, the error stays within manageable limits when simulating sensor failure (robustness). In all of our experiments, we tested different parameters and models. Even if we present a subset of these experiments in this chapter, all of them are gathered in our repository (Prado-Rujas et al., 2020).

Initially, Section 5.3.1 presents four baseline NNs: FC, RNN, LSTM, and BiLSTM. Then, we describe the proposed ST-SIF in Section 5.3.2, which works with mesh-grids series instead of sensor time series. We measure the models' error with the forecast skill, i.e., the percentual improvement of the RMSE with respect to the persistence model. We obtain the skill for every forecast horizon and sensor, with varying n_x, target variables, and mesh-grid shapes.

The baseline and ST-SIF models reach skills of 44.5 % and 41 % by specific sensors and horizons, respectively. In Section 5.4.2, we aggregate skills calculating the median across sensors for every horizon. The median skill moves between 14.4 and 35.5 % for the baselines and 19.2 and 32.2 % for the ST-SIF. Among the baselines, LSTMs and BiLSTMs tend to perform better, with a slight advantage for the latter. Remarkably, RNNs outperform the other three architectures for 1-minute ahead forecasts of GHI (see Table 5.3). As for the ST-SIF, a coarser grid shape of 8×8 provides slightly better results than for 10×10 (see Table 5.4).

Tables 5.2 and 5.3 indicate that the bigger n_x, the higher the skill. This behavior is evident on the further forecast horizons, which is something we can expect. Namely, more information from the past enhances predictions for the future. Arguably, the skill improves when $n_x \geq h$ for an h-minute ahead forecast. Tables 5.2 and 5.3 also show that using CSI as the target variable provides higher skills in general. All baselines obtain better skills for CSI than for standardized GHI on the three further horizons, except for the RNNs. However, the opposite happens for the 1-minute forecast. This event is likely due to the smoothing effect inherent to CSI, as Section 5.2.4 explains.

Section 5.4.3 presents skill boxplots, which show high variability in the skill distribution for the first horizon (see Figure 5.6). This situation occurs for the

baselines and ST-SIF models, which led us to analyze the correlation between skill and sensor location. Figure 5.7 (top-left plot) shows how skill distributes spatially for 1-minute ahead forecasts when using one of the baselines. The dominant winds from the north and east likely produce this behavior. Therefore, sensor operators must consider this fact when deciding their placement. Furthermore, laying peripheral sensors at carefully selected locations can enhance predictions from the other sensors. In our use case, sensors *ap06* and *ap07* likely improve forecasts for the rest.

One of the main outcomes of this chapter is the achievement of flexibility and robustness NFFs by the ST-SIF. These features arise from harnessing spatial information, which the baselines lack. In Section 5.4.4, we simulate sensor failure and compare the error to the original one using the skill criterion. Figure 5.8 indicates that RMSE stays within manageable limits, particularly on the further horizons. The figure shows that RMSE stays below 15 % of the original error when excluding observations from up to 5 of the 17 sensors. Skill stabilizes at around 6 % for $h = 11$ min. Besides, the decline halts when excluding more than 11 sensors for 31 and 61-minute forecasts. This could be because further horizons are more complex, making the forecast less precise. The experiments also show that we can easily vary the number and distribution of sensors without redesigning the model. As discussed in Section 5.2.5, the model requires that newly installed sensors are located within the spatial coverage $\mathcal{A}$. Furthermore, Section 5.4.5 shows the real-world applicability of the ST-SIF model through the CAIDE framework.

The ST-SIF model provides GHI estimations solely based on historical values of the same variable. However, given the correlation between solar irradiance and weather variables such as cloud coverage and wind, it is evident that the model could benefit from incorporating exogenous information. This limitation of the first use case is addressed in the next chapter, where weather and seasonal data are integrated through the use of exogenous datasets. Another limitation is the relatively small size of the ROI and the limited number of sensors. This issue is also tackled in Chapter 6, which explores a case that consists of a larger sensor network. Additionally, a model that combines multiple solar variables, such as DHI and DNI, would be of interest. We refer to this NFF as extensibility, which is partly explored in Chapter 6 and in greater detail in Chapter 7. Overall, this chapter focuses on two NFFs that are particularly relevant to the solar irradiance forecasting case: flexibility and robustness. These are especially important due to the dynamic nature of the set of solar sensors and the variable availability of their observations.

CHAPTER 6

Second use case: Mobility demand

The second use case in which we focus is mobility demand forecasting. Now, we model two variables and introduce exogenous data. In addition to error metrics, flexibility, and robustness, we study extensibility and integrability NFFs. This chapter is based on our work (Prado-Rujas et al., 2023a). We have made available the supplementary code (Prado-Rujas, Serrano, et al., 2021c) and datasets (Prado-Rujas, Serrano, et al., 2021a) for reproducibility.

This chapter is organized as follows. Section 6.1 introduces the topic and our contributions. Afterward, Section 6.2 applies our 7-phase methodology (described in Section 4.1) to mobility demand sensor observations. Section 6.3 details the baselines and our proposal, the Spatio-Temporal Mobility Demand Forecaster (ST-MDF). We present our experiments in Section 6.4. Finally, Section 6.5 discuss the results of the experiments.

6.1 Introduction and background

Commuting is one of the most time-consuming tasks of modern life, loathed by most people. Traffic jams have become humdrum, and the environment has long begged for a change. Efficient mobility and sustainable development of cities require analyzing and modeling public transport demand.

To this end, the first step is to learn about the commuting habits of citizens. Fiorello and Zani (2015) report that the car is the most widely used transport mode for frequent trips in the EU. Additionally, they report a low average occupancy rate in their work: 1.7 persons per car. These two facts bolster another critical element of mobility: traffic congestion. Vlahogianni et al. (2014) discuss related challenges such as the choice of data resolution and the identification of spatial and temporal flow patterns. In their comprehensive review, they also emphasize the role of AI as a flexible tool for the development of transportation applications

(e.g., Cui et al., 2020; Yao et al., 2018; Zhang, Zheng, et al., 2016). However, to capture the complete picture, we must ask: how can we meet the global mobility demand? To begin with, we have to collect and fuse urban data from individuals to later analyze and model it. However, this is a demanding task, as Liu, Li, et al. (2020) discuss. One way to bridge this gap is by harnessing open datasets that collect taxi, FHV, and public bike trip.

In this chapter, we approach the problem of forecasting spatio-temporal mobility demand, as described in Section 3.2. As in Chapter 5, we apply the methodology described in Chapter 4. As a first step, we analyze real-world open datasets of different mobility services. We focus on two variables: taxi trips and bike rides from a populous city. Afterward, we define the mobility demand mesh-grid based on time, location, and trip counts. This data structure captures the spatial correlations between city zones and mobility services (i.e., variables). The grid can be seen as a snapshot of mobility demand as time evolves, much like the frames of a video. After that, we develop the ST-MDF to predict mobility demand based on the defined mesh-grid. The main module of the model is based on the spatio-temporal one described in Section 4.2.1. Additionally, it includes two exogenous modules that extract seasonal and weather features (see Section 4.2.2). Afterward, we compare the error of the ST-MDF model to several baselines and proposals from the literature. Furthermore, we analyze how the different modules of our model contribute towards the final prediction. As we will discuss in the chapter, the presented ST-MDF model offers extensibility and integrability NFFs, as well as flexibility and robustness.

The contributions of this chapter are as follows:

1. Definition of a global mobility mesh-grid related to a certain ROI. Thanks to spatial information, this structure integrates heterogeneous mobility data, such as taxi trips and bike rides. Furthermore, extending it with additional data sources, such as scooters, metro, buses, etc., is possible.

2. Design, training, and evaluation of the ST-MDF model. The proposal harnesses mobility demand, weather, and periodical data as input. Moreover, the model provides the forecasts for several horizons at once.

3. Flexibility and robustness regarding the number and location of data sources (bike racks and taxi zones). These NFFs are fundamental for models that deal with components that vary over time.

4. Better error metrics than the baselines and related models from the literature.

6.2 Data analysis and transformation

Similarly to Chapter 5, we now apply the 7-phase data analysis and transformation methodology defined in Section 4.1. This time, we study mobility demand in the city of Chicago. Specifically, we consider taxi trips and bicycle rides obtained from open data portals. Additionally, we incorporate weather data from https://www.meteoblue.com/historyplus and seasonal information extracted from the timestamp.

6.2.1 Phase 1: Exploratory Data Analysis

Chicago's City Council data portal gathers numerous open datasets related to the city. Among them, we focus on the taxi trips (City of Chicago, 2024) based on the city census tracts. In addition, we harvest data from Divvy's data portal about bicycle rides (Lyft, 2024b). In this case, observations' geolocations rely on docking stations where users pick and drop the bikes. In the following, we discuss the EDA aspects that we laid on Section 4.1.1:

1. Sensors

The set $\mathcal{S}$ consists of 801 sensors for taxi trips and 684 more for bicycles. We consider one additional sensor for weather data acquisition. Figure 6.1b depicts the distribution of all taxi, bicycle, and weather sensors across Chicago.

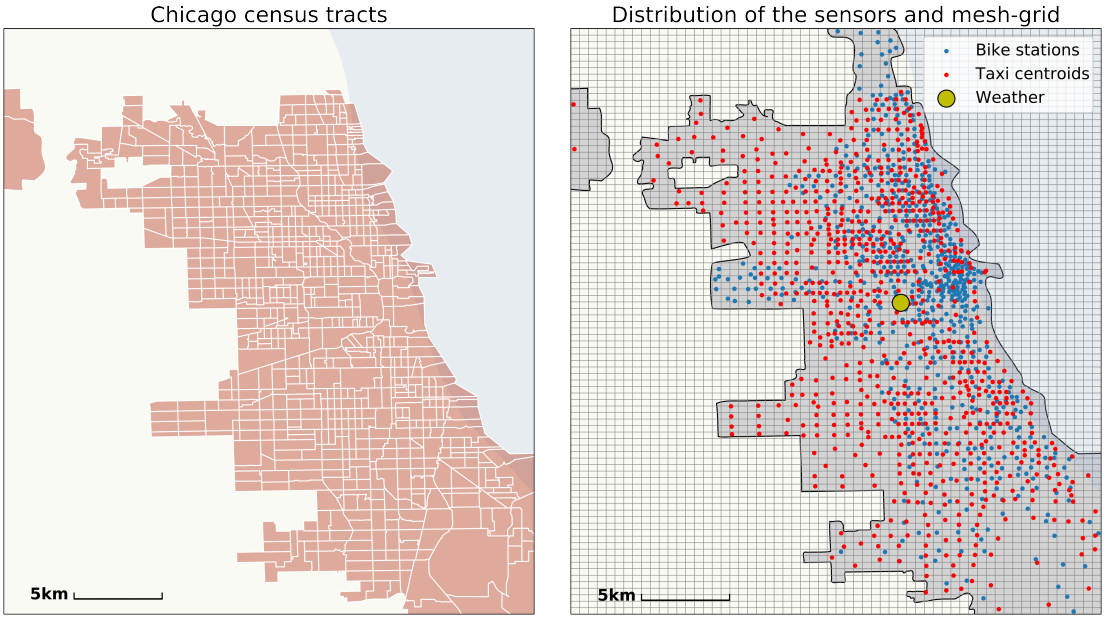
2. Variables

We forecast two variables regarding mobility: the number of taxi pick-ups and the number of bicycle rides, i.e., $n_v = 2$. Unlike in the previous use case, these are discrete variables since they can only take a countable set of values. The raw data considers several others, such as trip termination, length, and cost. However, we restrict ourselves to trip starts because they pinpoint mobility hotspots.

As for the weather variables, we consider the following ones: temperature, precipitation, snowfall, relative humidity, wind speed and direction, cloud cover, and solar irradiance. We also account for periodical aspects that affect mobility. These comprise the time of the day, time of the week, time of the year, weekday or weekend, and holiday or working day.

3. Variable units

The units for taxi and bike rides are counts. We will present the units from the weather variables at the end of this section. Periodical variables are measured in seconds (except weekdays and holidays, which are boolean).



(a) Chicago census tracts.

(b) Sensors and grid of shape 90×60.

Figure 6.1: Distribution of the sensors over Chicago's metropolitan area, USA. The left figure depicts the census tracts, whose centroids define the location of taxi zones. The right figure shows the location of the bicycle stations, taxi centroids, and weather station overlayed with a mesh-grid.

4. Variables observed by every sensor

In this use case, each mobility sensor observes just one variable: taxis or bicycles (see Figure 6.1b). A single sensor collects all weather variables. Periodical ones do not rely on sensors (they are inherent to the timestamps).

5. Geolocation of every sensor

Taxi pick-ups are monitored within each census tract of Chicago, which are depicted in Figure 6.1a. These census tracts have remained mostly unchanged over the past century. We consider the centroids of the census tracts as the taxi geolocations, which are shown in Figure 6.1b. The exact coordinates of these centroids are available in (Prado-Rujas, Serrano, et al., 2021a).

Bicycle rides, on the other hand, commence at docking stations. Determining sensor locations is more complex in this case due to variations in how Divvy recorded them over time. We followed the procedure below to establish the set of bike sensor locations:

- **2013 to 2017**: Lookup files are provided for each raw dataset, which can span different periods (e.g., yearly, monthly, quarterly). By combining all these lookup files, we came up with 586 unique sensor identifiers, though

some were repeated with a slightly different geolocation. In such cases, we computed the average latitude and longitude to obtain unique geolocations for these identifiers.

- **2020**: In July 2020, Divvy introduced 3500 dockless bikes[1]. Therefore, trip records for such bikes directly include latitude and longitude. For docking bikes, we applied the same process as before: extracting unique sensor identifiers and computing average geolocations for repeated ones. We then combined these sensors with the previous ones, prioritizing the ones derived from lookup files. This merge process yields 684 unique bike sensors.
- **2018 and 2019**: No additional lookup files or geolocations were provided in the trip records, only sensor identifiers. Therefore, we rely on the 684 sensors identified beforehand, ensuring all sensor identifiers from these years were covered by the previously calculated set. Only six identifiers were missing, but these were found to correspond to repair or test events conducted by Divvy.

By combining these pieces, we identified the 684 unique bike sensors shown in Figure 6.1b. As before, their specific geolocations are available in (Prado-Rujas, Serrano, et al., 2021a).

Finally, we chose the geolocation of the weather sensor as the centroid of all taxi and bicycle sensors.

6. Spatial coverage

The spatial coverage comprises the city of Chicago and its metropolitan area, which is approximately 47 by 31 km (see Figure 6.1).

7. Temporal granularity

- Taxi trips: $\tau = 15$ min.
- Bike rides: $\tau = 1$ s.
- Weather: $\tau = 1$ h.

8. Temporal coverage

We consider observations from January 2013 to December 2020. However, we decided to trim from mid-March 2020 onward due to the impact of the COVID-19 outbreak, covering over 86 months in the experiments. The pandemic radically impacted mobility patterns, making the data from mid-March to December 2020 atypical for our analysis.

[1] https://divvybikes.com/explore-chicago/expansion-temp (last accessed: 2025/03/15)

9. Missing observations

- Taxi trips: The original dataset consisted of 195 million trips, from which we removed about 11.9 % because the pick-up zone was missing. This leaves almost 172 million taxi trips.
- Bike rides: We discarded about 0.35 % of the trips because their longitude or latitude was missing. According to Lyft (2024b), they additionally eliminate trips lasting one minute or less and trips taken by their staff. The total number of bike rides used is over 24 M.
- Weather: There were no empty records.

10. Main statistics

Table 6.1 summarizes some statistics of the raw mobility demand with $\tau = 15$ min. The upper part corresponds to taxi trips, and the lower to bike rides. The table includes the 12 sensors with the highest volume of trips in both cases. It additionally shows statistics for all taxi and bike sensors aggregated. Table 6.1 reveals high variability and sparseness. The number of null observations is partly explained by periods of low activity (e.g., nighttime) and low-incidence regions. For further details, Prado-Rujas, Serrano, et al. (2021c) gathers the complete tables.

As for the weather, Table 6.2 presents the main statistics of the eight variables measured by the weather sensor from Chicago. In this case, temporal granularity is one hour. Angles are generally not a reliable input for DNNs because certain values (e.g., 0° and 359°) should be numerically close but are represented as significantly different. To address this, we converted wind speed and direction into a wind vector using the following equations:

$$\begin{aligned} \text{Wind-x} &= v \cdot \cos\left(\delta \cdot \pi/180\right) \\ \text{Wind-y} &= v \cdot \sin\left(\delta \cdot \pi/180\right), \end{aligned} \tag{6.1}$$

where v represents the wind speed in km/h and δ denotes the wind direction in degrees. The resulting wind vector is then converted into meters per second (m/s). Figure 6.2 illustrates a two-dimensional histogram of the wind vector observations.

6.2.2 Phase 2: Temporal resolution

In Section 6.2.1, we identified three temporal granularities in the raw data: 1 s, 15 min, and 1 h. Using one second is too fine for mobility demand forecasting, and within one hour, it can vary greatly. Therefore, we choose 15 min as a middle ground, as suggested in the literature (Guo et al., 2014). To accomplish this, we need to convert the bike rides and weather observations as described in Section 4.1.2. First, we bin time into 15-minute intervals and accumulate bike ride counts. Regarding weather, we forward-fill the three missing observations every hour, which is equivalent to a nearest-neighbor interpolation.

Table 6.1: Summary of the top-12 sensors (volume-wise) for taxi trips and bike rides (January 2013 – December 2020 with $\tau = 15$ min).

$\mathcal{V}$	Sensor	null	mean	std	min	25 %	50 %	75 %	95 %	max
Taxi trips	*812.01*	21.3	14.8	13.6	0	2	13	24	40	94
	818	19.6	15.5	14.7	0	3	12	25	43	133
	814.03	20.0	17.9	15.6	0	3	16	29	46	116
	622	3.3	19.9	13.0	0	10	19	28	43	129
	2819	24.3	23.0	25.3	0	1	14	37	74	184
	817	15.7	25.3	28.0	0	4	18	36	79	395
	815	14.4	28.5	24.2	0	5	26	46	71	199
	3204	3.0	28.5	22.2	0	9	25	44	69	219
	810	1.7	31.0	14.3	0	22	31	40	55	136
	3201	14.0	39.7	35.8	0	5	33	68	103	211
	9800	5.6	42.4	36.5	0	9	35	69	110	222
	8391	13.8	69.7	79.2	0	5	35	118	233	418
	all	93.1	0.8	6.0	0	0	0	0	2	418
Bicycle rides	*43*	68.4	0.8	2.1	0	0	0	1	4	48
	174	69.4	0.8	2.0	0	0	0	1	5	31
	195	64.8	0.9	1.8	0	0	0	1	4	25
	90	69.0	1.0	2.1	0	0	0	1	6	28
	85	68.9	1.0	2.2	0	0	0	1	5	43
	268	73.1	1.0	2.7	0	0	0	1	6	59
	77	63.1	1.0	2.3	0	0	0	1	5	34
	177	70.1	1.1	2.7	0	0	0	1	6	44
	192	67.4	1.1	2.8	0	0	0	1	7	47
	91	67.8	1.2	2.8	0	0	0	1	7	38
	76	69.7	1.3	2.9	0	0	0	1	7	41
	35	64.5	2.0	4.3	0	0	0	2	11	51
	all	91.7	0.1	0.7	0	0	0	0	1	59

Abbreviations are as follows. Null: percentage of null observations, min: minimum, max: maximum, std: standard deviation, X %: X^{th} percentile. Except for null, units are counts.

Table 6.2: Summary of the variables measured by the weather sensor from Chicago, USA (January 2013 – December 2020 with $\tau = 1$ h).

Variable	Units	mean	std	min	25 %	50 %	75 %	max
Temperature	°C	10.3	11.6	-34.5	0.9	10.6	20.2	36.1
Precipitation	mm	0.1	0.5	0.0	0.0	0.0	0.0	24.9
Snowfall	cm	0.0	0.0	0.0	0.0	0.0	0.0	2.8
Relative humidity	%	71.3	17.1	11.0	59.0	73.0	85.0	100.0
Wind-x	m/s	-0.2	4.9	-19.7	-3.6	-0.2	3.2	18.2
Wind-y	m/s	-1.1	4.3	-22.4	-3.6	-0.7	1.7	22.5
Cloud cover	%	56.7	44.5	0.0	7.5	74.0	100.0	100.0
Solar irradiance	W/m^2	170.8	241.4	0.0	0.0	12.5	294.6	939.8

Min: minimum, max: maximum, std: standard deviation, X %: X^{th} percentile.

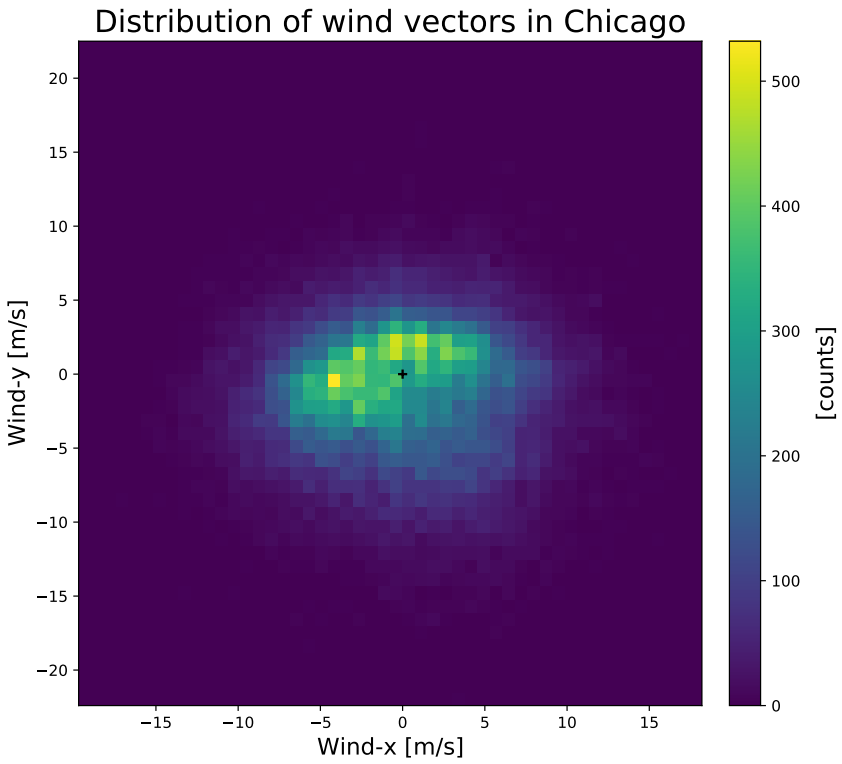


Figure 6.2: Distribution of wind vectors in Chicago (2013 – 2020).

6.2.3 Phase 3: Tabular dataset

At this stage, we have completed the EDA phase and converted the observations into a standard temporal resolution. Next, we gather all sensor observations into a table as described in Section 4.1.3. Accounting for two leap years, the total number of timestamps is $n_t = 4 \cdot 24$ timestamps/day $\cdot(8 \cdot 365 + 2)$ days $= 280\,512$ timestamps. For each timestamp, we observe 801 taxi trip sensors, resulting in over 224 million individual observations.

For bicycle rides, we also account for dockless bikes, and therefore we cannot rely solely on the 684 sensors identified earlier. Instead, we calculate the number of trips for each grid element and every timestamp. This step uses sensor identifiers for 2013 – 2019 and geolocation data for 2020. Effectively, we "forget" the original set of sensors and treat the grid elements as the new sensors, which turn out to be 857. This results in over 240 million observations, though many are null.

In addition, we prepare a table with weather variables in a similar manner. Unlike for taxis or bikes, we consider a single weather sensor with $\tau = 15$ min. Consequently, the weather table contains the same 280 512 timestamps, and eight weather variables.

6.2.4 Phase 4: Data scaling

As discussed in Section 4.1.4, we scale the sensor observations to ease the forecasting task. We apply min-max normalization described in Eq. 4.1 to obtain values in $[0, 1]$. The minimum value is null, as Table 6.1 reveals. Thus, Eq. 4.1 merely requires dividing by the maximum. Since the mobility sensors record counts, we opted to take the absolute maximum for each mobility service. This aspect differs from the solar irradiance use case from Chapter 5. We considered additional scaling techniques, such as taking the local minimum/maximum or using standardization (Eq. 4.2). Nevertheless, exploratory experiments indicated that models were less precise. In addition, we normalized weather observations by applying Eq. 4.1 on each variable. The code to reproduce these datasets is in the supplementary repository (Prado-Rujas, Serrano, et al., 2021c).

6.2.5 Phase 5: The mesh-grid

In this section, we build the mobility mesh-grid following phase 5 described in Section 4.1.5. First, we define the spatial resolution based on the number of sensors and their geolocations. Afterward, we build the mesh-grids for the spatio-temporal forecasting model.

Phase 5.1: Spatial resolution and grid placement

As discussed in Section 4.1.5, we seek a compromise between the spatial resolution and the number of sensors. First, we define a rectangular spatial coverage that covers all sensors using their minimum and maximum latitudes and longitudes. We add an offset $\varepsilon = 0.002^{\text{o}}$ to them, which yields an area of approximately 47 by 31 km. Then, we define $(n_r, n_c) = (90, 60)$ to obtain $\sigma \approx (500, 500)$ meters. Thus, the ratio between sensors and grid elements is about 15 % for both variables. Figure 6.1b depicts Chicago overlayed with the defined mesh-grid.

Phase 5.2: From sensors to mesh-grids

This section describes how we build the mobility mesh-grids. The first point to note is that the studied variables are discrete, as discussed in Section 6.2.1. Intuitively, the number of taxi trips at a specific timestamp does not imply a similar count at the next one. Similarly, the number of bike rides along Chicago's promenade plummets as one moves into the lake.

Given the sparseness of the observations (see Table 6.1), we choose to interpolate when building the mesh-grids. Specifically, we apply two-dimensional linear interpolation for both variables at each timestamp, independently. The inputs for interpolation are the sensor observations, and the target interpolation points are the grid elements. This method can only estimate points within the convex hull of $\mathcal{G}$, as discussed in Section 4.1.5. To bypass this, the algorithm assigns a value of zero to the points outside the convex hull, which mainly lie within the

lake or outside the city (see Figure 6.1b). Consequently, the components of the interpolated mesh-grid should not be interpreted as trip counts in these regions for the given timestamp. This is not only due to the interpolation procedure but also because the values are normalized (see Section 6.2.4).

6.2.6 Phase 6: Preservation and metadata

As in the previous use case, we use HDF5 format to store the resulting datasets:

- **Taxi trips counts**: This accounts for eight years of taxi trip observations from the 801 census tracts with $\tau = 15$ min.
- **Normalized taxi trips counts**: This corresponds to the same observations, normalized absolutely, as described in Section 6.2.4.
- **Bike rides counts**: This corresponds to eight years of bike ride observations from the 857 grid elements with $\tau = 15$ min, as described in Section 6.2.3.
- **Normalized bike rides counts**: This corresponds to the same observations, normalized absolutely, as described in Section 6.2.4.
- **Normalized taxi trips mesh-grids of shape** 90×60: This dataset gathers the interpolated grids of shape (90, 60) for taxi trips.
- **Normalized bike rides mesh-grids of shape** 90×60: Similarly, this dataset corresponds to the interpolated grids of shape (90, 60) for bike rides.

All these datasets are publicly available online (Prado-Rujas, Serrano, et al., 2021a). In addition, we store the following metadata: geolocation of the census tracts centroids, geolocation of bike stations, geolocation of grid elements for bikes, temporal resolution, sensor identifiers, etc.

6.2.7 Phase 7: Validation

Similarly to Section 5.2.7, we carried out different validations on every phase of the analysis and transformation process. For reference, these are gathered in Jupyter Notebooks in our repository (Prado-Rujas, Serrano, et al., 2021c), including:

- Plots of the number of trips over time. These show a decay in volume, likely caused by the rise of FHVs. Besides, the volume of trips plummets on mid-March 2020 due to the COVID-19 pandemic.
- Analysis and plots of the sensors' geolocations. They aided us when deciding how to build the mesh-grid.
- Exploration of different scaling techniques, as discussed in Section 6.2.4.
- Animated taxi demand plots, for visualizing their distribution and evolution

over time.

6.3 Forecasting model

In this section, we present the developed mobility demand forecasting models. Initially, Section 6.3.1 introduces two exogenous modules used by different models. Afterward, Section 6.3.2 describes baseline models that do not consider sensor geolocations. Finally, Section 6.3.3 presents our ST-MDF model.

6.3.1 Exogenous modules

As anticipated in Section 6.1, we incorporate two exogenous modules for the studied use case. They are based on Section 4.2.2 and described in the following:

- **Periodical module**: This module captures seasonal patterns that influence transport demand. The input tensor is derived from the most recent timestamp, functioning as a flat-extraction module as outlined in Section 4.2.2. It processes eight variables, encoded as follows:
  - time of the day, transformed using sine and cosine functions to values in the range $[-1, 1]$,
  - time of the week, transformed using sine and cosine functions to values in the range $[-1, 1]$,
  - time of the year, transformed using sine and cosine functions to values in the range $[-1, 1]$,
  - weekday or weekend, encoded as $\{0, 1\}$, and
  - holiday status[2], encoded as $\{0, 1\}$.

  The input tensor is then passed through a dense layer with 16 neurons, producing a periodical feature vector.
- **Weather module**: This module extracts weather-related features, as meteorological conditions have a significant impact on mobility choices. It uses as input the eight weather variables described in Section 6.2, spanning a temporal window of width n_x. Consequently, this module corresponds to the recurrent module, as detailed in Section 4.2.2. As such, the input tensor is processed through an LSTM layer with 16 neurons, yielding a weather feature vector.

[2]Data obtained from https://www.officeholidays.com/countries/usa/illinois/

6.3.2 Baselines

Similarly to Chapter 5, we propose several baseline models. They all work with a temporal input window of mobility observations and provide predictions for several horizons. Furthermore, the models work with the mobility mesh-grids built on Section 6.2. However, they are agnostic to the sensors' geolocations.

- **LSTM**: It comprises three modules: mobility, periodical, and weather. The last two modules are described in Section 6.3.1. The mobility module inputs a mobility mesh-grid of shape (n_x, n_r, n_c, n_v). This mobility tensor combines the taxi and bike grids for the past n_x timestamps. It flattens it on the last three dimensions and passes it through an LSTM of 16 neurons. The output is combined with those of the other two modules. Afterward, it passes by two dense layers of 32 and $n_y \cdot n_r \cdot n_c \cdot n_v$ neurons. The resulting vector is reshaped into the output tensor of shape (n_y, n_r, n_c, n_v). Finally, the opposite interpolation of this tensor yields the (n_y, n_s, n_v) predictions.

- **BiLSTM**: It has the same structure as the baseline LSTM. However, it relies on the BiLSTM instead of the LSTM. This baseline incorporates 8 BiLSTM neurons that return 16 values (8 in the forward pass and 8 more in the backward pass).

- **Persistence**: It behaves as described on Section 5.1.2. It is based on the assumption that the forecasted variable will remain unchanged for the devised forecasted horizon. Hence, we can express it as:

$$f\left(\left\{\, \mathcal{M}_v^t \mid t = t_{\gamma-(n_x-1)}, \ldots, t_\gamma,\, v \in \mathcal{V} \,\right\}\right) \equiv \mathcal{M}_v^{t_\gamma}.$$

- **Naive**: Its prediction is the point-wise mean of the input window, which we can write as:

$$f\left(\left\{\, \mathcal{M}_v^t \mid t = t_{\gamma-(n_x-1)}, \ldots, t_\gamma,\, v \in \mathcal{V} \,\right\}\right) \equiv \frac{1}{n_x} \sum_{i=0}^{n_x-1} \mathcal{M}_v^{t_{\gamma-i}}.$$

6.3.3 ST-MDF

The baseline models could suffice if time series correlations were merely temporal. However, mobility demand also features spatial ones, as discussed in Section 6.2. Therefore, we describe the proposed ST-MDF model, which aims to bridge this gap.

The ST-MDF model has at its core a mobility module based on Section 4.2.1. This module aims at extracting spatio-temporal mobility patterns, carrying these steps:

1. Initially, it takes the last n_x taxi trips and bike ride mesh-grids as input. This tensor has shape (n_x, n_r, n_c, n_v), same as for the baselines.

2. Afterward, it passes them through ConvLSTM and max-pooling layers.
3. The resulting three-dimensional tensor is then flattened into a mobility feature vector.

Afterward, the ST-MDF model proceeds as follows:

1. It concatenates the mobility feature vector with the periodical and weather ones. These are obtained as described in Section 6.3.1.
2. Next, the model applies a dense layer, yielding a one-dimensional vector.
3. This vector is reshaped into a rank-4 tensor to match the forecast horizons, latitude, longitude, and mobility service.
4. The last step is a deconvolution, as described in Section 4.2.1.
5. The final predictions for every horizon, sensor, and service are obtained by reversing the interpolation described in Section 6.2.5.

We consider variations of the ST-MDF model based on the modules that constitute them:

- **ST-MDF**: It only considers the mobility module. It contains 944 466 trainable parameters when configured with $n_x = 4$, $n_r = 90$, $n_c = 60$, $n_y = 4$, and $n_v = 2$.
- **ST-MDF$^{\mathbf{t}}$**: It includes the periodical module besides the mobility one, yielding 963 874 trainable parameters with the same settings as the ST-MDF.
- **ST-MDF$^{\mathbf{w}}$**: Similarly, it considers the mobility and weather modules, resulting in 965 330 trainable parameters for the same configuration as the ST-MDF.
- **ST-MDF$^{\mathbf{*}}$**: It combines all three modules for a total of 984 738 trainable parameters.

Figure 6.3 presents the ST-MDF* model from end-to-end. In our experiments, the mobility module has one ConvLSTM layer with three filters, kernel size of (8, 8), and hyperbolic tangent activation. The max-pooling layer has size (4, 4), and the dense layer has 1204 neurons and linear activation. Finally, the deconvolution has two filters, kernel size of shape $(n_y, 6, 6)$ and strides of shape (2,2,2) with linear activation. We apply Eq. 4.5 on every horizon, sensor, and mobility service to evaluate the error of the models. This approach provides RMSE values that are comparable to the ones obtained by the baseline models (see Section 6.3.2).

Where appropriate, the parameters n_x, l, and n_y will be displayed as a subscript of the model name to simplify the reading. For instance, ST-MDF$^*_{12,4,8}$ refers to

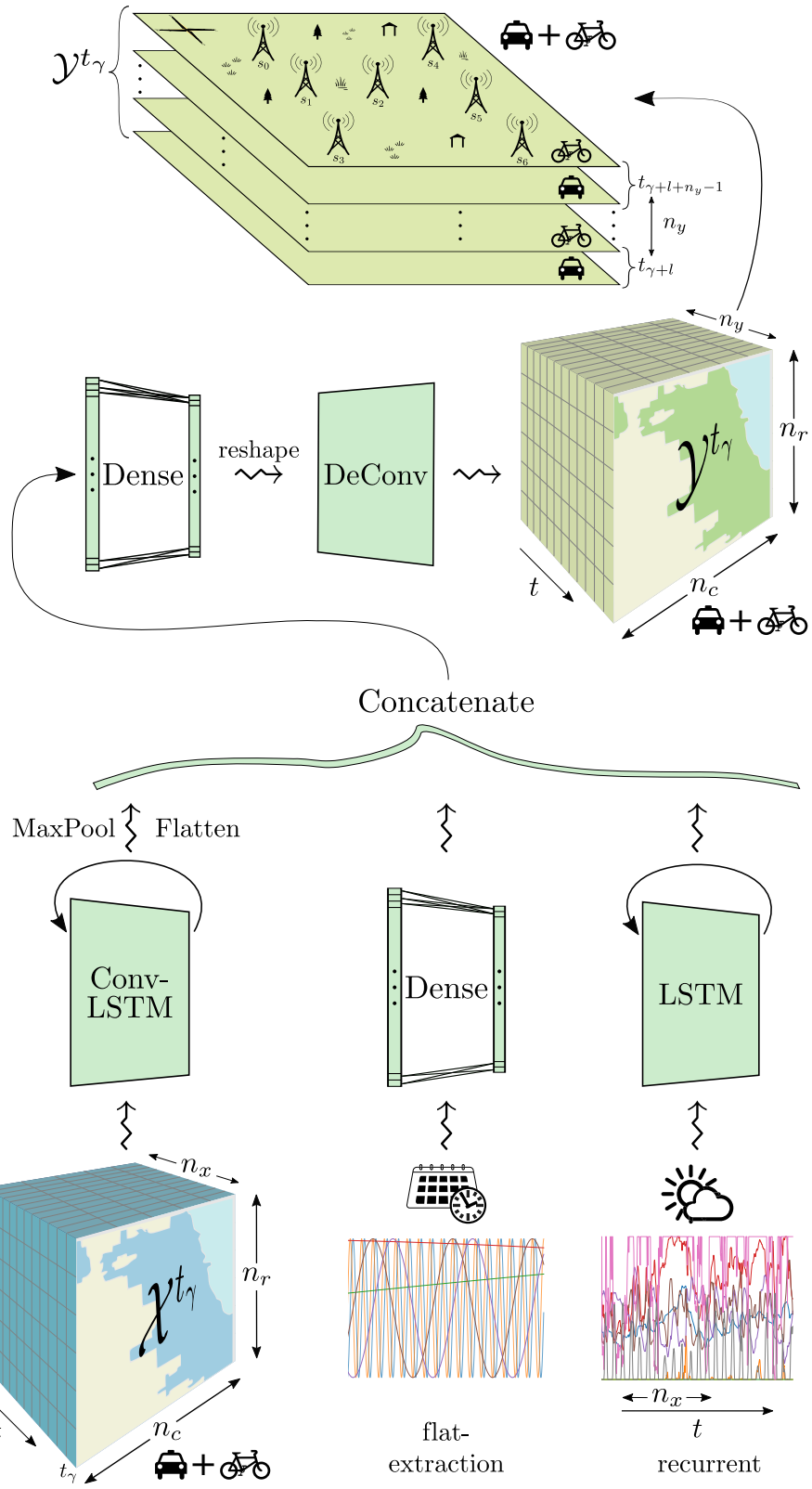


Figure 6.3: Spatio-Temporal Mobility Demand Forecaster (ST-MDF) model.

the ST-MDF* model with $n_x = 12$, $l = 4$, and $n_y = 8$.

6.4 Experiments and evaluation

This section presents the experiments and results regarding the mobility demand use case. As discussed in the previous use case, we prioritize NFFs over achieving low error metrics. Nevertheless, our proposal's error competes against the literature's baselines and models.

Section 6.4.1 initially depicts the parameter choice and explains the experimental setup. Afterward, Section 6.4.2 presents the error scored by the ST-MDF model under different scenarios. It also compares its error metric to those of the baselines and models from the literature. Section 6.4.3 studies how the error behaves when varying the parameter n_x. Then, we study flexibility and robustness in Section 6.4.4. Finally, we investigate extensibility and integrability NFFs in Sections 6.4.5 and 6.4.6, respectively.

6.4.1 Experimental setup

As in the previous use case, we developed the models using the Keras library (Chollet et al., 2015) with the Tensorflow backend (Abadi et al., 2015). We explored eight optimizers (Adadelta, Nadam, SGD, RMSprop, Adagrad, Adamax, Adam, Ftrl) and three loss functions (MSE, MAE and MSLE). We trained the ST-MDF$^*_{4,4,4}$ model during ten epochs to choose the best possible combination. The training times were similar (around 9 minutes per epoch). Regarding the error, the best results were obtained for MSE loss with Adam, Adamax, RMSprop, and Nadam optimizers, with a slight preference for the latter. Therefore, we chose the Nadam algorithm to optimize the MSE loss function.

The parameters we used in the subsequent experiments are as follows:

- **Variables**: Taxi trip and bicycle ride counts, so $n_v = 2$ as discussed in Section 6.1.
- **Temporal granularity**: $\tau = 15$ minutes, as discussed in Section 6.2.2.
- **Models**: The modules and parameters of the baseline and ST-MDF models were already discussed in Sections 6.3.2 and 6.3.3.
- **Number of input timestamps**: $n_x \in \{4, 8, 12, 16, 20\}$. These temporal windows correspond to 1, 2, 3, 4, or 5 hours, given that $\tau = 15$ min.
- **Forecast horizons**: Horizons range from 1 h to 3 h and 45 min depeding on the experiment, with $n_y = 4$ or 8.
- **Number of sensors**: 801 for taxis and 857 for bikes.

- **Mesh-grid shape**: 90×60.
- **Optimizers**: Nadam.
- **Learning rate**: 0.001.
- **Loss function**: MSE.
- **Batch size**: 256.
- **Number of epocs**: 50.

As Section 6.2.1 explains, the temporal coverage ranges between 2013 and 2020. However, we considered only until mid-March 2020 due to the COVID-19 pandemic. After that date, mobility patterns deviate from the usual, as we saw in the EDA phase (see Section 6.2.7). Specifically, we used the period between 2013 and 2017 for training (∼70 %), 2018 for validation (∼14 %), and 2019 and from January to mid-March 2020 for testing (∼16 %). We conducted the experiments using a Ubuntu machine with 12 cores and 24 logical processors. Additionally, the machine was equipped with an NVIDIA GeForce GTX TITAN X GPU. As a reference, training times range from 9 to 34 minutes per epoch, depending on the parameters n_x and n_y.

6.4.2 Error metrics

This section compares the error of the ST-MDF model to that of baselines and models from the literature. Our goal is to develop real-world models that provide NFFs while complying with acceptable error metrics. We use the RMSE metric (Eq. 4.5) to evaluate the error during the testing period. Additionally, we calculated the MAE, which is available in our repository (Prado-Rujas, Serrano, et al., 2021c). Nevertheless, MAE and RMSE offer similar insights.

In Section 4.3.1, we explained that we calculate the error metric on every horizon, sensor, and variable. In this chapter, we only consider the grid elements that contain taxi zone centroids or bike stations in the error estimation. This approach provides a more representative evaluation than if we would consider the error in the entire mesh-grid. Besides, we apply weighting factors based on the volume of mobility demand. We obtain these weights by calculating the mean number of trips per zone over τ, which we normalize using Eq. 4.1. We denote this metric as weighted RMSE (wRMSE) throughout this chapter.

Comparison with baselines

We described the baselines models and their parameters in Section 6.3.2. Note that the persistence and naive models do not have weights that need training. Instead, they forecast mobility demand directly from the input grids. In the experiments, we fixed $l = 4$ (i.e., one hour) and trained all combinations of

$n_x, n_y \in \{4, 8\}$ (i.e., one or two hours). Additionally, we trained the models for the case where n_x, l, and n_y are 8 (two hours).

The ST-MDF and the four baseline models forecast both variables simultaneously. For clarity, we split their evaluation into two tables: Table 6.3 for taxi trips and Table 6.4 for bike rides. Both present the wRMSE on each forecast horizon. We denote each horizon h_i as $h_i = (i + l) \cdot 15$ min, where i takes values from $\{0, ..., n_y - 1\}$. The wRMSE per horizon displayed in the tables is the mean wRMSE across sensors.

Table 6.3: Error metrics of the ST-MDF and baseline models for the taxi trips task.

n_x	l	n_y	**Model**	**Time**	**Mean wRMSE** $[10^{-2}$ **counts]** **across** $\mathcal{S}_v$							
					h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7
4	4	4	ST-MDF*	07:35	14.6	14.7	15.0	15.3				
			LSTM	04:24	22.5	21.9	22.2	21.9				
			BiLSTM	05:29	19.4	19.1	19.0	19.0		-		
			Persistence	-	14.3	15.1	15.8	16.5				
			Naive		15.1	15.8	16.5	17.2				
		8	ST-MDF*	09:40	15.4	15.3	15.3	15.4	15.6	15.8	16.2	16.6
			LSTM	05:44	19.6	19.5	19.5	19.5	19.6	19.3	19.4	19.3
			BiLSTM	07:50	17.5	17.5	17.4	17.3	17.3	17.3	17.4	17.4
			Persistence	-	14.3	15.1	15.8	16.5	17.1	17.8	18.4	19.0
			Naive		15.1	15.8	16.5	17.2	17.8	18.4	18.9	19.5
8	4	4	ST-MDF*	13:05	14.5	14.5	14.7	15.0				
			LSTM	06:42	18.6	18.8	19.1	19.6				
			BiLSTM	08:18	19.2	19.3	19.2	19.3		-		
			Persistence	-	14.3	15.1	15.8	16.5				
			Naive		16.3	16.9	17.5	18.1				
		8	ST-MDF*	14:52	15.3	15.1	15.0	15.1	15.2	15.5	15.8	16.2
			LSTM	07:44	19.9	20.2	20.5	20.1	21.0	21.2	21.0	22.0
			BiLSTM	11:02	68.3	70.1	70.6	71.0	70.9	69.4	67.5	65.8
			Persistence	-	14.3	15.1	15.8	16.5	17.1	17.8	18.4	19.0
			Naive		16.3	16.9	17.5	18.1	18.7	19.3	19.8	20.3
8	8	8	ST-MDF*	14:44	16.2	16.0	15.8	15.8	15.8	16.0	16.2	16.5
			LSTM	07:48	17.0	16.6	16.2	16.1	16.0	16.0	15.9	16.0
			BiLSTM	11:18	13.9	14.1	14.2	14.4	14.5	14.6	14.8	15.0
			Persistence	-	17.1	17.8	18.4	19.0	19.4	20.0	20.5	21.0
			Naive		18.7	19.3	19.8	20.3	20.8	21.3	21.8	22.2

Given the leap l, the forecast horizon is obtained as: $h_i = (i + l) \cdot 15\,\text{min}$, with $i \in \{0, \ldots, 7\}$. Units of n_x, l, and n_y are [15 min]. Time refers to training time, displayed in `HH:MM` format.

Comparison with the literature

Finding works that are comparable to our proposal is a challenging task. The foremost reason is that the majority of research studies do not disclose experimental code. In addition, the studied problem relies on many parameters that may differ,

Table 6.4: Error metrics of the ST-MDF and baseline models for the bike ride task.

n_x	l	n_y	**Model**	**Time**	**Mean wRMSE** $[10^{-2}$ **counts]** **across** $\mathcal{S}_v$							
					h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7
4	4	4	ST-MDF*	07:35	8.5	8.6	8.6	8.6				
			LSTM	04:24	13.6	13.5	14.4	14.2				
			BiLSTM	05:29	8.0	8.2	8.2	8.2		-		
			Persistence	-	7.9	8.4	8.8	9.2				
			Naive		7.5	7.9	8.3	8.6				
		8	ST-MDF*	09:40	8.5	8.5	8.6	8.6	8.6	8.6	8.6	8.7
			LSTM	05:44	8.0	8.3	8.1	8.0	8.0	8.6	8.4	8.2
			BiLSTM	07:50	8.2	8.2	8.3	8.4	8.5	8.6	8.7	8.7
			Persistence	-	7.9	8.4	8.8	9.2	9.5	9.7	9.9	10.1
			Naive		7.5	7.9	8.3	8.6	8.8	9.0	9.2	9.3
8	4	4	ST-MDF*	13:05	8.5	8.6	8.6	8.6				
			LSTM	06:42	9.1	9.2	9.2	9.2				
			BiLSTM	08:18	9.9	9.6	9.6	9.6		-		
			Persistence	-	8.0	8.4	8.8	9.2				
			Naive		7.8	8.1	8.3	8.6				
		8	ST-MDF*	14:52	8.5	8.5	8.6	8.6	8.6	8.6	8.6	8.7
			LSTM	07:44	9.7	9.4	9.9	9.3	9.2	9.2	9.0	9.0
			BiLSTM	11:02	30.1	30.9	30.6	29.4	28.0	26.1	24.8	24.3
			Persistence	-	8.0	8.4	8.8	9.2	9.5	9.7	9.9	10.1
			Naive		7.8	8.1	8.3	8.6	8.8	8.9	9.0	9.2
8	8	8	ST-MDF*	14:44	8.7	8.7	8.7	8.7	8.7	8.7	8.8	8.8
			LSTM	07:48	8.7	8.4	8.1	8.0	7.8	7.7	7.8	7.9
			BiLSTM	11:18	6.2	6.1	6.1	6.2	6.2	6.3	6.4	6.5
			Persistence	-	9.5	9.7	9.9	10.1	10.2	10.3	10.4	10.5
			Naive		8.8	8.9	9.0	9.2	9.3	9.4	9.4	9.5

Given the leap l, the forecast horizon is obtained as: $h_i = (i + l) \cdot 15\,\text{min}$, with $i \in \{0, \ldots, 7\}$. Units of n_x, l, and n_y are [15 min]. Time refers to training time, displayed in `HH:MM` format.

such as forecasted variables, temporal granularity, forecast horizons, target city, temporal coverage, exogenous information, error metrics, etc. Therefore, any comparison with the literature should not be overstated. Even so, if done carefully, we consider such a comparison enlightening.

This section aims to compare the error metrics of our proposal with the model and the baselines from Liu, Wu, et al. (2020). The authors of this work tackle the taxi trip forecasting task on NYC and Chengdu, China. Table 6.5 presents the (unweighted) RMSE of the ST-MDF$^*_{4,4,4}$ model for $h = 1$ h, which is the shortest studied horizon. The table shows that the errors of our and their proposal are akin, even if the cities and horizons differ. Another work by Xu et al. (2017) presents an LSTM-based model for taxi demand prediction in NYC. From their results, we can derive that their RMSE is above 2.5 on average. This value is consistent with Table 6.5 results. Lastly, Yao et al. (2018) present another spatio-temporal model

for taxi demand forecasting in Guangzhou, China. Their proposal and baseline models obtain RMSE metrics between 9.6 and 12.2.

Table 6.5: Error metrics of the ST-MDF and the models from Liu, Wu, et al. (2020) for the taxi trip task.

Variable	Reference	Horizon	City	Model	RMSE [-]
Taxi trips	Ours	1 h	Chicago, USA	ST-MDF*	2.43
	Liu, Wu, et al. (2020)	15 min	NYC, USA	CACRNN	3.17
				STGCN	3.48
				DCRNN	3.50
				LSTM	3.65
				ARIMA	9.53
				HA	5.84
			Chengdu, China	CACRNN	2.73
				STGCN	2.77
				DCRNN	3.14
				LSTM	4.30
				ARIMA	5.75
				HA	7.10

Context-aware Attention-based Convolutional RNN (CACRNN) is the model from Liu, Wu, et al. (2020). The baselines they present are Spatial-Temporal Graph Convolutional Network (STGCN), Diffusion Convolution RNN (DCRNN), LSTM, ARIMA, and Historical Average (HA). As discussed, consider the differences in horizons and target cities in the comparison.

In addition, we compare our proposal to the model and baselines from Chai et al. (2018) for bike demand forecasting in Chicago. Table 6.6 presents the (unweighted) RMSE of the ST-MDF$^*_{4,4,4}$ for $h = 1$ h. Their model is based on multi-graph CNNs, as discussed in Section 2.3.2. The table presents the RMSE as the mean across the 5 and 10 stations with the highest demand and the global mean.

6.4.3 Impact of temporal window width

It is worth considering whether certain parameters can enhance predictions on a given horizon. Specifically, we are interested in the impact of the number of input timestamps n_x on the error. In other words, does a wider n_x lead to improved forecasts? To investigate this, we trained several ST-MDF models with fixed $l = 4$ and $n_y = 4$ while increasing the value of n_x from one to five hours. Figure 6.4 depicts the results of this experiment. Section 6.5 discusses the results in further detail.

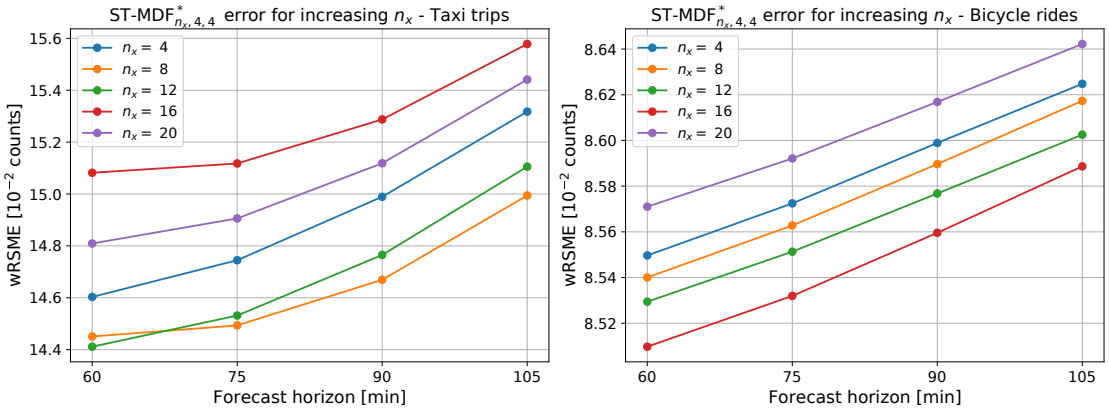
6.4.4 Flexibility and robustness

Earlier in the chapter, we discussed the need to analyze features beyond the error when modeling spatio-temporary fused data. As in the previous use case, we now investigate flexibility and robustness NFFs.

Table 6.6: Error metrics of the ST-MDF and the models from Chai et al. (2018) for the bike ride task.

Variable	Reference	Horizon	City	Model	RMSE [-] Top 5	Top 10	Mean
Bike rides	Ours	1 h	Chicago, USA	ST-MDF*	5.331	4.646	1.702
	Chai et al. (2018)			MG-CNN	5.177	4.930	3.658
				LSTM	6.231	5.853	4.405
				GBRT	5.945	5.738	4.410
				SARIMA	6.797	6.175	4.608
				ARIMA	9.853	8.535	6.163
				HM	7.078	6.179	4.347

"Top n" means that the RMSE is evaluated on the n sensors with higher demand. Multi-Graph CNN (MG-CNN) is the model from Chai et al. (2018). The baselines they present are LSTM, GBRT (Gradient Boosting Regression Tree), SARIMA (Seasonal ARIMA), ARIMA, and HM (Historical Mean).



(a) Results for taxi trips.

(b) Results for bicycle rides.

Figure 6.4: Impact of the temporal window width on the ST-MDF error. The ST-MDF$^*_{n_x,4,4}$ model is evaluated for $n_x \in \{4, 8, 12, 16, 20\}$, i.e., between 1 and 5 hours. Units of n_x, l, and n_y are [15 min].

For mobility demand, flexibility refers to adapting the model to a varying number and distribution of taxi zones or bike racks. Again, harnessing mesh-grids as described in Section 6.2.5 is critical for this NFF. Indeed, the interpolation step converts the (variable) number of stations/racks into the fixed-size mesh-grid. In this way, the input to the model is decoupled from the number and location of zones and bike stations. Therefore, taxi zones and bicycle racks can be added or removed without modifying the ST-MDF* model.

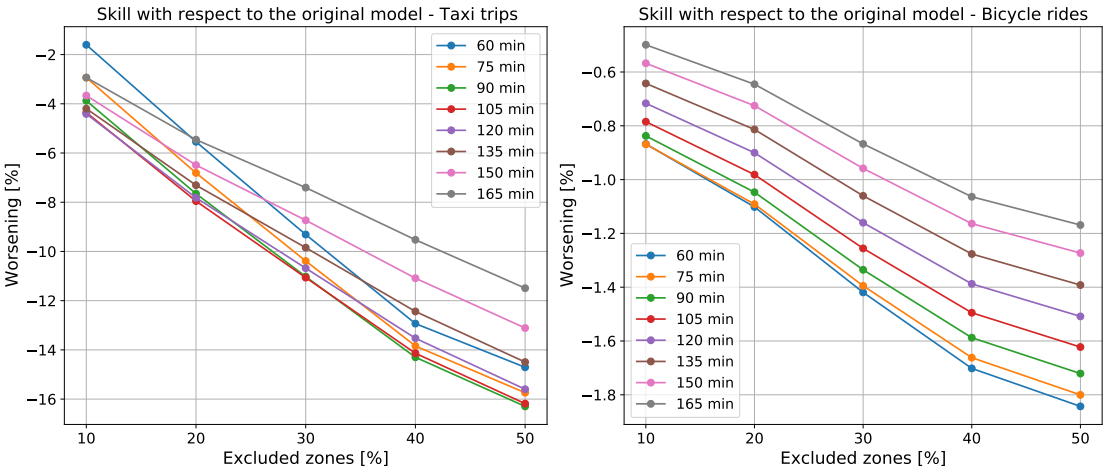
Considering the set of taxi and bike sensors, one may expect that some may temporarily fail. In this situation, a robust forecasting framework would recover an estimation of that missing data. Again, we can fill those gaps thanks to the interpolation into mesh-grids with the aid of neighboring observations. This method,

in turn, allows the model to produce reliable predictions. We followed the approach described in the methodology to evaluate this feature (see Section 4.3.3). Table 6.7 shows how errors evolve when random bike racks or taxi zones stop working. The percentage of failing sensors ranges from 10 % to 50 % (with steps of 10 %). We evaluate the test set for each case when removing those random sensors for twenty independent repetitions. We apply the skill criterion (Eq. 4.9) to evaluate how the error degrades. We use the original model that considers all sensors as the reference to calculate the skill. Additionally, Figure 6.5 splits the resuls by horizons for the ST-MDF$^*_{4,4,8}$ model.

Table 6.7: Robustness test results for the ST-MDF* model.

Variable	n_x	l	n_y	**Worsening [%] when missing** 10 %	20 %	30 %	40 %	50 %
Taxi trips	4	4	4	-11.7	-15.3	-15.1	-25.3	-26.3
			8	-3.5	-6.9	-9.8	-12.7	-14.7
	8	4	4	-5.6	-13.8	-20.3	-23.7	-29.2
			8	-5.3	-8.3	-11.6	-13.3	-15.6
	8	8	8	-5.8	-10.3	-8.7	-12.9	-16.7
Bike rides	4	4	4	-1.3	-1.6	-1.6	-2.2	-2.2
			8	-0.7	-0.9	-1.2	-1.4	-1.5
	8	4	4	-0.8	-1.3	-1.6	-1.9	-2.2
			8	-0.5	-0.7	-0.9	-1.2	-1.4
	8	8	8	-0.1	-0.4	-0.2	-0.5	-0.5

The worsening is the average across horizons of the average across 20 random repetitions. Units of n_x, l, and n_y are [15 min].



(a) Results for taxi trips.

(b) Results for bicycle rides.

Figure 6.5: Robustness test results for the ST-MDF* model with $n_x = 4, l = 4$, and $n_y = 8$. Units of n_x, l, and n_y are [15 min].

6.4.5 Extensibility

Section 6.2.5 detailed how to combine different mobility variables into a single data structure. This section evaluates the extensibility of the ST-MDF model. In other words, we seek to determine whether a model with both variables scores better error metrics than one model for each variable. To investigate this, we trained an ST-MDF$^{*}_{4,4,4}$ model for taxi trips and another one for bike rides. The model that works with both mobility services at once obtained 2.7, 3.5, 4, and 4.2 % better results on each of the four horizons. As for bike rides, ST-MDF improved 5.6, 5.4, 5.1, and 4.8 % the error per horizon. Table 6.8 presents the wRMSE scored by these models.

Table 6.8: Extensibility test results for the ST-MDF$^{*}_{4,4,4}$ model.

Variable	Combined variables	wRMSE $[10^{-2}$ counts]			
		h_0	h_1	h_2	h_3
Taxi trips	Yes	14.6	14.7	15.0	15.3
	No	15.0	15.3	15.6	16.0
Bike rides	Yes	8.5	8.6	8.6	8.6
	No	9.1	9.1	9.1	9.1

Given that the leap $l = 4 \cdot 15$ min: $h_0 = 60$, $h_1 = 75$, $h_2 = 90$, and $h_3 = 105$ minutes.

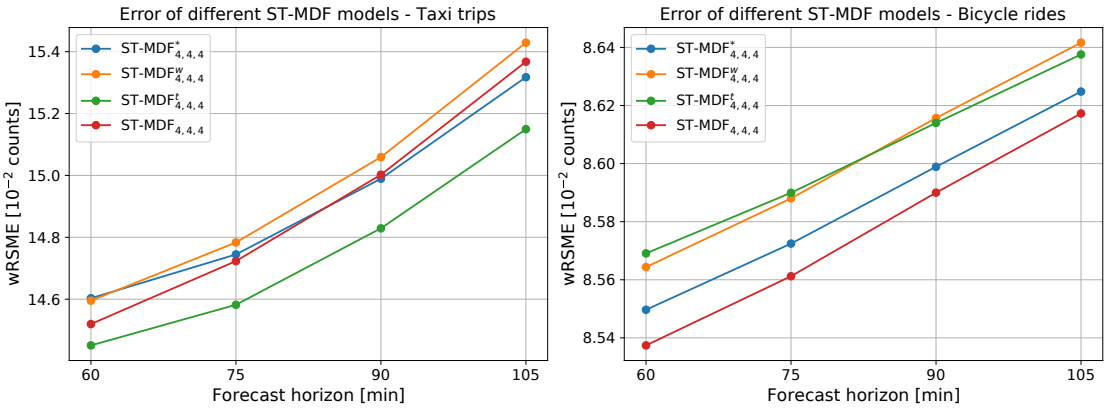
6.4.6 Integrability

This section evaluates the integrability NFF of the ST-MDF model. For this, we perform an ablation study on the modules that constitute it. This way, we can assess their impact on the model's error. As described in Section 6.3.3, we consider four variations of the model: ST-MDF, ST-MDF$^{\text{t}}$, ST-MDF$^{\text{w}}$, and ST-MDF*. Figure 6.6 shows the results when choosing $n_x = 4, l = 4$, and $n_y = 4$. We further discuss the results in Section 6.5.

6.5 Summary and discussion of results

The proposed ST-MDF model for mobility demand forecasting offers several NFFs. Besides, it combines heterogeneous spatio-temporal data thanks to the mesh-grids. ST-MDF presents these benefits while obtaining similar or better error metrics than the baselines and models from the literature. The proposal integrates two mobility demand variables, multiple timestamps, and sensors in the input and output. It additionally benefits from a modular structure, so the model can accommodate exogenous inputs (see Section 6.3.1).

Section 6.4.2 studies the error of the ST-MDF compared to baselines and the literature. Tables 6.3 and 6.4 show that our proposal beats the baselines in most of the studied situations. As discussed in Chapter 5, the persistence model is a reasonable baseline, particularly for short horizons. Applying the skill criterion



(a) Results for taxi trips.

(b) Results for bicycle rides.

Figure 6.6: Integrability test results for the ST-MDF model with $n_x = 4$, $l = 4$, and $n_y = 4$. Units of n_x, l, and n_y are [15 min].

(Eq. 4.9) with persistence as a reference, we get that the ST-MDF is up to 21.1 % better for taxis and 16.5 % for bikes. We also compared our model with several from the literature. Table 6.5 shows that our model's error is within the same range of Liu, Wu, et al. (2020) for taxi trips. In the comparison, we should consider that the horizon varies slightly and the target city differs. As for bike rides, Table 6.6 compares our model with those of Chai et al. (2018). In this case, the horizon and city are the same, and the error is comparable or lower. Notably, models in the literature work with a single mobility variable, while the ST-MDF combines both.

Section 6.4.3 studies the impact of the temporal window width n_x on the error. Interestingly, Figure 6.4 shows that different n_x are optimal depending on the predicted variable. For taxis, $n_x = 8$ provides the best error, while for bikes it is $n_x = 16$. We fixed $l = 4$ and $n_y = 4$ in all cases. The different volumes of taxi trips and bike rides may cause this behavior. Another cause could be the more stable dynamics or the taxi ride problem. Additionally, the error per horizon is rather consistent with the choice of the parameter n_x.

The mobility mesh-grid allows the ST-MDF to work with a variable number of sensors. We study this property in Section 6.4.4. When changes in the number or distribution occur, the model can continue working. We study how the model behaves when random sensors stop working to test its robustness. The missing information is dynamically filled thanks to the grid interpolation. Table 6.7 shows that error deteriorates within manageable limits. Besides, Figure 6.5 splits the results per horizon for the ST-MDF$^{*}_{4,4,8}$ model. Figure 6.5a reveals that wRMSE deteriorates only 11 % or less when randomly removing 30 % for taxis. Similarly, wRMSE worsens less than 1.4 % for bike rides when up to 30 % of the testing data is missing (see Figure 6.5b). The different volume of trips likely causes the contrast between the robustness of taxis and bikes. Specifically, the number of taxi trips is over seven times higher than that of bicycles.

We also studied extensibility NFF in this use case. Section 6.4.5 shows that the ST-MDF* model benefits from combining two variables into a single data structure. The RMSE of this model is 3.6 % and 5.2 % better on average than the same model for just taxis or bicycles, respectively. This experiment suggests that predictions of a mobility service can benefit from observations of a different one. For example, good weather may increase bikes demand while reducing taxis demand.

The modular structure of the ST-MDF opens the door to studying the integrability NFF. Section 6.3.1 describes two exogenous modules for harnessing seasonal and weather observations. Additionally, the main module of our proposal consumes the mobility demand mesh-grids. Section 6.4.6 studies how the ST-MDF behaves when we remove the periodical, weather, or both modules. This kind of experiment is usually known as an ablation study. Figure 6.6a shows that the ST-MDF$^{\text{t}}$ model consistently outperforms its three peers for taxi trips. However, the opposite happens for bike rides (see Figure 6.6b). As for the ST-MDF model, even if it comes first for this second task, its performance for taxi trips could be better (especially for the farthest horizons). We can observe a similar situation for the ST-MDF$^{\text{w}}$ model for the second problem. Therefore, ST-MDF* seems to provide the most consistent behavior among its peers when considering both mobility prediction tasks. Furthermore, in both cases, its wRMSE increase for farther horizons is less pronounced than for the other three models.

As we have seen, the ST-MDF model provides several benefits over a classical DL model. For example, during the data analysis process, we noticed that some bike stations changed their location over time. Construction works or relocations may have generated this artifact. Such a situation can pose a forecasting challenge for a model that relies on specific sensor locations. However, we bypassed this problem by working with mesh-grids, which provide flexibility. Additionally, situations such as the COVID-19 pandemic can be a pitfall for forecasting. We opted for trimming the testing period since the lockdown significantly changed people's mobility habits, as discussed in Section 4.1.1. Nevertheless, we can always tune the models to adapt to new mobility dynamics if such situations arise.

The ST-MDF model estimates taxi and bicycle ride demand based on historical observations of those variables and exogenous information. However, urban mobility demand is inherently more complex, particularly in large cities that offer a wide variety of transport options. Recently, traditional public transport systems, such as buses and metro lines, have been complemented by micromobility alternatives, like shared bikes, electric scooters, and mopeds. With such a variety of options available, citizens often combine them on their commuting habits, making mobility forecasting increasingly challenging. Therefore, the extension of our model with additional mobility variables is highly topical, an idea we further discuss as future work (see Section 8.2.2). In this context, the four NFFs studied in this chapter provide a solid foundation for future expansions of the ST-MDF model.

CHAPTER 7

Third use case: Air quality

The third and last use case that we study is air quality forecasting. In this chapter, we model up to eleven variables and introduce exogenous observations from several sensors. As in the previous use case, we study error metrics alongside the following NFFs: flexibility, robustness, extensibility, and integrability. This chapter is based on our work (Prado-Rujas et al., 2024). We have made available the supplementary code (Prado-Rujas et al., 2023c) and datasets (Prado-Rujas et al., 2023b) for reproducibility.

This chapter is organized as follows. Section 7.1 introduces the topic and our contributions and describes the characteristics and hazards of the studied pollutants. Afterward, Section 7.2 applies our 7-phase methodology (described in Section 4.1) to air quality sensor observations. Section 7.3 details the baselines and our proposal, the Spatio-Temporal Air Quality Forecaster (ST-AQF). We summarize our experiments in Section 7.4, and discuss them in Section 7.5.

7.1 Introduction and background

7.1.1 Introduction

Today, pollution is one of the leading health threats. Diseases induced by outdoor air pollution caused around 4.2 million premature deaths worldwide in 2015 (Landrigan et al., 2018) and in 2019 (World Health Organization, 2022b). Even so, these figures do not take into account household air pollution. According to other sources (Health Effects Institute, 2020, and OpenAQ[1]), this figure almost reached 7 million in 2019, ranking fourth as a death risk factor. Furthermore, around 89 % of these deaths occurred in low-income and middle-income countries (World Health Organization, 2022a). On top of that, air pollution is particularly harmful to children and older people. These facts make air pollution monitoring

[1] https://openaq.org/why-air-quality/

and modeling fundamental for city councils and researchers worldwide. On the one hand, **collecting and analyzing air pollutants concentrations** allows to: (1) detect the most polluted areas; (2) identify the causes of such pollution; and (3) evaluate the effect of the adopted policies to reduce it. We discussed this aspect of air pollution in Section 2.4.1. On the other hand, **modeling air pollution** is helpful to anticipate high-pollution episodes and advise the population accordingly, as seen in Section 2.4.2. This advice is particularly important for sensitive areas, such as schools, nursing homes, and hospitals, where children, older people, and sick people are present.

Despite the current effort to model air quality, this field still has room for improvement. As we saw in Section 2.4, other works present some limitations. These models are often limited to a single pollutant and a certain number of sensors. They also struggle to handle sensor failure, and their temporal coverage is usually too small to provide reliable results. Besides, exogenous data are usually neglected. In this scenario, NFFs are crucial due to the dynamic nature of air pollution and how these data are harvested.

This chapter develops the ST-AQF model. Our proposal combines measured air pollutants, weather, and calendar information to predict future air quality. Data are collected from real-world sensors spread throughout a ROI and cover the eleven pollutants described in Section 7.1.2. As we will discuss, we could adapt the model to accommodate the number of pollutants specific to the use case. The key features of this framework are as follows:

- **Error metric**: ST-AQF outperforms the baseline models while achieving an error comparable to that of related work.
- **NFFs:**
  - **Flexibility**: Adaptability to a variable number of sensors (up to 24 in our experiments).
  - **Robustness**: Ability to cope with sensor failure.
  - **Extensibility**: Up to eleven pollutant variables.
  - **Integrability**: Exogenous information improves predictions.
- **Spatio-temporal**: Multiple timestamps and locations in the input and output.
- **Fine spatial and temporal resolution**: ∼500 m and 1 h.
- **Extensive temporal coverage**: Ten years of observations.

7.1.2 Background

According to European Environment Agency et al. (2020), some of the most harmful pollutants for human beings are particulate matter (PM), NO_2, and ground-level O_3. PM is usually further divided into PM_{10} and $PM_{2.5}$ based on their diameter: 10 or 2.5 micrometers or less in diameter, respectively. Other hazardous gases include SO_2, NO_x, CO, and hydrocarbons, such as toluene (TOL), benzene (BEN), and ethylbenzene (EBE) (Domingo & Rovira, 2020; Kim et al., 2018). Recently, the World Health Organization (WHO) published an air quality guidelines report based on a systematic review (World Health Organization, 2021). The EC has also established standards for exposure to these pollutants (Directorate-General for Environment, European Commission, 2008). Based on these studies, we consider the following pollutants in this use case:

- **Coarse particulate matter (PM_{10}) and fine particulate matter ($PM_{2.5}$)**: Generally, particulate matter serves as a proxy indicator of air pollution. Fundamentally, it includes sulfate, nitrates, ammonia, sodium chloride, black carbon, mineral dust, and water. The difference between PM_{10} and $PM_{2.5}$ is that the former cannot penetrate the lungs into the bloodstream, while the latter can. PM is closely related to mortality and morbidity (World Health Organization, 2022a).

- **Ozone (O_3)**: This pollutant mainly affects the lungs and can cause asthma (World Health Organization, 2022a). Furthermore, it worsens the mortality of chronically ill people. It appears when sunlight reacts with pollutants emitted mainly by cars, such as NO_2. Therefore, higher concentrations of O_3 appear during the spring and summer seasons. Thermoelectric plants, industrial activities, and intensive livestock farming are additional sources of O_3.

- **Nitrogen dioxide (NO_2)**: According to World Health Organization (2022a), it can cause bronchitis in asthmatic children and reduce lung function and growth. The primary source of NO_2 are fossil fuels burned by internal combustion engines.

- **Nitrogen monoxide (NO)**: It produces nitric acid when it reacts with water, contributing to acid rain. In addition, it reacts with ozone, resulting in the depletion of the ozone layer. It also forms in combustion systems.

- **Nitrogen oxides (NO_x)**: It is a generic term for denoting nitrogen oxides, including NO_2 and NO.

- **Sulfur dioxide (SO_2)**: World Health Organization (2022a) reports that SO_2 can inflame the respiratory system, exacerbating asthma and chronic bronchitis. It can also cause eye irritation. Furthermore, SO_2 forms sulfuric acid when combined with water, which is the main component of acid rain.

- **Carbon monoxide (CO)**: The most common source of this pollutant is thermal combustion, both from combustion engines and industry. Prolonged exposure to this toxic gas can cause death.
- **Benzene (BEN)**: C_6H_6 (or BEN for simplicity, based on Madrid's Open Data Portal notation) is a hydrocarbon. It is a key component of crude oil and can appear due to both natural (e.g., wild fires and volcano eruptions) and human causes (e.g., plastic industry, combustion engines, tobacco, etc.). BEN is a carcinogen, and no exposure to it is considered safe.
- **Toluene (TOL)**: C_7H_8 is another hydrocarbon usually used as a precursor to benzene. Furthermore, it is commonly employed as a solvent for paints and sometimes to improve certain fuels. Even if it is much less toxic than benzene, it could cause unconsciousness and even death.
- **Ethylbenzene (EBE)**: C_8H_{10}, yet another hydrocarbon, is obtained from benzene and ethylene and commonly used to produce polystyrene. Moreover, ethylbenzene appears when gas or coal is burned and can be found in tobacco smoke. As for its effects, it is a precursor to smog and causes eye and throat sensitivity.

These facts evince the importance of air pollution for human health. Therefore, a tool capable of predicting high-risk scenarios would help take preventive measures. Such a tool should predict future values of these pollutants but also be flexible and robust. These NFFs are meaningful in a real-world scenario where existing sensors can fail and new ones are regularly installed. Furthermore, incorporating new pollutants into the study would also be relevant. In addition, integrating meteorological and public holiday information into the tool should improve forecasts. This chapter combines all these NFFs without worsening the error compared to baselines and the literature, as the experiments will show.

7.2 Data analysis and transformation

As in the previous use cases, we start by adapting our 7-phase data analysis and transformation methodology defined in Section 4.1. Our study focuses on air quality in Madrid, using data obtained from the Madrid City Hall's open data portal (Ayuntamiento de Madrid, 2024). The AEMET also provided weather observations from several stations. In addition, we incorporate seasonal information extracted from the timestamp, as in Chapter 6.

7.2.1 Phase 1: Exploratory Data Analysis

The Madrid City Hall's open data portal gathers air quality measurements using a network of 24 sensors. Every few seconds, each of them observes different pollutants using various measurement techniques (listed in this document). These

observations are then averaged every ten minutes. These values, in turn, are averaged hourly and published in the open data portal. In this section, we go over the EDA questions that we laid on Section 4.1.1:

1. Sensors

The set $\mathcal{S}$ consists of 24 air pollution sensors. Each of them observes different variables, as we will see subsequently. Additionally, we consider six weather sensors. Figure 7.1 depicts the identifiers of all of them over Madrid.

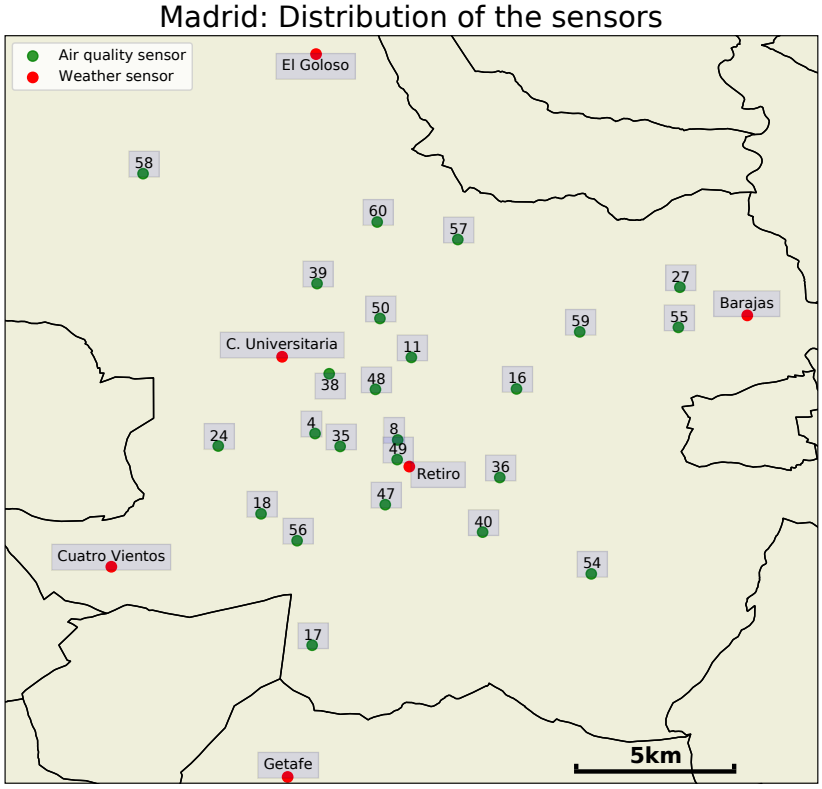


Figure 7.1: Madrid's metropolitan area, Spain, scattered with the geolocations of the 24 air pollution sensors (in green) and six weather stations (in red).

2. Variables

We study the eleven pollution variables described in Section 7.1.2, namely: $PM_{2.5}$, PM_{10}, O_3, NO, NO_2, NO_x, SO_2, CO, TOL, BEN, and EBE. Therefore, $n_v = 11$. These are all continuous variables.

Additionally, we consider the following weather variables: temperature, precipitation, wind speed and direction, and humidity. We also introduce periodical variables that influence air quality. As in Chapter 6, we consider time of the day, time of the week, time of the year, weekday or weekend, and holiday or working day.

3. Variable units

All pollution variables' units are $\mu g/m^3$, except for CO, measured in mg/m^3. We will present the weather variables' units at the end of this section. Periodical variables are measured in seconds (except weekdays and holidays, which are boolean).

4. Variables observed by every sensor

As mentioned earlier, each sensor observes a different number of pollutants. Besides, not all sensors measure their variables at all times. For instance, $PM_{2.5}$ is covered by at least four sensors over 99 % of the temporal coverage. Other pollutants are generally covered by all 24 sensors, such as NO, NO_2, and NO_x. Table 7.1 presents which sensors observed which variables between 2010 and 2019. In addition, each graph from Figure 7.2 displays the number of sensors that observed each pollutant over time. The recurrent drop at the begining of each year is due to the daylight saving time shift.

Each weather sensor observes all weather variables. The minimum number of active sensors was four for precipitation and humidity, while for the remaining variables was three. The median number of active sensors was five for wind speed and direction and six for the rest. Periodical variables do not rely on sensors but are inherent to the timestamps, as discussed in Section 6.2.1 of the previous use case.

5. Geolocation of every sensor

Figure 7.1 shows the spatial distribution of the 24 air pollution sensors and the six weather sensors. The median distance between the pollution sensors is 8.7 km, and the average minimum distance to another sensor is 2.6 km. As for the weather sensors, the median distance is 14.4 km and the average minimum distance is 9.3 km. Their specific coordinates can be found in our repository (Prado-Rujas et al., 2023c).

6. Spatial coverage

The spatial coverage comprises the city of Madrid and part of its metropolitan area. This encompasses an area of approximately 20 by 17 km, as Figure 7.1 shows.

7. Temporal granularity

In this use case, all air quality and weather variables have $\tau = 1$ h, except for humidity. This variable is measured at 00:00, 07:00, 13:00, and 18:00 (UTC).

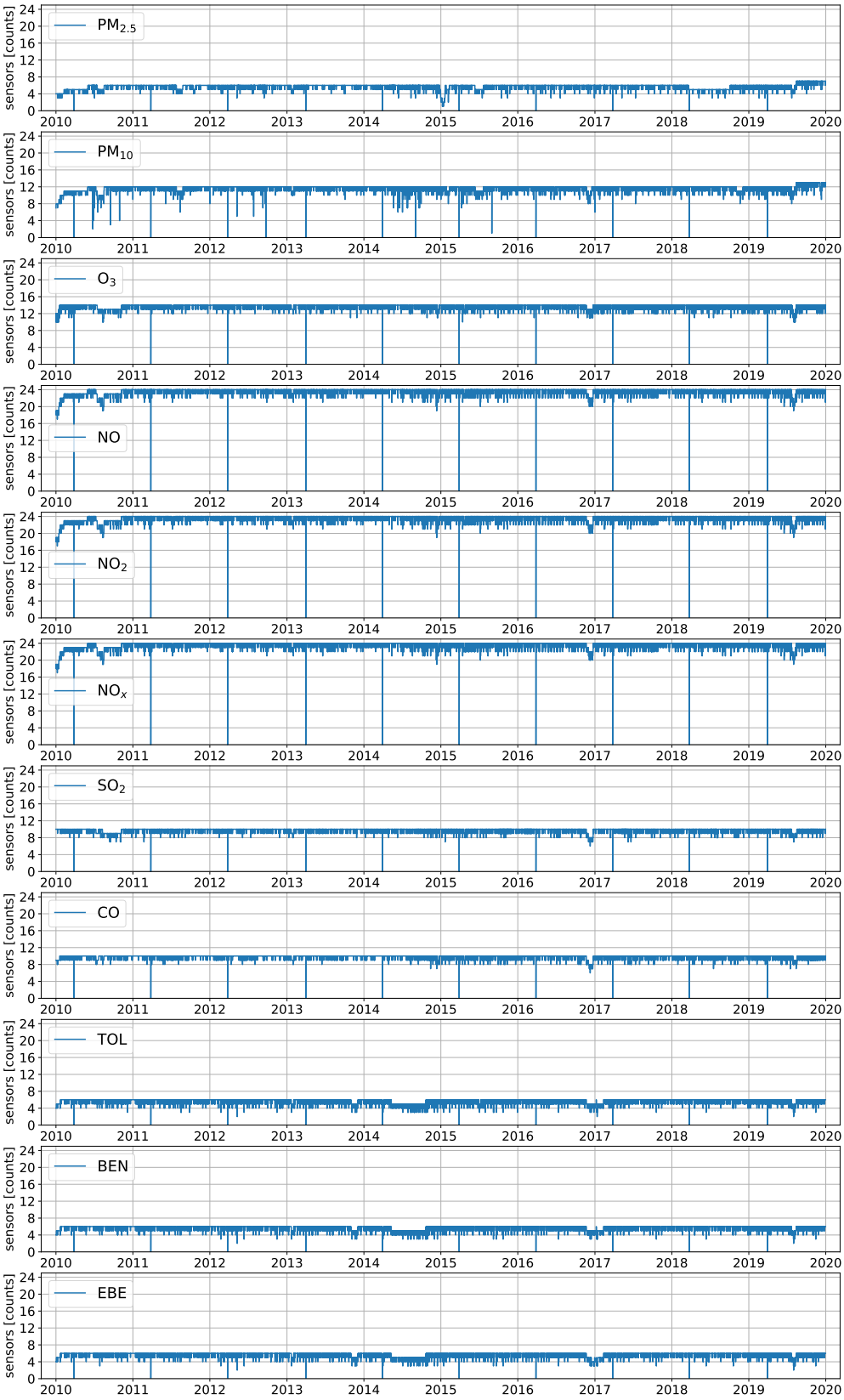


Figure 7.2: Hourly air quality sensor count per variable (2010 – 2019).

Table 7.1: Variables observed by air quality sensors (2010 – 2019, $\tau = 1$ h).

Sensor	$PM_{2.5}$	PM_{10}	O_3	NO	NO_2	NO_x	SO_2	CO	TOL	BEN	EBE	**total**
04				✓	✓	✓	✓	✓				5
08	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	11
11				✓	✓	✓			✓	✓	✓	6
16			✓	✓	✓	✓		✓				5
17			✓	✓	✓	✓	✓					5
18		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10
24	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	11
27			✓	✓	✓	✓						4
35			✓	✓	✓	✓	✓	✓				6
36		✓		✓	✓	✓	✓	✓				6
38	✓	✓		✓	✓	✓	✓		✓	✓	✓	9
39			✓	✓	✓	✓		✓				5
40		✓		✓	✓	✓	✓					5
47	✓	✓		✓	✓	✓						5
48	✓	✓		✓	✓	✓						5
49			✓	✓	✓	✓						4
50	✓	✓		✓	✓	✓						5
54			✓	✓	✓	✓						4
55		✓		✓	✓	✓			✓	✓	✓	7
56	✓	✓	✓	✓	✓	✓		✓				7
57		✓		✓	✓	✓	✓	✓				7
58			✓	✓	✓	✓						4
59			✓	✓	✓	✓						4
60		✓	✓	✓	✓	✓						5
min	1	1	10	17	17	17	6	6	2	2	2	
median	6	12	14	24	24	24	10	10	6	6	6	
max	7	13	14	24	24	24	10	10	6	6	6	

The last column is the total number of pollutants observed by each sensor. The last rows are the minimum, median, and maximum number of sensors recording each pollutant.

8. Temporal coverage

We studied observations from January 2001 to April 2022. However, we decided to restrict the temporal coverage to the decade 2010 – 2019. The main reason is that some variables were not observed by enough sensors, such as $PM_{2.5}$ and hydrocarbons.

9. Missing observations

The number of sensors observing each variable is not constant, as Table 7.1 shows. Given this fact, we cannot directly estimate a single value of missing observations per variable. Nevertheless, given that the median and maximum number of sensors (last and second last rows of Table 7.1) almost match, we can assume very few missing observations. Figure 7.2 also supports this fact, where the sensor count closely matches the maximum number of available sensors.

In the upcoming statistics section, we additionally provide the exact number of observations per variable. This information shows that missing observations are below 5 % for all pollutants. As for weather variables, the number of missing observations is below 4 %, except for wind, which is below 9 %.

10. Main statistics

Table 7.2 summarizes the main statistics of the raw air quality observations with $\tau = 1$ h. In addition, Figures 7.3 and 7.4 show hourly boxplots of the pollutant data for the studied period (2010 – 2019). We removed the 2 % highest measurements from the pollutant analysis to obtain a clearer description of the data, i.e., from Table 7.2 and Figures 7.3 and 7.4.

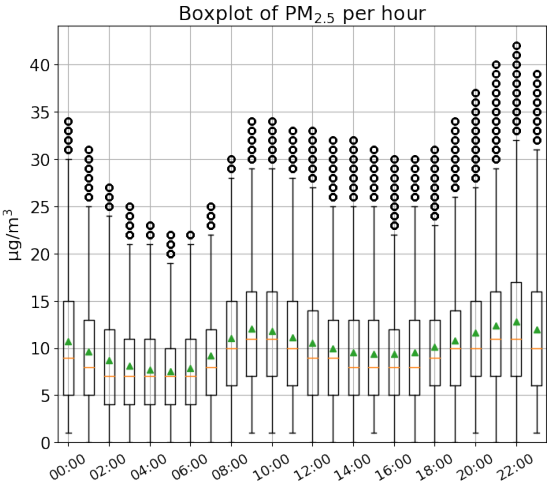
The boxplots show the periodic trend of the pollutants: higher concentrations tend to accumulate around rush hours. O_3 behaves differently, obtaining the lowest measurements in the early morning and the highest ones between 15:00 and 16:00. The orange lines and green triangles show the median and mean values, respectively. Whiskers are calculated from the first and third quartiles (Q1 and Q3). If the interquartile range IQR ≡ Q3 - Q1, then the upper whisker extends to the last datum smaller than Q3 + 1.5 · IQR. Similarly, for the lower whisker: Q1 - 1.5 · IQR.

Table 7.3 presents the same statistics as in Table 7.2 for the weather variables. As in Section 6.2, we convert wind speed and direction into a wind vector using Eq. 6.1.

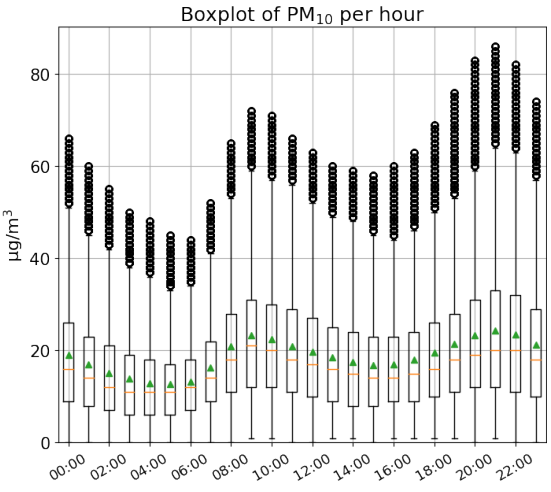
Table 7.2: Summary of the air quality variables measured in Madrid, Spain (2010 – 2019, $\tau = 1$ h).

Variable	Units	Number of observations	Values recorded by the sensors						
			mean	std	min	25 %	50 %	75 %	max
$PM_{2.5}$	$\mu g/m^3$	497 512	10.1	6.4	0.0	5.0	9.0	14.0	32.0
PM_{10}	$\mu g/m^3$	1 012 481	18.6	13.0	0.0	9.0	16.0	25.0	66.0
O_3	$\mu g/m^3$	1 187 574	46.8	31.2	0.6	19.0	46.3	69.7	123.7
NO	$\mu g/m^3$	2 037 902	17.1	28.4	0.0	2.0	6.0	18.0	178.0
NO_2	$\mu g/m^3$	2 037 535	36.7	25.7	1.0	16.0	31.0	52.0	119.0
NO_x	$\mu g/m^3$	2 037 818	63.2	65.0	2.0	20.0	40.0	81.0	380.0
SO_2	$\mu g/m^3$	847 376	6.3	3.8	0.0	3.0	5.0	8.0	19.0
CO	mg/m^3	846 442	0.3	0.2	0.1	0.2	0.3	0.4	1.0
TOL	$\mu g/m^3$	493 950	2.4	2.4	0.1	0.7	1.5	3.5	12.5
BEN	$\mu g/m^3$	495 486	0.6	0.6	0.0	0.2	0.4	0.7	3.2
EBE	$\mu g/m^3$	494 176	0.5	0.6	0.0	0.1	0.3	0.8	3.4

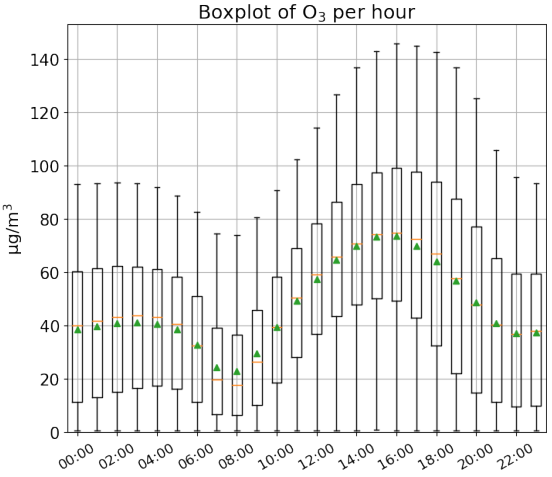
Abbreviations are as follows. Min: minimum, max: maximum, std: standard deviation, X %: X^{th} percentile. The 2 % highest values were removed from each pollutant to facilitate analysis.



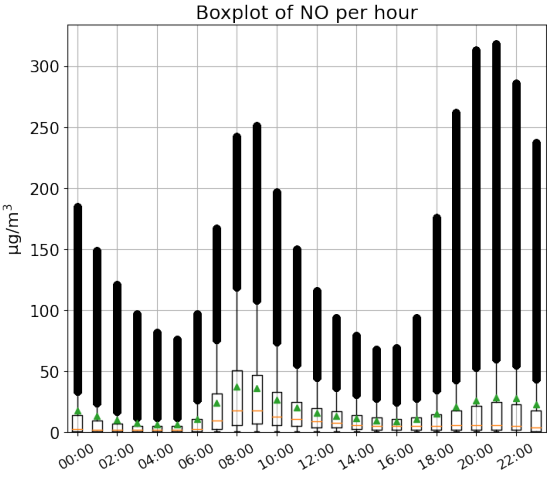
(a) Results for $PM_{2.5}$.



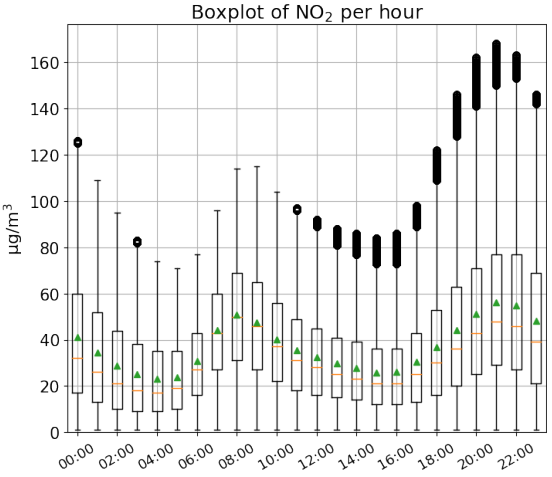
(b) Results for PM_{10}.



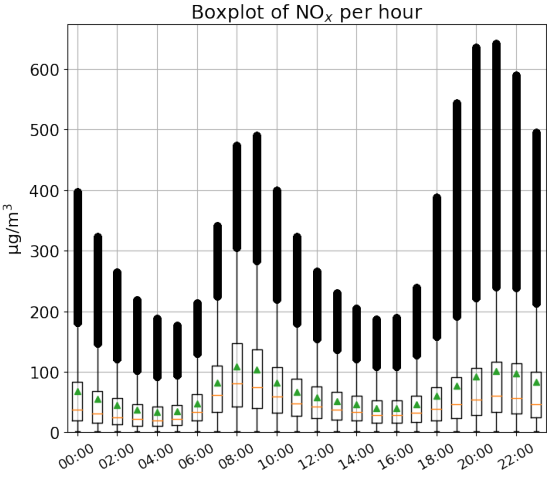
(c) Results for O_3.



(d) Results for NO.

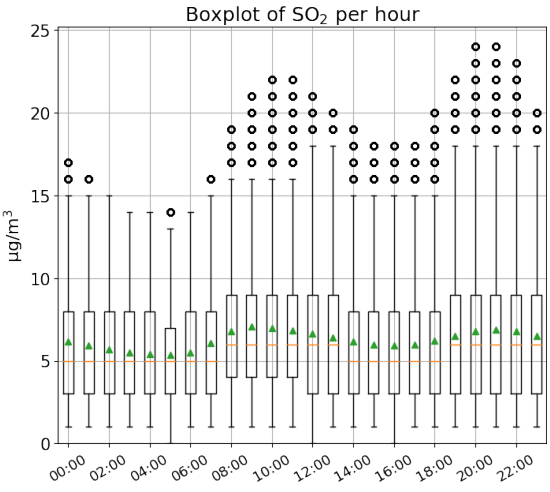


(e) Results for NO_2.

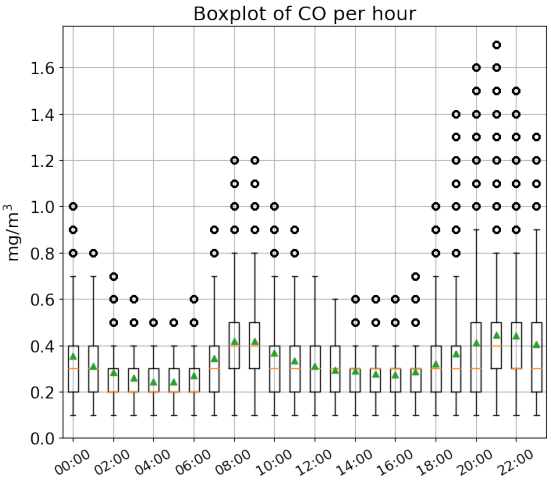


(f) Results for NO_x.

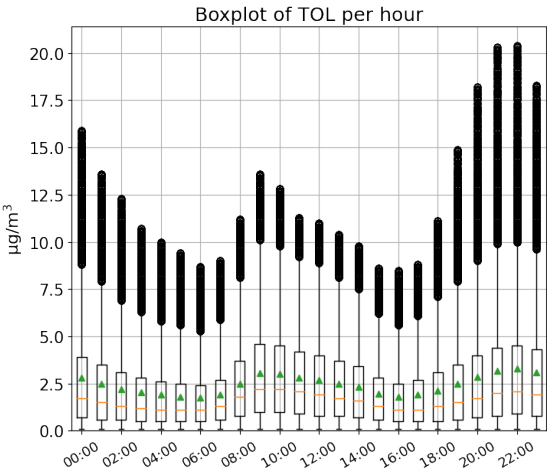
Figure 7.3: Hourly boxplots of each pollutant (2010 – 2019), part 1.



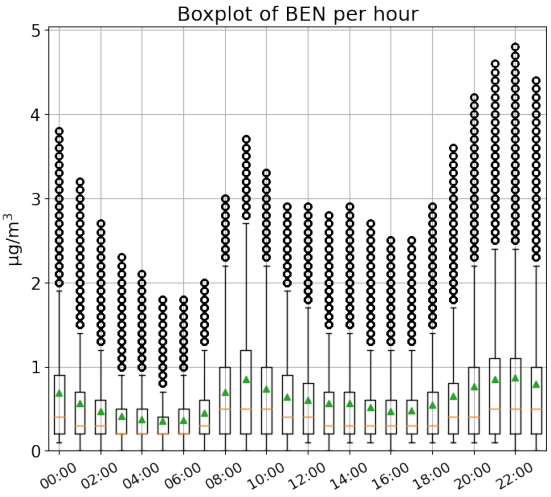
(a) Results for SO_2.



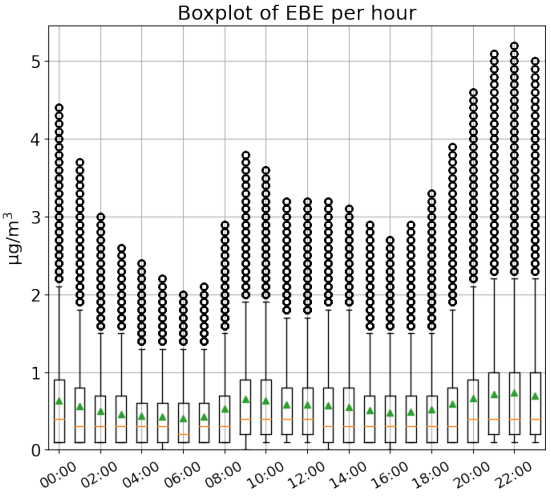
(b) Results for CO.



(c) Results for TOL.



(d) Results for BEN.



(e) Results for EBE.

Figure 7.4: Hourly boxplots of each pollutant (2010 – 2019), part 2.

Table 7.3: Summary of the weather variables measured in Madrid, Spain (2010 – 2019).

Variable	Units	Number of observations	Values recorded by the sensors						
			mean	std	min	25 %	50 %	75 %	max
Temperature	1/10 °C	514 606	151.9	89.0	-90.0	82.0	140.0	218.0	419.0
Precipitation	1/10 mm	508 128	0.4	3.6	0.0	0.0	0.0	0.0	430.0
Wind-x	km/h	478 661	-0.1	7.2	-59.0	-3.5	0.0	3.8	58.0
Wind-y	km/h	478 661	-1.4	8.1	-59.2	-4.8	0.0	3.5	74.8
Humidity	%	85 818	56.8	23.1	5.0	38.0	55.0	76.0	100.0

Abbreviations are as follows. Min: minimum, max: maximum, std: standard deviation, X %: X^{th} percentile. For all variables, $\tau = 1$ h except for humidity (measured at 00:00, 07:00, 13:00, and 18:00 UTC).

7.2.2 Phase 2: Temporal resolution

In Section 7.2.1, we established that $\tau = 1$ h for all air quality and weather variables, except humidity. Sensors only record this weather variable at 00:00, 07:00, 13:00, and 18:00 (UTC). We forward filled the remaining hours by taking the latest available ones. After this transformation, $\tau = 1$ h for all sensor observations.

7.2.3 Phase 3: Tabular dataset

We have now explored the data and converted them into a standard temporal resolution in Sections 7.2.1 and 7.2.2, respectively. In this phase, we aggregate all observations to facilitate the mesh-grid construction later. Once done, we can index all observations based on the timestamp, sensor, and variable (as in Table 4.3).

We have that $n_v = 11$, $n_s = 24$, and $n_t = 24$ timestamps/day $\cdot(10 \cdot 365 + 2)$ days $= 87\,648$ timestamps (considering two leap years). We would have over 23 million air quality observations if every sensor covered all variables and timestamps. However, as discussed in Section 7.2.1, this is not true. Nevertheless, the total number of air quality observations is almost 12 million (see Table 7.2). Considering the actual sensors that observe each variable, there are less than 5 % missing observations (see Section 7.2.1). We do not need to estimate these values since the mesh-grid interpolation will handle them in Section 7.2.5.

Regarding weather observations, six sensors observed five variables. The number of timestamps is the same as for air pollutants. Section 7.2.1 discussed that missing values are below 4 %, except for wind, which is below 9 %. We estimate them based on observations from the station with the least gaps (i.e., Barajas) on the same timestamp. If these values were also missing, the next most reliable station was utilized (i.e., Retiro).

7.2.4 Phase 4: Data scaling

As in the previous use cases, we scale data observations to facilitate their modeling. We used both min-max normalization (Eq. 4.1) and standardization (Eq. 4.2). We explore which scaling technique behaves better for this use case in Section 7.4.1. We additionally normalized weather variables following Equation 4.1. We scaled each variable independently in all cases.

7.2.5 Phase 5: The mesh-grid

In this section, we build the air quality mesh-grid as described in phase 5 of our methodology (Section 4.1.5). Initially, we fix the spatial resolution according to the number of sensors and their geolocations. Based on this, we build the mesh-grids for the spatio-temporal forecasting module.

Phase 5.1: Spatial resolution and grid placement

First, we define a rectangular spatial coverage encompassing the air quality sensors by taking their minimum and maximum latitudes and longitudes. We add an offset $\varepsilon = 0.004^{\text{o}}$ to these values, yielding an area of about 20 by 17 km. We then fix the parameters $(n_r, n_c) = (35, 30)$, obtaining $\sigma \approx (570, 570)$ meters. Therefore, the ratio between sensors and grid elements is about 2.3 %. However, this value suffices to build a reasonable grid since all variables are continuous, and the sensors are fairly spread out.

Phase 5.2: From sensors to mesh-grids

In this section, we describe how we build the air quality mesh-grids. For all pollutants and timestamps, we generate two different sets of mesh-grids: one using two-dimensional nearest-neighbor interpolation and a second one using linear interpolation. As discussed in Section 6.2.5, linear interpolation can only provide estimates for points within the convex hull of $\mathcal{G}$. Furthermore, air pollutants may dilute rather than plummet outside the convex hull, unlike for the discrete variables of the previous use case (see Section 6.2.5). Therefore, we fill the interpolation points outside the convex hull using nearest-neighbor estimates. Section 7.4.1 explores which interpolation method performs better for this use case: nearest-neighbor or linear (with nearest-neighbor corrections).

Additionally, for both interpolation methods, we apply the following criteria when sensor observations are scarce:

- No observations available: We use values from the previous timestamp. This situation occurs only once per year, during the daylight saving time shift (see Figure 7.2).
- One to three observations available: Missing sensor values are filled with

non-empty observations from the previous timestamp Then, we achieve four or more observations and interpolate normally.

- Four or more observations available: Interpolation is performed normally.

7.2.6 Phase 6: Preservation and metadata

As in the previous use cases, we use HDF5 format to store the resulting datasets. The final dataset that gathers ten years of observations of the eleven variables has shape (87 648, 35, 30, 11). According to the scaling (normalized, standardized, or raw) and interpolation (nearest-neighbor or linear) methods, the dataset has six variants. We also store the hourly raw observations with no scaling or interpolation to test the developed models. In addition, we store the following metadata: geolocation of the air pollutant sensors, geolocations of the weather sensors, correspondence between sensors and variables, units of the variables, temporal resolution, temporal coverage, sensor identifiers, etc. All these datasets, metadata, and code to reproduce them are publicly available online (Prado-Rujas et al., 2023b, 2023c).

7.2.7 Phase 7: Validation

As in previous use cases, we carried out several checks and validations as we progressed in the data exploration and transformation processes. We made them available online as Jupyter Notebooks for reference (Prado-Rujas et al., 2023c), including:

- Visualization of the sensors' lifespans and the pollutants they observe over time. They helped us fix the temporal coverage and study pollutant variables. For instance, we saw too few observations of three pollutants, which we discarded. Those pollutants were total hydrocarbons (TCH), methane (CH_4), and non-methane hydrocarbons (NMHC).
- Boxplots of the air pollutants by station, hour (see Figs. 7.3 and 7.4), weekday, month, and year.
- Evolution over time of certain pollutants on specific sensors.
- Spatial and temporal coverage of the air pollutants by sensors.
- Analysis of alternative methods for constructing the mesh-grid (see Section 7.2.5).
- Animated air quality visualizations.

7.3 Forecasting model

One way to tackle the air quality forecasting problem is to train independent models for each pollutant, geolocation, and forecast horizon. However, this approach neglects the correlations between contiguous horizons, geolocations, and pollutants. Arguably, a single model that combines all of these dimensions should achieve better results.

This section details the baselines and our air quality forecasting proposal. First, Section 7.3.1 describes several air quality baseline models and their parameters. Afterward, Section 7.3.2 presents our proposal, namely the ST-AQF model. As in Chapter 6, we explore four variants based on the modules that comprise it.

7.3.1 Baselines

This section presents the baseline models to which we compare our proposal. As in previous use cases, we feed them observations from an earlier temporal window, and they provide predictions for several horizons. Besides, they work with mesh-grids in the input and output (as in Chapter 6). Therefore, they yield the $n_y \cdot n_s \cdot n_v$ predictions by reversing the interpolation of the output tensor of shape (n_y, n_r, n_c, n_v). The baseline models are the following:

- **LSTM**: Initially, it flattens the mesh-grid to obtain a vector on every of the n_x input timestamps. Then, these vectors are passed by an LSTM layer with 16 neurons and two fully-connected layers of 32 and $(n_y \cdot n_r \cdot n_c \cdot n_v)$ neurons. The result is reshaped into the predicted tensor of shape n_y, n_r, n_c, n_v.

- **Conv3D**: It passes the n_x input mesh-grids by a 3D-convolutional layer with three filters and kernel size $(8, 8, 8)$. Afterward, it performs a $(1, 4, 4)$ max-pool and flatten its result. This vector is passed through a dense layer of 896 neurons, reshaped, and deconvoluted. The deconvolution has n_v filters with kernel size $(n_y, 4, 3)$ to match the output shape (n_y, n_r, n_c, n_v). Unlike previous baselines, this model considers sensors' geolocations thanks to the convolutional operator.

- **Persistence**: As in the prior use cases, it is a simple model that assumes that the forecasted variable remains unchanged. Therefore, we express it as follows for any horizon:

$$f\left(\left\{\, \mathcal{M}_v^t \mid t = t_{\gamma-(n_x-1)}, \ldots, t_\gamma,\ v \in \mathcal{V} \,\right\}\right) \equiv \mathcal{M}_v^{t_\gamma}.$$

  As discussed, it is a tough contestant for short forecasting horizons.

7.3.2 ST-AQF

This section describes the developed model for air quality forecasting based on the modular methodology detailed in Section 4.2. We already accomplished the

analysis and transformation processes and constructed the air quality mesh-grid in Section 7.2. This data structure fuses the pollutant observations into a single entity that accounts for temporal and spatial correlations. In the following, we describe the spatio-temporal module, following Section 4.2.1. Analogously to the previous use cases, this is the main component of our proposal, the ST-AQF. The steps that the spatio-temporal module performs are as follows:

1. The input to the module is the n_x air quality mesh-grids, as for the baselines.
2. These are fed to the ConvLSTM layer, which produces a three-dimensional tensor.
3. This tensor is then shrunk by a max-pooling layer and flattened, yielding the air quality feature vector.

At this stage, the ST-AQF model concatenates the air quality feature vector with feature vectors from any exogenous module. Then, it continues as follows:

1. The resulting vector is passed by a dense layer and reshaped into a four-dimensional tensor, matching the output dimensions: time, latitude, longitude, and air pollutant.
2. The next step is a deconvolution that yields an output tensor of shape (n_y, n_r, n_c, n_v).
3. The final step is to undertake the opposite conversion from the one that yielded the mesh-grids. In so doing, the model obtains a prediction for every horizon, geolocation, and variable.

In our experiments, the ConvLSTM layer has four filters of kernel size $(4, 3)$ and hyperbolic tangent activation. The max-pooling layer has size $(2, 2)$, and the dense layer has 896 neurons with linear activation. Lastly, the deconvolution has n_v filters, kernel size of shape $(n_y, 4, 3)$, and linear activation. As usual, we evaluate error metrics by applying the corresponding equation on every horizon, sensor, and variable. In this chapter, we calculate the RMSE (Eq. 4.5), nRMSE (Eq. 4.6), and MAE (Eq. 4.7).

Exploration of the data highlights the periodic behavior of the air quality variables (see Section 7.2.1). Additionally, weather conditions influence the air quality observed at a given location, as discussed by Abirami and Chitra (2021), among others. Therefore, we introduced periodical and weather feature extraction modules, following Section 4.2.2. These exogenous modules are as follows:

- **Periodical module**: This flat-extraction module seizes seasonal information from each grid's timestamp. The input variables are not treated as time series but instead are encoded as follows:

- time of the day, transformed using sine and cosine functions to values in the range $[-1, 1]$,
- time of the week, transformed using sine and cosine functions to values in the range $[-1, 1]$,
- time of the year, transformed using sine and cosine functions to values in the range $[-1, 1]$,
- weekday or weekend, encoded as $\{0, 1\}$, and
- holiday status, encoded as $\{0, 1\}$ (obtained from Comunidad de Madrid, 2023).

These encoded features form an input vector that is passed through a dense layer containing 128 neurons. The resulting output vector is concatenated with the flattened output of the ConvLSTM layer halfway through the prediction process.

- **Weather module**: This module processes weather data corresponding to the same time window as the air quality mesh-grid. Since these data are inherently temporal, the module functions as a recurrent one (see Section 4.2.2). As a result, they are fed into a LSTM layer with 128 neurons, which iterates on n_x. The resulting feature vector is then concatenated with the outputs from the spatio-temporal and periodical modules.

As in Chapter 6, we study four variants of our air quality forecasting model based on the modules that comprise it:

- **ST-AQF**: This variant utilizes only the spatio-temporal air quality module described above. It contains 807 147 trainable parameters when configured with $n_x = 4$, $n_r = 35$, $n_c = 30$, $n_y = 4$, and $n_v = 11$.
- **ST-AQF$^{\mathbf{t}}$**: It incorporates the air quality module alongside the periodical module, resulting in 922 987 trainable parameters under the same configuration as the ST-AQF.
- **ST-AQF$^{\mathbf{w}}$**: It combines the air quality module with the weather module, yielding 1 003 243 trainable parameters for the same configuration as the ST-AQF.
- **ST-AQF***: It integrates all three modules (i.e., air quality, periodical, and weather), resulting in 1 119 083 trainable parameters with the same settings as the ST-AQF.

Figure 7.5 shows the ST-AQF* model, which includes all three modules. Furthermore, Prado-Rujas et al. (2023c) gathers plots of the four ST-AQF variants,

including input and output shapes for each layer.

7.4 Experiments and evaluation

This section starts with the experimental setup and hyperparameter tuning in Section 7.4.1. Afterward, Section 7.4.2 compares the error of the ST-AQF to several baselines and models from the literature. Then, we analyze the impact of the temporal window n_x in Section 7.4.3. After that, we explore robustness and flexibility NFFs in Section 7.4.4. Finally, Section 7.4.5 and 7.4.6 study how to incorporate multiple pollutants and exogenous data into the ST-AQF model, respectively. Since the ST-AQF model considers 11 variables, we have limited the number of plots in each experiment, keeping NO_2 in all as a proxy for air pollution. In any case, the graphs for the remaining pollutants can be found in Prado-Rujas et al. (2023c).

7.4.1 Experimental setup

As in previous use cases, we developed all models using the Keras library (Chollet et al., 2015) with the Tensorflow backend (Abadi et al., 2015). The training data covers a decade between 2010 and 2019, as Section 7.2.1 explains. Specifically, we used the years between 2010 and 2016 for training (∼70 %), 2017 for validation (∼10 %), and 2018 and 2019 for testing (∼ 20 %). We employed early stopping based on the validation loss with ten epochs of patience. All the error metrics presented in this experiment section correspond to the test set. We conducted the experiments on three Ubuntu machines, each with 12 cores, 24 logical processors, and three graphic cards (see Table 7.4). The following presents the fine-tuning experiments we used to choose the optimizer, loss function, learning rate, scaling, and interpolation techniques.

Table 7.4: GPUs utilized in the air quality use case experiments.

GPU model	GPU count	Compute capability	Memory
NVIDIA GeForce GTX TITAN X	4	5.2	12 GB
NVIDIA GeForce GTX TITAN Black	2	3.5	6 GB
NVIDIA GeForce GTX TITAN	1	3.5	6 GB
NVIDIA GeForce RTX 2070	1	7.5	8 GB
NVIDIA GeForce RTX 2080	1	7.5	12 GB

Fine-tuning the optimizer, loss function, and learning rate

In this experiment, we trained the models using only historical air quality and not weather or calendar data for 100 epochs. Besides, $n_x = 4$, $l = 1$, $n_y = 4$, $n_v = 11$, and we fixed the scaling and interpolation methods to standardized and

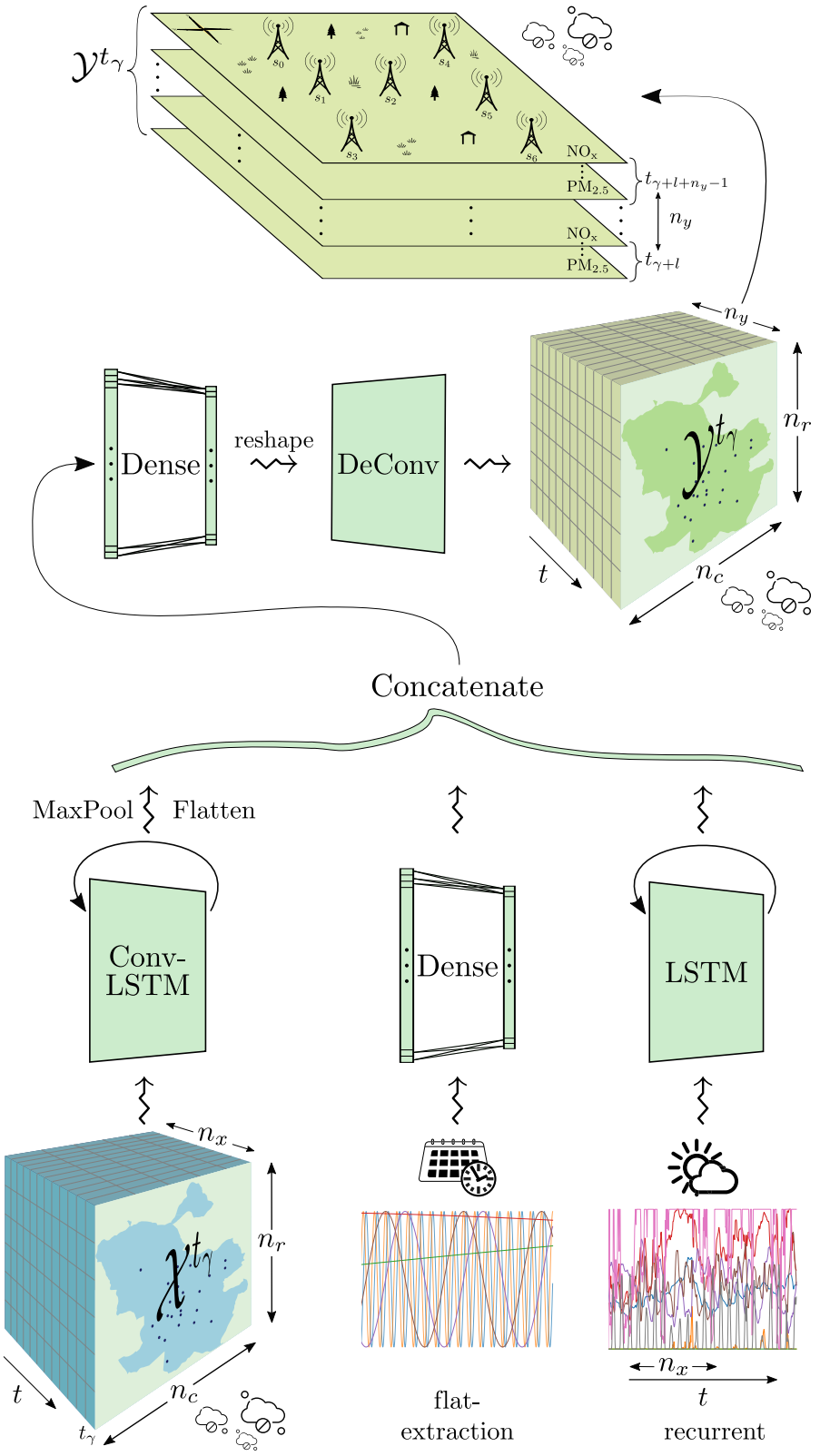


Figure 7.5: Spatio-Temporal Air Quality Forecaster (ST-AQF) model.

nearest-neighbor, respectively. The 96 cases studied derive from:

- 8 optimizers: Adam, Nadam, Adamax, Adadelta, Adagrad, SGD, RMSprop, and Ftrl.
- 3 loss functions: MSE, MAE and MSLE.
- 4 learning rates: 10^{-3}, 10^{-4}, 10^{-5}, $5{\cdot}10^{-6}$.

The optimizers that stood out the most were Adam, Nadam, Adamax, and RMSprop, with Adamax being the most consistent. Furthermore, setting the learning rate at 10^{-3} derived the best results. Experiments indicate that the best loss function is specific to pollutants, obtaining better results both for MAE and MSE. All in all, fixing the optimizer to Adamax and the learning rate to 10^{-3} shows a balance between these two. Specifically, the former obtains an improvement of just 2 % on average (over variables and horizons). Given these facts and to facilitate the reproducibility of the experiments, we chose MSE as the loss function.

Additionally, we explored the best combinations of loss function and optimizer for $n_v = 1$. We fixed the learning rate to 10^{-3} and varied the optimizers and loss function in the same way. In this case, each model works with just one variable. Thus, the number of trained models is 11 times higher. These 264 additional models provided conclusions very similar to those obtained in the case of $n_v = 11$. Therefore, we used the Adamax algorithm with a learning rate of 10^{-3} to optimize the MSE loss function in the training phase for all trained models. The full table of results that includes different metrics (RMSE, nRMSE, MAE) is available in Prado-Rujas et al. (2023c) for completeness.

Fine-tuning the scaling and interpolation methods

In this experiment, we studied what combination of scaling and interpolation methods derive the best models. As in the previous analysis, we trained them without weather or calendar data for 100 epochs. Besides, $n_x = 4$, $l = 1$, $n_y = 4$, and we fixed the optimizer, loss function, and learning rate to Adamax, MSE, and 10^{-3}, as previously discussed. The scaling methods we studied were raw (without conversion), standardized (see Eq. 4.2), and normalized (see Eq. 4.1). Regarding the interpolation methods, we considered nearest-neighbor and linear interpolation over the ROI. These six datasets are available on Zenodo (Prado-Rujas et al., 2023b).

In the case of $n_v = 1$, we trained 66 models, given that each gas is modeled separately. The results show that normalization is the worst scaling method of the three studied. Differently, standardizing is the best and most consistent. It usually scores the smallest nRMSE (or follows raw scaling very closely in other cases). Nearest-neighbor interpolation behaves better in almost every case if we fix standardization as the scaling method. However, looking closer, we can see that the

difference is minor. Specifically, it is 1.8 % better over variables and horizons on average, having minimum and maximum values of -2.2 % and 8.4 %, respectively. Therefore, standardization with nearest-neighbor interpolation is our choice for $n_v = 1$. The upper part of Table 7.5 shows the nRMSE scored by the ST-AQF model with these parameters.

Regarding $n_v = 11$, we trained six models since each considers all 11 variables simultaneously. The results were similar for the scaling method, but linear interpolation behaves better in this case. However, again, this improvement is small (2.3 % on average, with minimum and maximum values of -1.1 % and 7 %). The lower part of Table 7.5 shows the nRMSE scored by the ST-AQF model with these parameters. Nearest-neighbor interpolation has some benefits compared to linear interpolation because of its nature. These advantages include faster calculation times and fewer memory requirements. Therefore, considering the slight improvement, we use nearest-neighbor over linear interpolation for the remaining chapter. As mentioned, Prado-Rujas et al. (2023c) hosts the complete results table.

Table 7.5: Error metrics of the ST-AQF model with $n_v = 1$ and $n_v = 11$. On both cases, we fix the best combination of scaling and interpolation methods.

Scaling	**Interp.**	n_v	**Variable**	**Median nRMSE** [-] **across** $\mathcal{S}_v$ 1 h	2 h	3 h	4 h	**Mean** [-]
Standardized	Nearest-neighbor	1	$PM_{2.5}$	0.454	0.525	0.557	0.582	0.530
			PM_{10}	0.422	0.511	0.563	0.600	0.524
			O_3	0.234	0.279	0.357	0.430	0.325
			NO	1.157	1.329	1.579	1.804	1.467
			NO_2	0.338	0.433	0.544	0.633	0.487
			NO_x	0.617	0.725	0.873	1.007	0.806
			SO_2	0.217	0.252	0.290	0.312	0.268
			CO	0.350	0.413	0.488	0.552	0.451
			TOL	0.922	1.127	1.245	1.329	1.156
			BEN	0.639	0.777	0.892	0.969	0.819
			EBE	1.533	1.643	1.717	1.786	1.670
	Linear	11	$PM_{2.5}$	0.646	0.638	0.642	0.660	0.647
			PM_{10}	0.719	0.721	0.731	0.743	0.728
			O_3	0.527	0.537	0.555	0.570	0.547
			NO	1.543	1.574	1.712	1.863	1.673
			NO_2	0.478	0.529	0.601	0.661	0.567
			NO_x	0.745	0.824	0.942	1.043	0.888
			SO_2	0.489	0.489	0.500	0.513	0.498
			CO	0.474	0.496	0.537	0.586	0.523
			TOL	1.312	1.290	1.306	1.357	1.316
			BEN	1.148	1.139	1.163	1.182	1.158
			EBE	2.151	2.150	2.150	2.151	2.150

The training time is almost 1 h for all models with $n_v = 1$ and 2 h 20 min for the one model with $n_v = 11$ (for 100 epochs). We obtain the nRMSE per horizon as the median across the available sensors for each variable. The last column shows the mean of these median values.

Summary of the fine-tuning results

In the following, we recap the main results of the fine-tuning experiments:

- Adamax algorithm with a learning rate of 10^{-3} is used to optimize the MSE loss function, both for $n_v = 1$ and $n_v = 11$.
- The scaling method that yields the best results is standardization for both n_v values.
- The nearest-neighbor interpolation method behaves better for $n_v = 1$, whereas linear interpolation is more suited for $n_v = 11$. All in all, we choose nearest-neighbor over linear interpolation due to its benefits.
- Table 7.5 shows that the nRMSE for $n_v = 1$ is better than the one for $n_v = 11$, for all variables and horizons. Section 7.4.5 will further explore this.

Hence, we now list the parameters used in the subsequent experiments:

- **Variables**: We used between one and eleven variables, so $n_v \in [1, 11]$ as discussed in Section 7.1.2.
- **Temporal granularity**: $\tau = 1$ hour, as discussed in Section 7.2.2.
- **Models**: The modules and parameters of the baseline and ST-AQF models were already discussed in Sections 7.3.1 and 7.3.2, respectively.
- **Number of input timestamps**: $n_x \in \{1, 2, 3, 4, 8, 12, 16, 20, 24\}$, corresponding to hours.
- **Forecast horizons**: Horizons are between one and four hours, with $n_y = 4$.
- **Number of sensors**: There are 24 sensors in total, which observe a different number of variables (see Table 7.1).
- **Mesh-grid shape**: 35×30.
- **Optimizers**: Adamax.
- **Learning rate**: 10^{-3}.
- **Loss function**: MSE.
- **Batch size**: One week of observations, thus 168 (given that $\tau = 1$ h).
- **Number of epocs**: 100.

7.4.2 Error metrics

This section compares the error of the ST-AQF with the baseline models and the literature for a single pollutant (i.e., $n_v = 1$). To this aim, we calculate error metrics for every forecast horizon, sensor, and air pollutant. We obtain the median across sensors for each horizon and variable to summarize these values. Afterward, we compute the mean across horizons, yielding one error metric per variable and model. In this section, depending on the experiment, we display RMSE or nRMSE. Nevertheless, the MAE is also included in Prado-Rujas et al. (2023c) for completeness.

Comparison with baselines

To begin with the error experiments, we compare the ST-AQF to the baselines described in Section 7.3.1. As mentioned earlier, the models from this experiment consider a single variable for a more transparent comparison. Nevertheless, they could work with $n_v > 1$ (as seen in Section 7.3.1).

As in Section 7.4.1, we fix $n_x = 4$, $l = 1$, $n_y = 4$. Table 7.6 shows the nRMSE scored per model, variable, and horizon. The best model for every variable and horizon appears highlighted in bold. The table shows the median nRMSE across the available sensors for every variable and horizon. The last column displays the mean of those n_y nRMSE values. In addition, the table displays training times considering early stopping, as discussed in Section 7.4.1. Table 7.6 reveals that the nRMSE degrades as the forecast horizon moves forward in time. This behavior can be caused by the accumulation of errors in the prediction. In other words, forecasts for four hours ahead are more unstable than those for one hour.

To complement Table 7.6, we conducted the Almost Stochastic Order (ASO) test (Del Barrio et al., 2018; Dror et al., 2019) as implemented by Ulmer et al. (2022). Table 7.7 presents its results for the daily RMSE scored on the test set (i.e., over 700 days for each horizon). The table compares the ST-AQF model with $n_v = 1$ to the three baseline models for every pollutant with a confidence level $\alpha = 0.05$. Table 7.7 shows that the ST-AQF is stochastically dominant over the persistence and LSTM models ($\varepsilon_{\text{min}} < 0.5$) in practically every case. The differences are less clear-cut for the Conv3D, where $\varepsilon_{\text{min}} < 0.5$ for three pollutants and $\varepsilon_{\text{min}} \geq 0.5$ for the other eight. This circumstance can be expected since the only difference between the ST-AQF and the Conv3D baseline is replacing the ConvLSTM layer with a 3D-convolution. In any case, Table 7.6 shows that their error metrics are within the same range.

Comparison with the literature

As discussed in Section 6.4.2, direct comparison of multivariable spatio-temporal models is usually tricky. Differences in data sources, forecast horizons, timestep, and evaluation metrics make it complex. In addition, many works do not disclose

Table 7.6: Error metrics of the ST-AQF and baseline models for $n_v = 1$.

Variable	Model	Training time	Median nRMSE [-] across $\mathcal{S}_v$ 1 h	2 h	3 h	4 h	Mean [-]
$PM_{2.5}$	ST-AQF	17	0.462	**0.521**	**0.560**	0.593	**0.534**
	Persistence	-	**0.459**	0.579	0.632	0.683	0.588
	Conv3D	31	0.466	0.531	0.562	**0.583**	0.536
	LSTM	6	0.484	0.536	0.564	0.602	0.547
PM_{10}	ST-AQF	56	0.435	**0.516**	0.563	**0.601**	**0.529**
	Persistence	-	**0.422**	0.553	0.633	0.693	0.575
	Conv3D	31	0.437	0.520	**0.561**	0.603	0.530
	LSTM	8	0.466	0.536	0.585	0.626	0.553
O_3	ST-AQF	56	0.225	**0.272**	**0.358**	**0.433**	**0.322**
	Persistence	-	0.221	0.368	0.481	0.568	0.409
	Conv3D	31	0.245	0.285	0.360	0.435	0.331
	LSTM	17	**0.218**	0.299	0.374	**0.433**	0.331
NO	ST-AQF	56	1.166	**1.319**	**1.590**	1.819	**1.474**
	Persistence	-	1.176	1.766	2.156	2.397	1.874
	Conv3D	31	**1.162**	1.337	1.611	1.823	1.483
	LSTM	19	1.172	1.433	1.621	**1.745**	1.493
NO_2	ST-AQF	54	0.340	**0.423**	**0.539**	0.632	**0.484**
	Persistence	-	0.362	0.580	0.730	0.830	0.625
	Conv3D	31	**0.334**	0.436	0.549	0.641	0.490
	LSTM	20	0.339	0.466	0.562	**0.626**	0.498
NO_x	ST-AQF	56	0.612	**0.714**	**0.875**	**1.009**	**0.803**
	Persistence	-	0.635	0.972	1.202	1.345	1.038
	Conv3D	31	**0.598**	0.729	0.885	1.017	0.807
	LSTM	20	0.602	0.770	0.890	0.971	0.808
SO_2	ST-AQF	52	0.205	0.249	0.279	**0.304**	0.259
	Persistence	-	**0.160**	**0.231**	**0.278**	0.311	**0.245**
	Conv3D	10	0.241	0.264	0.292	0.315	0.278
	LSTM	17	0.329	0.352	0.373	0.397	0.363
CO	ST-AQF	54	0.354	0.424	0.500	0.560	0.459
	Persistence	-	**0.329**	0.499	0.612	0.685	0.531
	Conv3D	31	0.340	**0.400**	**0.480**	**0.540**	**0.440**
	LSTM	19	0.377	0.465	0.558	0.623	0.506
TOL	ST-AQF	56	**0.927**	**1.125**	**1.247**	**1.329**	**1.157**
	Persistence	-	0.960	1.314	1.523	1.642	1.360
	Conv3D	31	0.971	1.146	1.266	1.341	1.181
	LSTM	20	1.051	1.183	1.298	1.359	1.223
BEN	ST-AQF	55	0.672	0.812	0.919	0.989	0.848
	Persistence	-	**0.630**	0.897	1.068	1.172	0.942
	Conv3D	31	0.660	**0.795**	**0.906**	**0.981**	**0.835**
	LSTM	13	0.754	0.883	0.977	1.030	0.911
EBE	ST-AQF	54	**1.504**	**1.626**	1.692	1.754	**1.644**
	Persistence	-	1.687	1.986	2.076	2.172	1.980
	Conv3D	26	1.528	1.629	**1.684**	**1.750**	1.648
	LSTM	19	1.709	1.817	1.869	1.915	1.828

Training time is displayed in minutes (accounting early stopping). The nRMSE per horizon is obtained as the median across the available sensors per variable, and the mean of these median values is displayed in the last column.

Table 7.7: ASO test comparing the ST-AQF to the baselines for $n_v = 1$.

Variable	ST-AQF vs. persistence	ST-AQF vs. Conv3D	ST-AQF vs. LSTM
$PM_{2.5}$	**0.011**	1.000	**0.160**
PM_{10}	**0.048**	0.898	**0.030**
O_3	**0.006**	0.603	**0.068**
NO	**0.004**	0.684	**0.096**
NO_2	**0.001**	0.755	**0.275**
NO_x	**0.002**	0.562	**0.310**
SO_2	1.000	**0.006**	**0.000**
CO	**0.016**	0.733	**0.001**
TOL	**0.086**	**0.343**	**0.017**
BEN	**0.230**	1.000	**0.000**
EBE	0.982	**0.204**	**0.002**

Bold format is applied when $\varepsilon_{\text{min}} < 0.5$, which expresses an upper bound to the amount of violation of stochastic order (see Ulmer et al., 2022). The experiment is available at Prado-Rujas et al. (2023c).

their code or trained models. Nevertheless, we looked for related work that could be comparable to ours to some extent.

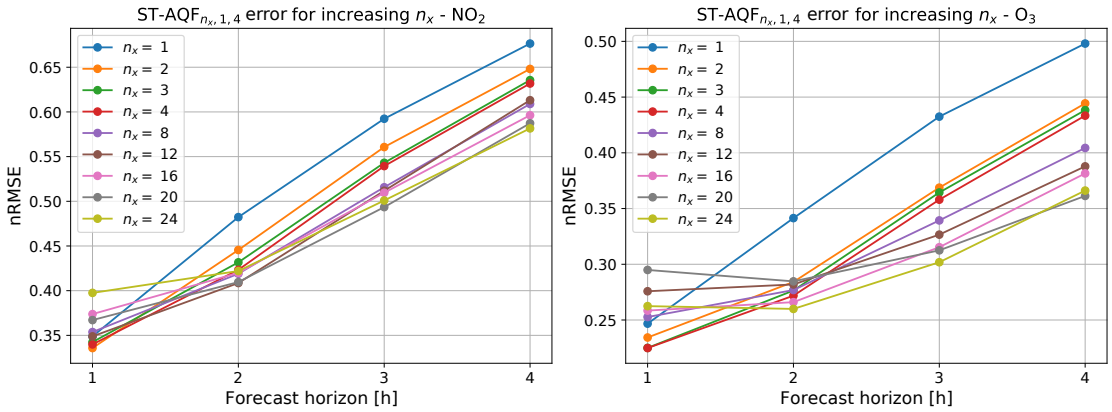
Aznarte (2017) and García and Aznarte (2020) forecast 1-hour ahead NO_2 measurements, each at one location in Madrid. Specifically, the best RMSE reported by Aznarte (2017) for station 4 (see Fig. 7.1) is 8.1 % better than the one scored by the ST-AQF model for the same station and horizon. As for García and Aznarte (2020), the ST-AQF obtains 33.3 % better RMSE for station 8 (see Fig. 7.1).

The more recent work of de Medrano et al. (2021) models four pollutants in the city of Madrid. We discussed it in Section 2.4.2. However, they do not share the model for intellectual property reasons. Furthermore, they forecast for the next 48 hours, and hence, a straightforward comparison is not possible. However, it is a proxy to show that error metrics are akin. Specifically, the RMSE scored by the ST-AQF is 8.1 % better for PM_{10}, and 5.6 %, 18.3 %, and 7.3 % worse for $PM_{2.5}$, NO_2 and O_3, respectively. This work does not consider any of the remaining seven air pollutants we study.

7.4.3 Impact of temporal window width

Similarly to Section 6.4.3, we experiment to understand the impact of the input window size n_x. As for the other parameters, we set $l = 1, n_y = 4$, and $n_v = 1$. Figure 7.6 compares the nRMSE of the same ST-AQF model with increasing values of n_x. We study two pollutants: NO_2 on the left and O_3 on the right-hand side. The impact can be seen on every horizon, obtained as the median of the nRMSE over the available sensors. The n_x parameter takes values between 1 and 24 hours.

The results for the other pollutants are similar, obtaining smaller nRMSE for higher values of n_x. However, fixing higher n_x leads to an increase in the training times. Thus, we choose to conduct the remaining experiments using $n_x = 4$ as a compromise between model performance and computational load.



(a) Results for NO_2. **(b)** Results for O_3.

Figure 7.6: Impact of the temporal window width on the ST-AQF error. The model is evaluated for $n_x \in \{1, 2, 3, 4, 8, 12, 16, 20, 24\}$. Units of n_x, l, and n_y are [h]. The same experiment for the remaining pollutants can be found in Prado-Rujas et al. (2023c).

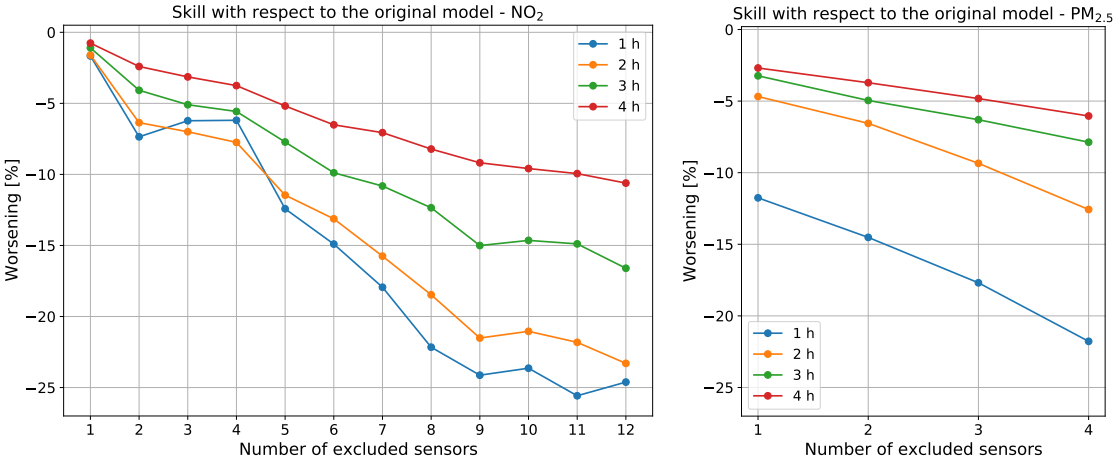
7.4.4 Flexibility and robustness

Once the air quality sensors are operational, they stream a continuous flow of measurements to a central server. Unavoidably, some of those observations are invalid or empty due to hardware breakdowns or external factors. In addition, some sensors may become obsolete and be removed from the network, and some additional ones may be installed. In this vein, this section studies the robustness and flexibility NFFs of the ST-AQF model.

First, we evaluate robustness by simulating sensor failure as described in Section 4.3.3. For this, we investigate how the error of the test set degrades when a growing number of sensors fail. We conducted this experiment for individual pollutants ($n_v = 1$) to isolate the effects of missing information. Besides, we fix the usual parameters: $n_x = 4$, $l = 1$, and $n_y = 4$. The result of the experiment is a tensor of RMSE values of shape $(\lceil n_s/2 \rceil, r, n_y, n_s)$, where r is the number of repetitions. To compare it to the original RMSE, we apply the skill criterion (Eq. 4.9) for each repetition (on every horizon and sensor).

Figure 7.7 shows the results of the experiment for NO_2 and $PM_{2.5}$. The maximum number of sensors n_s is 24 and 7, respectively (see Table 7.1). Also, we fix r at 20. To present the results, we averaged the skill across repetitions and calculated the median across the sensors. Figure 7.7a shows the results for NO_2. The figure shows that RMSE degrades, on average, less than 15 % on the four horizons when 25 % (or less) of the sensors are not working. When that number increases to

half of them (12 sensors), the worsening stays around or below 25 %. Similarly, Figure 7.7b presents the results for $PM_{2.5}$. In this case, the RMSE degradation also stays below 25 % when more than half of the sensors do not send information. With fewer sensors, the experiment for $PM_{2.5}$ presents higher stability than that of NO_2. In both cases, the RMSE degradation is almost ordered by the horizon. In other words, further horizons are less affected by missing data than the closest ones. Interestingly, the 1-hour ahead forecast degrades slightly less than the 2-hour one for 3 and 4 missing sensors in the case of NO_2. The random nature of the experiment could cause this behavior, and a higher number of repetitions could smooth it out.



(a) Results for NO_2. **(b)** Results for $PM_{2.5}$.

Figure 7.7: Robustness test results for the ST-AQF for two pollutants. The baseline considers all sensors are active (24 for NO_2 and 7 for $PM_{2.5}$). The same experiment for the remaining pollutants can be found in Prado-Rujas et al. (2023c).

Flexibility is inherent to the ST-AQF model thanks to the integration of the mesh-grid. Time series at the sensor level are incorporated into the fixed-size mesh-grid, as explained in Section 7.2.5. This structure enables the model to accommodate a variable number of sensors in the input and output. Furthermore, the robustness experiment shows this in practice. Section 7.4.2 presents another practical example, specifically for ST-AQF models where $n_v = 1$. For each pollutant, the model employs a variable number of sensors without changing the framework's structure.

As discussed in Section 4.1.5, the distribution and number of sensors condition the grid shape and, therefore, the granularity of the mesh-grid. For a given n_r and n_c, the maximum number of sensors that the grid should consider is $n_r \cdot n_c$ to prevent observation loss. If we need to increase n_s beyond this limit, the spatial granularity σ should be set to a finer value.

7.4.5 Extensibility

Sections 7.4.2, 7.4.3, and 7.4.4 explore different properties of the ST-AQF model, always with $n_v = 1$. So far, we only considered $n_v = 11$ on the hyperparameter tuning (Section 7.4.1). This section studies how the ST-AQF model behaves when progressively incorporating variables. We refer to this property as extensibility, as discussed in Section 3.3.3. To this end, we fix a variable and train several ST-AQF models while progressively incorporating more variables:

- $n_v = 1$: Initially, we train a model for NO_2 and another one for PM_{10}.
- $n_v = 2$: Then, we train another ST-AQF model for both of them.
- $n_v = 4$: Afterward, we incorporate SO_2 and CO.
- $n_v = 6$: We include NO and $PM_{2.5}$.
- $n_v = 8$: We add NO_x and O_3.
- $n_v = 11$: We integrate the remaining ones (TOL, BEN, and EBE).

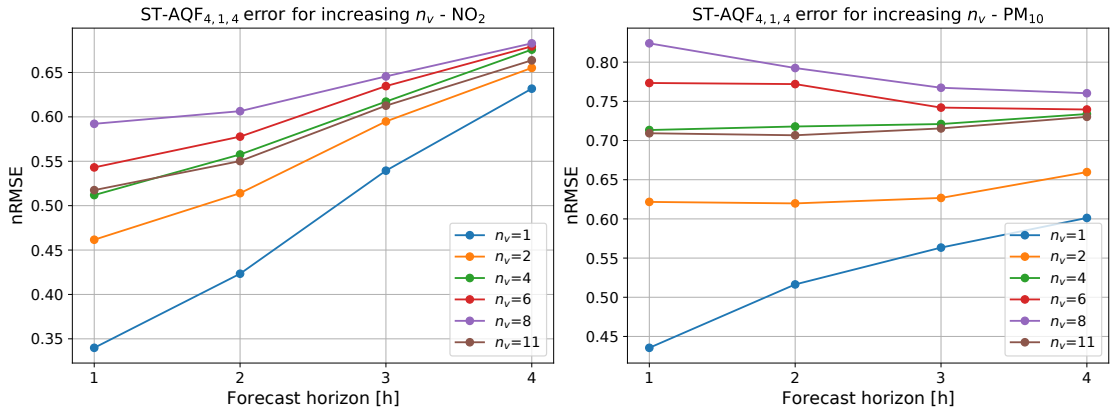
The aim is to study whether the model benefits from observing multiple variables simultaneously. Additionally, we seek to determine if there are differences between distinct values of n_v. As usual, we fix the parameters: $n_x = 4, l = 1$, and $n_y = 4$.

Evaluation for several n_v

Figure 7.8 presents the results of this experiment for NO_2 (Fig. 7.8a) and PM_{10} (Fig. 7.8b). The figure represents the nRMSE scored on each forecast horizon by every model. For both pollutants, the model with $n_v = 1$ scores the lowest error, which increases for every consecutive value of n_v on all horizons. However, results show that $n_v = 11$ stands between $n_v = 2$ and $n_v = 4$, being very close to the latter. Admittedly, this experiment merely serves as a proxy of the full-length one (all combinations of incorporating variables one at a time). Considering a single pollutant, the total number of models to explore would be the factorial of 10, more than 3.5 million cases. Furthermore, this figure increases to almost 40 million when considering the 11 pollutants. Figure 7.8 presents a small subset of this experiment. However, it indicates that a careful combination of multiple variables could improve the forecasting results of separate models. Nevertheless, we should conduct a sensitivity analysis to understand whether some pollutants lower the prediction error of others.

Fine-grained evaluation

In the previous section and in Figure 7.8, we study extensibility by grouping results with the median across sensors. Here, we compare the cases $n_v = 1$ and $n_v = 11$ at the sensor and horizon levels in Figure 7.9. As before, Figure 7.9a refers



(a) Results for NO_2. **(b)** Results for PM_{10}.

Figure 7.8: Extensibility test results for the ST-AQF. The model is evaluated for two different pollutants when $n_v \in \{1, 2, 4, 6, 8, 11\}$, i.e., progressively increasing the number of variables considered in the input and output.

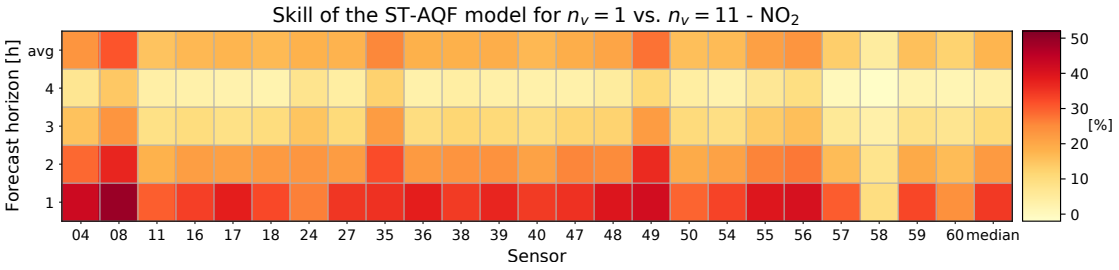
to NO_2 and Figure 7.9b to PM_{10}. The figures show the improvement for the case $n_v = 1$ when compared to $n_v = 11$ using the skill criterion (Eq. 4.9). We calculate this metric using the nRMSE of the model with $n_v = 11$ as reference and the one of $n_v = 1$ as the proposed one. Results show that the skill is greater than 30 % or even 40 % for the 1-hour ahead forecasts. However, it diminishes for further horizons. The skill is negative only once out of the 148 cases shown in Figure 7.9. This skill is -1.3 %, and it appears in Figure 7.9a for sensor 58 on the 4-hours ahead forecast. However, there are meaningful differences between sensors. These variations can be caused by the specific location or weather particularities (i.e., being close to a park, being more or less exposed to wind, etc.).

Further horizons

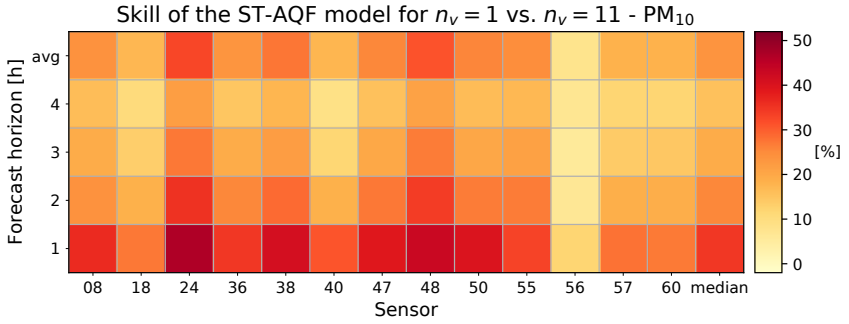
Table 7.5 presents the nRMSE of ST-AQF for $n_v \in \{1, 11\}$ (see Section 7.4.1). For both n_v values, we choose the best combination of scaling and interpolation methods. We can also study this table from a different angle: comparing the nRMSE of the two n_v values for each pollutant and horizon. As seen in this section, the table highlights that a model with $n_v = 1$ outperforms the model with $n_v = 11$ for every pollutant. We can expect this pattern since the model architecture is unchanged, but the number of variables to predict is 11 times higher. However, Table 7.5 also shows that the improvement is monotonically decreasing on the horizons for all variables. Therefore, this leads us to think that models with $n_v = 1$ and $n_v = 11$ should score akin nRMSE values for horizons further than four hours.

7.4.6 Integrability

As in Section 6.4.6, we now evaluate the integrability NFF of the ST-AQF model. For this, we conduct an ablation study on the modules that constitute it.



(a) Results for NO_2.



(b) Results for PM_{10}.

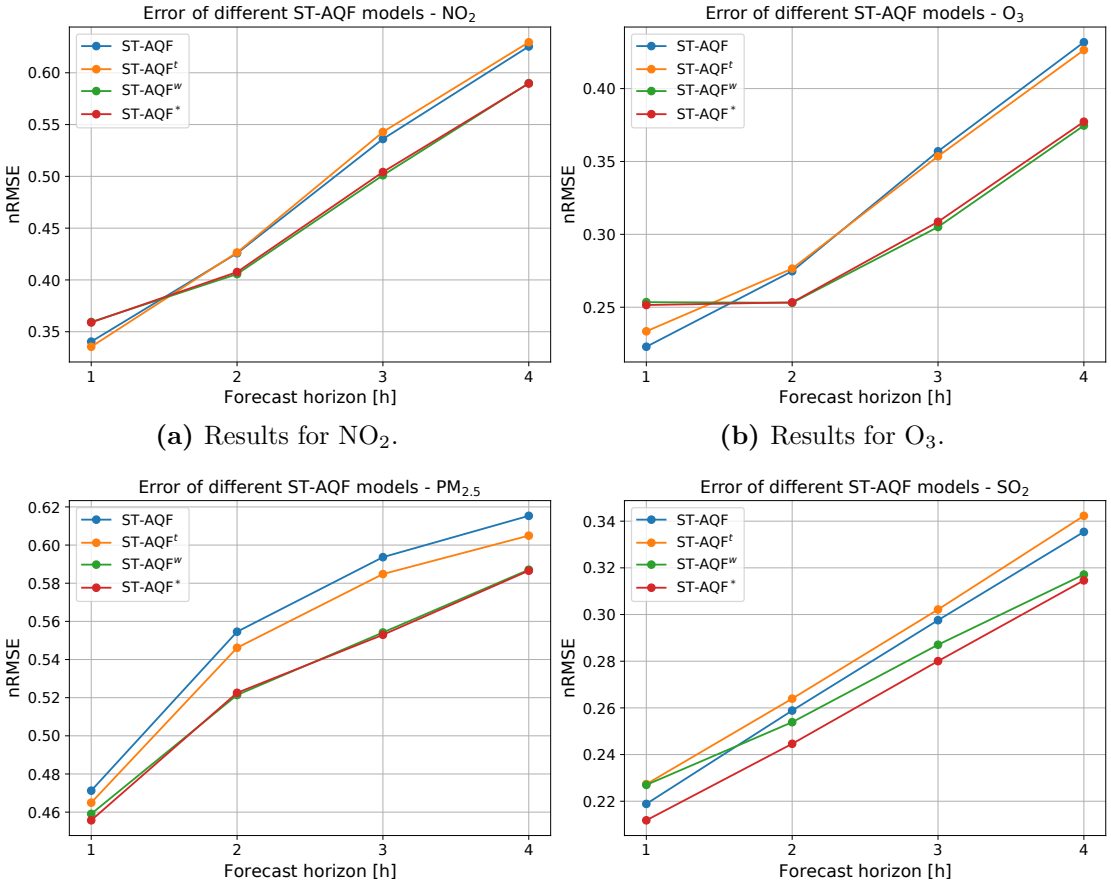
Figure 7.9: Fine-grained evaluation of the extensibility of the ST-AQF model for two pollutants. The skill compares $n_v = 1$ to $n_v = 11$ at sensor and horizon levels, including the median per horizon across sensors and the mean per sensor across horizons. The same experiment for the remaining pollutants can be found in Prado-Rujas et al. (2023c).

Section 7.3.2 described four variations of our proposed model, namely: ST-AQF, ST-AQF^{t}, ST-AQF^{w}, and ST-AQF*. Figure 7.10 compares the four models for four pollutants: NO_2, O_3, $PM_{2.5}$ and SO_2. In all cases, the parameters are $n_x = 4, l = 1$, and $n_y = 4$. Besides, we set n_v to one to isolate the effects of incorporating exogenous data.

Figure 7.10a presents the results for NO_2. The ST-AQF* and ST-AQF^{w} obtain less error than their peers for forecasts of 2, 3, and 4 hours. Applying Equation 4.9, it achieves a skill over 6.5 % for the third horizon, compared to ST-AQF. Results for ST-AQF are very similar to those of ST-AQF^{t}. The same thing happens for ST-AQF^{w} and ST-AQF*. Therefore, the periodical module does not seem to provide much helpful information when predicting NO_2. However, the weather module is improving the results of ST-AQF and ST-AQF^{t} on most horizons. The case of O_3 (Figure 7.10b) is similar. However, it presents the particularity that ST-AQF* reaches a skill of almost 15 % compared to the ST-AQF on the third horizon.

Figure 7.10c show the results for $PM_{2.5}$. They are similar to the previous cases but also improve in the 1-hour ahead forecast. The skills of the ST-AQF^{w} and ST-AQF* models are close to or above 5 % for the three further horizons, compared to ST-AQF. Finally, the SO_2 presents a more stable behavior. In this

case, the periodical module is providing some information to the ST-AQF* model, as Figure 7.10d shows. All in all, results show that we should base our module's choice on the variable under study. Nevertheless, the weather module provides the most significant improvement to the ST-AQF model.



(a) Results for NO_2.

(b) Results for O_3.

(c) Results for $PM_{2.5}$.

(d) Results for SO_2.

Figure 7.10: Integrability test results for the ST-AQF model for four pollutants. Results for the remaining pollutants can be found in Prado-Rujas et al. (2023c).

7.5 Summary and discussion of results

This chapter introduces a model for spatio-temporal air quality forecasting. Our proposal predicts up to eleven different pollutants measured in Madrid. Furthermore, the chapter evaluates the model from different perspectives to study the benefits it can provide beyond scoring low error metrics. For this purpose, we first apply our data analysis and transformation methodology in Section 7.2. Afterward, we describe our proposed ST-AQF and baselines in Section 7.3.

Once we have described the use case and model, we evaluate its effectiveness. Initially, Section 7.4.1 studies several mesh-grid designs, data scaling techniques,

and hyperparameters. This analysis allowed many alternatives to be left out and to detect which ones were the most consistent across pollutants. We fix the Adamax algorithm with a learning rate of 10^{-3} to optimize the MSE loss function. Additionally, we choose standardization and nearest-neighbor as scaling and interpolation methods, respectively.

Before studying NFFs for this use case, we study the error metrics of the ST-AQF in Section 7.4.2. Table 7.6 compares the nRMSE of the ST-AQF to the one of the baseline models. The baselines that we devised for this use case are persistence, Conv3D, and LSTM (see Section 7.3.1). The table shows that the ST-AQF obtains better results than its peers on most pollutants and horizons. As in Chapter 6, we additionally look for related work when contrasting predictive capabilities. This comparison is usually not straightforward, as discussed in Section 7.4.2. Nevertheless, we choose three related cases and show that their error metrics resembles that of the ST-AQF. We also study the impact of the n_x parameter on the error (see Section 7.4.3). Results show that $n_x \in \{2, 3, 4\}$ hours yields better short-term forecasts (i.e., for one hour). Further horizons require bigger n_x, although they come with a computational penalty. Therefore, we choose $n_x = 4$ as a balance between the two.

In the scenario studied, robustness and flexibility NFFs are beneficial for the operational aspects of the framework. Section 7.4.4 studies how the error evolves when a growing number of sensors fail to provide pollutant measurements. To test this, we exclude up to half of the available sensors from the input mesh-grid for two air pollutants. Results show a degradation of 25 % or less both for NO_2 and $PM_{2.5}$, as Figure 7.7 reveals. This experiment indicates that the ST-AQF is robust and flexible concerning sensor failure.

Subsequently, Section 7.4.5 studies the ability of the ST-AQF model to incorporate pollutants, i.e., the extensibility NFF. Even if it obtains the best results for $n_v = 1$, the difference decreases as the forecast horizon moves away (see Fig. 7.8). Therefore, this suggests that the error for $n_v = 1$ and $n_v = 11$ may be comparable for horizons beyond four hours. We also study this behavior at the sensor level and come to the same conclusion (see Fig. 7.9).

Section 7.4.6 closes the experiments section by studying the integrability NFF. To this end, we conducted an ablation study comparing four different variants of the ST-AQF model. The experiment aims to discern which modules improve the forecast error. On the basis of the results, the weather module is more important for air quality forecasting than the periodical one. Furthermore, the model that incorporates all modules obtains the smallest nRMSE for most of the pollutants and horizons studied. Figure 7.10 presents these results.

Even if the proposed system incorporates several NFFs, we should note some limitations. The mesh-grids necessary to cover the entire ROI are obtained by

interpolation from known data, i.e., actual measured data. These estimates of air pollution are reasonable, although they have not been acquired with instruments. The only solution to improve this limitation is incorporating more sensors into the network. This step is straightforward thanks to the flexibility of the ST-AQF. Despite the model's ability to withstand the malfunction of multiple sensors, if too many of them fail, the system as a whole is not feasible. In particular, the model needs at least three operational measurements to continue forecasting future air quality. A way to avoid this pitfall is to replace the faulty instruments, keeping sufficient sensors working.

CHAPTER 8

Conclusions and Future Work

This chapter summarizes the key findings of this research. First, we highlight its contributions to the field in Section 8.1, addressing the objectives set in Chapter 1. Additionally, we discuss the limitations encountered during the study, providing insights into areas where improvements or further investigations are pertinent. Finally, we outline potential future research directions in Section 8.2, proposing several research questions that could further expand or refine this work.

8.1 Conclusions

This section presents the main conclusions of the thesis. First, we recap the findings from the three studied use cases in Sections 8.1.1, 8.1.2, and 8.1.3. Following that, we provide general conclusions and discuss the limitations of our methodology in Section 8.1.4.

8.1.1 Solar irradiance

In Chapter 5, we apply the methodology described in Chapter 4 to the task of solar irradiance forecasting. To achieve this, we introduce the Spatio-Temporal Solar Irradiance Forecaster (ST-SIF) model in Section 5.3.2, which is not tied to any specific set of sensors. Instead, the ST-SIF model constitutes a flexible system capable of recovering from missing sensor observations. This is possible by leveraging information from neighboring stations without requiring a model redesign or retraining. The key contributions of this chapter are:

- Integration of spatial and temporal solar irradiance features into a unified mesh-grid structure.
- Ability to process multiple sensor observations from different timestamps as both input and output.

- Competitive error metrics compared to baseline models.
- Flexibility in the model interface, allowing the number and distribution of sensors to vary without altering the model's structure (as long as they lie within the spatial coverage).
- Robustness with respect to sensor failures, enabling the model to continue forecasting irradiance across the entire spatial coverage even when certain sensors temporarily fail.

Section 5.4 presents the evaluation of the ST-SIF model. In the experiments, we predict Global Horizontal Irradiance (GHI) (Eq. 5.1) using the persistence (Eq. 5.3) and ST-SIF models. Our model achieves skill scores ranging from 19.2 to 32.2 % (Eq. 4.9) across all forecast horizons for two mesh-grid configurations. As discussed throughout this thesis, obtaining reliable error metrics is a critical requirement. The experiments reveal that the error is conditioned by location for the shortest forecast horizon (one minute). This analysis can help guide network operators when deciding on the optimal layout of solar irradiance sensors. We also conducted experiments to assess the flexibility and robustness Non-Functional Features (NFFs) of the proposed model by simulating random sensor failures and recoveries over time. The results show that the model maintains reasonable forecasting error under these stress tests, which is a valuable asset for real-world applications. These NFFs are a result of integrating solar irradiance mesh-grids into the ST-SIF model.

Chapter 5 explores the error, flexibility, and robustness of the ST-SIF model. These features reduce risks on Photovoltaic (PV) plant energy forecasting by ensuring that the model's predictions remain reliable even when multiple sensors malfunction. As a result, the ST-SIF model enhances PV plant operations, providing stable performance despite sensor failures.

8.1.2 Mobility demand

Chapter 6 builds on the solar irradiance use case by expanding the application of our methodology to the mobility demand forecasting domain and exploring additional NFFs. In this context, we propose an extension to the mesh-grid that effectively captures the necessary information to model transport demand. To support this, we introduce the Spatio-Temporal Mobility Demand Forecaster (ST-MDF) model, which combines and forecasts two mobility demand variables (taxi trips and bicycle rides), integrating weather and temporal features. The key contributions of this chapter are:

- Refining the mesh-grid structure to incorporate multiple variables (two in this use case), enabling the exploration of the extensibility NFF.
- Modular design to accommodate exogenous datasets that can enhance the forecasting error (i.e., integrability).

- Achieving lower error metrics compared to baseline models and existing approaches in the literature.
- Despite these enhancements, the ST-MDF model continues to provide spatio-temporal forecasts while maintaining flexibility and robustness.

Section 6.4 summarizes the experiments conducted on the ST-MDF model. First, the model achieves comparable or better error metrics than the four proposed baselines. Additionally, we compare the Root-Mean-Squared Error (RMSE) of the ST-MDF with several models from the literature, obtaining similar or superior results. As discussed in Section 6.4.2, such comparisons can be challenging due to the various factors influencing the forecasting problem. Therefore, these analyses should be interpreted with caution and used primarily as a benchmark. Furthermore, as in Chapter 5, we examine the flexibility and robustness of the model, observing consistent performance despite forecasting two variables simultaneously. A crucial outcome of Chapter 6 is the extensibility of the ST-MDF model (Section 6.4.5). This NFF suggests that combining related variables into a unified model can improve forecasts compared to independent models. Experiments also indicate that the model benefits from integrating exogenous information. The modular structure of the ST-MDF facilitates ablation studies, which highlight the importance of the exogenous datasets.

Chapter 6 evolves the application of our methodology by studying the error, flexibility, robustness, extensibility, and integrability of the ST-MDF model. The result is a global mobility model for the entire city, with fine-grained spatio-temporal resolution. The ST-MDF model benefits from a modular structure that allows for the integration of additional data sources, making it a valuable tool for optimizing transport in congested urban environments.

8.1.3 Air quality

Air quality is of utmost importance for human health, as it directly affects the respiratory system and overall well-being. Consequently, governments, organizations, and individuals are urged to:

- take steps to reduce air pollution,
- promote cleaner energy sources,
- improve public transportation, and
- implement effective environmental policies.

The collection, analysis, and forecasting of air pollutant concentrations are essential for adopting policies to enhance air quality. Chapter 7 delves into the application of our methodology for air quality forecasting. Specifically, we develop the Spatio-

Temporal Air Quality Forecaster (ST-AQF), which provides forecasts across multiple geolocations and time horizons. The key contributions of this chapter include:

- Extension of the mesh-grid to forecast up to eleven variables.
- Enhancement of the modular design by integrating multiple weather stations.
- Achieving reasonable error levels compared to established baselines and literature.
- Preservation of flexibility and robustness NFFs.

Section 7.4 presents the experiments conducted to evaluate the ST-AQF model. These experiments utilize an extensive 10-year air quality dataset covering eleven different pollutants. Using our methodology (outlined in Chapter 4), we achieve high spatial and temporal forecasting resolutions of approximately 500 meters and one hour, respectively. As with previous use cases, the ST-AQF scores suitable error metrics compared to baseline models and existing literature. Additionally, the model naturally integrates exogenous information (such as weather conditions and temporal features), enhancing its predictions. The ST-AQF model can adapt to sensor failures and changes within the sensor network, denoting flexibility and robustness. Most importantly, this case study highlights the model's ability to forecast multiple pollutants simultaneously. Although the best results are achieved for individual variables, a specific combination of variables and fine-tuning suggests room for further improvement.

The described NFFs facilitate the application of this framework in real-world, dynamic scenarios. For example, these features are useful when new sensors are installed, others fail, or different pollutants become relevant. Altogether, the ST-AQF model is an advantageous framework to guide air quality experts in developing policies to reduce air pollution.

8.1.4 General conclusions

Our goal is to identify beneficial properties for real-world operation of spatio-temporal forecasting models, as outlined in Chapter 3. We refer to these characteristics as NFFs. This thesis explores three diverse forecasting use cases to evaluate the methodology described in Chapter 4.

The first use case involves solar irradiance forecasting with fine spatial and temporal resolutions (Chapter 5). The proposed model demonstrates lower forecast error metrics compared to baseline methods. However, what stands out is its capability to adapt to a different number of solar irradiance sensors. This ability translates into flexibility and robustness NFFs, as discussed in Section 8.1.1. Nevertheless, this use case is limited to a one-square-kilometer Region Of Interest (ROI) with only 17 sensors (Sengupta & Andreas, 2010). The subsequent use cases

explore larger areas and a greater number of sensors to address these constraints.

In the second use case, we shift our focus to urban mobility demand forecasting (Chapter 6.) We follow the same modeling approach while fusing two mobility variables: taxi and bicycle rides (City of Chicago, 2024; Lyft, 2024b). Additionally, we integrate weather and temporal exogenous data, which enhances the forecasting error. The use case involves approximately 800 sensors for each variable, distributed across a 630 km^2 ROI. This is achieved while maintaining competitive error metrics and retaining flexibility and robustness NFFs. Although we combine two variables in this case, we aim to extend this NFF further in the upcoming use case.

The third and final use case explores the application of the methodology for air quality forecasting (Chapter 7). The key aspect of this case is the extension to multiple variables. Experiments illustrate the model's ability to generalize in practice, successfully modeling up to eleven pollutants simultaneously. This case study includes ten years of measurements from 24 sensors spread across 340 km^2 (Ayuntamiento de Madrid, 2024). In addition to achieving low errors, the model accomplishes flexibility, robustness, and integrability NFFs. These results highlight the methodology's potential to support comprehensive, data-driven air quality monitoring strategies.

The three studied use cases allows for a thoughtful evaluation of the methodology, covering multiple aspects in practice:

- **Fine-grain resolution**: Comprehensive spatial and temporal granularities.
- **Error metrics**: Accurate forecasts that are within or better than baselines and the literature.
- **Flexibility**: Ability to incorporate or remove sensors into the network.
- **Robustness**: Ability to recover missing sensor observations based on neighboring or past readings.
- **Extensibility**: Ability to extend the forecast with new related variables.
- **Integrability**: Ability to integrate exogenous information to improve predictions.

Compared to related predictive frameworks, these are distinctive aspects of our methodology and the Spatio-Temporal Forecaster (ST-F), a generic spatio-temporal sensor data model that leverages multiple NFFs.

Each use case presented diverse challenges and limitations, some of which have been discussed in their respective chapters. However, several horizontal lessons emerged across all use cases. First, combining discrete variables with mesh-grids is both challenging and unintuitive, compared to continuous variables. Interpolation

techniques require careful handling to avoid misrepresenting observations. The main challenge here is data sparsity, for which tools like sparse tensors can be especially helpful. Second, we found that "less is more" often applies. The number of model configurations explodes exponentially when dealing with multiple variables, large mesh-grids shapes, and broad input and output windows (i.e., high values of n_v, n_r, n_c, n_x, and n_y). This issue not only complicates model selection but can also degrade performance, as seen in the air quality case when using high values of n_v. These challenges reflect the well-known *curse of dimensionality*, which hinders generalization and increases the risk of overfitting. Third, data quality and availability remain a limiting factor. There are few open datasets suitable for spatio-temporal sensor data forecasting with high resolution. For this reason, we hope that our review of data resources in Chapter 2 will prove useful to other researchers. Finally, while it is often possible to improve the forecasting error metrics of a model, doing so requires significant time and computational resources. In this sense, our focus on NFFs explores additional facets of model evaluation, emphasizing practical applicability alongside performance. Overall, these experiences have reinforced the importance of balancing model accuracy with the need for resilient frameworks that can generalize across domains.

8.2 Future Work

This thesis presents a methodology for handling and forecasting spatio-temporal sensor data. Additionally, it focuses on the importance of developing models with benefitial properties to effectively address challenges once deployed in the real world. Chapters 5, 6, and 7 evaluate the methodology across three diverse forecasting use cases: solar irradiance, mobility demand, and air quality (respectively). Sections 8.2.1, 8.2.2, and 8.2.3 discuss future work regarding each case study. Additionally, Section 8.2.4 explores further ways to extend and enhance the proposed methodology.

8.2.1 Solar irradiance

Chapter 5 leaves some unanswered research questions that we discuss subsequently.

1. *Can the flexibility of the ST-SIF adapt to abrupt changes in wind direction?*

One of the main outcomes of Chapter 5 is that the proposed model accomplishes flexibility and robustness while scoring a competitive forecast skill. Compared to the persistence model, error metrics improve by over 40 % on the first horizon (1 minute), by 20 % on the intermediate horizons (11 and 31 minutes), and by 22 % on the longest one (61 minutes). Notably, the correlation between location and skill is particularly strong for short-term predictions. Abrupt changes in wind direction can negatively affect skill figures, especially for sensors located on the western side

of the map (see Section 5.4.3). A worthwhile future analysis is examining how the RMSE behaves on days when predominant winds come from the west or south of the ROI. If the error increases under these conditions, adding sensors on those sides could help mitigate the issue.

2. *Can the proposed approach be scalable?*

Scalability is another advantageous NFF for the studied use case. It refers to the model's ability to handle the computational demands that arise as the number of measurements of the target variable increases. For a model to be considered scalable, it must be able to operate even as the data grows by orders of magnitude. Scalability can involve the addition of more sensors (finer spatial granularity) or more measurements per time unit (finer temporal resolution). Spatial scalability is a particular case of the flexibility NFF, where the number of sensors is significantly higher. As the number of sensors increases within the same ROI, they will be positioned closer together. This implies that the spatial granularity σ must become finer to ensure the mesh-grid captures all sensor data, thereby increasing model complexity.

Scalability is crucial for smart city applications, where hundreds of thousands of interconnected nodes collaborate to achieve common goals. We must explore more sturdy solutions based on High-Performance Computing (HPC) to address such large-scale problems. The literature provides multiple examples that point in this direction. For example, the URBAN-WASTE EU project (Gruber et al., 2017) deploys sensors to monitor the fill levels of rubbish bins, optimizing waste collection and route planning. Similarly, Grodi et al. (2016) use sensors across parking spaces to provide real-time information on available spots. Huang and Kuo (2018) is another example, where the authors combine Convolutional Neural Networks (CNNs) and Long Short-Term Memorys (LSTMs) for fine particulate matter ($PM_{2.5}$) forecasting.

In the context of a city with solar panels installed on rooftops, we can foresee challenges related to scalability and the transfer of large amounts of data. As seen in Section 5.4.5, the ST-SIF can be integrated into solar panel monitoring systems to forecast solar irradiance. In our experiments, we utilized mesh-grid shapes no larger than 10×10, as there are only 17 pyranometers in the studied case. However, Convolutional LSTMs (ConvLSTMs) are capable of handling larger grid shapes with adequate hardware, as demonstrated in Chapters 6 and 7. Although large open datasets for solar irradiance are scarce, synthetic data offers a viable alternative worth experimenting with.

3. *Can the proposed approach be portable?*

Portability is another important NFF, typically referring to the usability of a framework across different environments. In our context, portability enables the

transfer of the forecasting system to a new location without requiring modifications to its architecture. This means the same model can be deployed elsewhere, provided it is properly tuned with appropriate data. This NFF is feasible as long as the ROIs and climatological conditions are similar. We could leverage the Deep Learning (DL) technique of transfer learning to evaluate this feature.

In the experiments presented in this chapter, we used squared irradiance mesh-grids. However, CNNs can process any rectangular mesh-grid shape, making them suitable for different ROIs. Additionally, we could explore the use of Graph Neural Networks (GNNs) to enhance the system's portability further, enabling the sensor network to adapt to non-square or irregular distributions. Other novel approaches to solar forecasting include transformer-based models, which learn the graph structure rather than relying on a predefined one. For example, Grigsby et al. (2023) apply this architecture to forecast spatio-temporal solar irradiance, as well as traffic and metro departures. Building on this idea, Piantadosi et al. (2024) introduce a transformer-based architecture specifically tailored to PV power prediction. Also recently, Xu et al. (2024) adopt Vision Transformers (ViTs) for short-term PV generation forecasting. A different approach is proposed by Nassimiha et al. (2023), who apply the state-space form of Gaussian processes to obtain similar forecasts, while also incorporating predictive uncertainty.

8.2.2 Mobility demand

Regarding the future work in Chapter 6, there are several directions for further exploration:

1. *Can the extensibility of the ST-MDF model accommodate additional variables?*

We could investigate whether incorporating additional data sources, such as For-Hire Vehicle (FHV), metro, and bus data, would enhance the model's error. For instance, City of Chicago (2021) provides e-scooter trip data from Chicago, as discussed in Section 2.3.1. The Chicago Data Portal also offers datasets on bus and rail services, though they are coarse-grained (e.g., daily or monthly trips). Other public transit agencies, such as the Massachusetts Bay Transportation Authority (MBTA) in Boston, United States of America (USA), offer historical subway and commuter rail data (Massachusetts Bay Transportation Authority, 2024).

The experiments in Chapter 6 utilize a dock-based bike-sharing system (Lyft, 2024b). However, many companies now offer dockless alternatives, allowing users to leave bikes anywhere within set boundaries. Given the flexibility of the ST-MDF model, this type of mobility service could be integrated. Thus, bike counts would be aggregated at each timestamp and across every grid element, eliminating the need for fixed sensors.

2. *Can the integrability of the ST-MDF model benefit from other exogenous data sources?*

We could incorporate points of interest, such as shopping centers or sports facilities, as an additional layer within the mobility mesh-grid. Alternatively, they could be introduced in the model through a separate module. Their spatio-temporal nature could enhance predictions during specific events, such as concerts or sports games. Additionally, meteorological conditions can vary significantly across a large area like the one under study. Therefore, integrating more weather stations could further improve the model. If the network of weather sensors is sufficiently dense, we could even construct a weather mesh-grid similar to the mobility grid.

3. *Can the ST-MDF model benefit from trip terminations?*

Chapter 6 examines the mobility demand problem solely from the perspective of trip beginnings. However, both datasets used in Chapter 6 provide concluding geolocations and timestamps. This information could enhance mobility demand forecasts in several ways. Firstly, it can identify mobility hotspots for future trip starts. Additionally, trip terminations provide valuable insights for fleet managers regarding taxi and bike distribution throughout the city.

4. *Is the ST-MDF model consistent?*

Consistency is an important NFF for the mobility demand use case. A consistent model produces similar outputs for similar inputs. In other words, it shows reliable behavior under varying conditions or across multiple iterations. Mobility demand observations typically vary slightly from one fine-grained timestamp to the next. Therefore, a consistent model should provide reliable forecasts that account for these progressive shifts.

There are additional analyses that we could conduct in the future. Section 6.4.3 highlighted the influence of the parameter n_x on the scored error. We should further investigate this and explore the possibility of incorporating n_x into the periodical module. Furthermore, utilizing GNNs could be beneficial, as they are already employed in related tasks. Some examples include traffic forecasting (Li et al., 2018; Yu et al., 2018) and ride-hailing (Geng et al., 2019; Jin et al., 2020). These GNN-based approaches tackle the forecasting problem in a similar manner to ours, but observations are recorded in a graph-structured data array. Therefore, the model considers the dependencies between adjacent nodes in the graph, making their physical distance less significant. Finally, we could study the portability of the ST-MDF model to a different city, such as New York City (NYC), where similar data are available online.

8.2.3 Air quality

In Chapter 7, several questions remain unresolved, which could be addressed in future work.

1. *Can a thoughtful combination of air pollutants improve the overall error*

metrics of the ST-AQF?

Section 7.4.5 shows that the error of the ST-AQF is higher for $n_v = 11$ than for $n_v \in \{4, 6, 8\}$, in both the nitrogen dioxide (NO_2) and coarse particulate matter (PM_{10}) cases. To further enhance the sensitivity analysis, we should investigate whether certain pollutants contribute to improving the forecasting performance of others. Given the large number of possible combinations accounting for eleven variables, this analysis would be computationally demanding. However, this task should become manageable if appropriate hardware and heuristics are used to eliminate less promising cases. Additionally, complementing this analysis with parameter tuning for the optimal combinations of variables may lead to further improvements in the error.

2. *Could interpretability approaches facilitate the usability of the ST-AQF and its extension to other variables?*

Interpretability is a horizontal NFF that has gained increasing attention in recent years. In the context of the ST-AQF, interpretability can help air quality experts better understand the model's decision-making process and improve trust. Additionally, it could simplify the extension of the ST-AQF to other pollutants, as discussed in the previous question. This NFF could be particularly valuable when deploying the model in different ROIs or when fine-tuning is required.

3. *Could the ST-AQF model benefit from traffic observations or models?*

Transportation emissions significantly impact air quality, especially in urban environments. Incorporating traffic observations as additional exogenous information could improve the results of the ST-AQF. An alternative approach is incorporating traffic predictions for the forecasted horizons using a trained model, such as the one proposed by Méndez et al. (2023). Moreover, mobility demand predictions from our ST-MDF model (described in Chapter 6) can also serve as an indicator of traffic density.

Additionally, further analyses are worth exploring. One potential direction is to examine whether incorporating mesh-grids into the weather module could enhance the predictive capabilities of the ST-AQF. It would also be valuable to explore alternative network architectures, such as GNNs (Ge et al., 2021; Jiang et al., 2022) and transformers (Chen et al., 2022; Jamil & Roy, 2023). We are also interested in investigating alternative imputation methods to improve robustness, particularly DL-based approaches (Ni & Cao, 2022). Furthermore, additional NFFs could be considered, such as spatial or temporal scalability and fine-tuning pre-trained models for different ROIs (portability).

Reliable, adaptive, robust, and flexible air quality forecasting models are highly valuable. However, beyond forecasting, it is essential to take action based on these models' results and historical data analysis to address air pollution and mitigate

the otherwise inevitable climate change. As Landrigan et al. (2018) discuss, such actions include:

- deploying monitoring systems,
- identifying pollution sources,
- enforcing pollution reduction policies, and
- developing and facilitating the smooth transition to clean energy and transportation alternatives.

8.2.4 Methodology

This section discusses potential future work related to the overall methodology, following the examination of each use case in the previous three sections. Several improvements could enhance the forecasting framework and its features. One possible direction is to aim for further forecasting horizons, as the studied cases only went up to an eight-hour timeframe. However, longer horizons often introduce greater variability, increasing the challenge. A common approach to address this issue is to iteratively feed the model's predictions back into its input, as demonstrated in previous studies (Lam et al., 2023). For example, to forecast a variable's value 24 hours ahead with a model limited to an eight-hour horizon, the model could be executed three times, with each output feeding back as input for the next prediction.

Our methodology considers sensor observations with three dimensions: time, longitude, and latitude. However, this approach could be extended to a fourth dimension, i.e., height. Such an extension could be useful in cases where the z-coordinate is relevant (e.g., computational particle physics). Adapting the model to include this additional dimension would increase computational demand slightly. Nonetheless, this enhancement would expand the model's applicability to fields that require this additional dimension.

This thesis studies various NFFs for forecasting spatio-temporal sensor data, focusing on flexibility, robustness, extensibility, and integrability. The preceding sections discuss additional ones for the studied use cases:

- *scalability* and *portability* for solar irradiance forecasting,
- *consistency* for the mobility demand use case, and
- *interpretability* in the context of air quality forecasting.

In addition to those already considered, several other NFFs are worth noting for both the studied and potential future use cases. A key methodological step involves constructing the mesh-grid and implementing its reversal (see Section 4.1.5). A

decisive property of this conversion is *reversibility*, which supports both generalization and reproducibility. Reversible mesh-grid constructions allow the retrieval of original values. Specifically, if f is the mesh-grid construction function and f^{-1} its reversal, then $f^{-1}(f(x)) = x_s^t$ for any variable observation x from sensor s at timestamp t. Further research into these techniques could enhance the sensor interface, making it even more flexible and robust.

Given the generalization capabilities of the proposed models, a final question worth considering is the following:

1. *Could the proposed ST-F models be integrated into a larger smart city project?*

Our proposal could be incorporated into a "system of systems" that consolidates various smart city utilities. An example of such a system is Google's Environmental Insights Explorer (Google, 2024). This project aims to equip cities and researchers with data and modeling capabilities to support emission reduction efforts. The primary focus areas of this project include transportation and building emissions, rooftop solar potential, and tree canopy coverage. We believe that the ST-SIF, ST-MDF, and the ST-AQF models could serve as valuable tools within this or similar projects. Moreover, the NFFs studied in this work support the integration of these forecasting models into a wide range of scenarios and environments. Overall, our methodology can help urban planners and policymakers establish emissions baselines, explore renewable energy potential, and make data-driven decisions toward sustainability goals.

Bibliography

Aasim, Singh, S., & Mohapatra, A. (2019). Repeated wavelet transform based ARIMA model for very short-term wind speed forecasting. *Renewable energy*, *136*, 758–768. DOI: 10.1016/j.renene.2019.01.031 (Cited on p. 1).

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ (last accessed: 2025/03/15). (Cited on pp. 74, 99, 126)

Abirami, S., & Chitra, P. (2021). Regional air quality forecasting using spatiotemporal deep learning. *Journal of Cleaner Production*, *283*, 125341. DOI: 10.1016/j.jclepro.2020.125341 (Cited on p. 124).

Acquier, A., Jordan, E., Prado-Rujas, I.-I., Serrano, M., & Smith, B. (2022). *Sensing Air - A citywide citizen science air quality monitoring system* (poster presentation). Youth Climate Assembly 2022, University of Galway, Ireland. (not available online). (Cited on p. 7).

AEMET. (2024). *AEMET OpenData - API REST for accessing meteorological and climatological data* (dataset). https://opendata.aemet.es/centrodedescargas/inicio (last accessed: 2025/03/15). (Cited on pp. 13, 27)

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Spinger. DOI: 10.1007/978-3-031-29642-0. (Cited on p. 44).

Agrawal, S., Barrington, L., Bromberg, C., Burge, J., Gazen, C., & Hickey, J. (2019). Machine Learning for Precipitation Nowcasting from Radar Images. *arXiv preprint*. DOI: 10.48550/arXiv.1912.12132 (Cited on pp. 15, 26, 28, 53).

Almendras, L., Campoy, J., Prado-Rujas, I.-I., Risco-Martin, J. L., Perez, M. S., & Olcoz, K. (2022). A distributed IoT-based simulation framework for solar energy management and forecasting, In *Jornadas SARTECO 2022*, Alicante, Spain. DOI: 10.5281/zenodo.7075818. (Cited on p. 6).

Alzahrani, A., Shamsi, P., Dagli, C., & Ferdowsi, M. (2017). Solar irradiance forecasting using deep neural networks. *Procedia Computer Science*, *114*, 304–313. DOI: 10.1016/j.procs.2017.09.045 (Cited on pp. 14, 28, 72).

Amaro e Silva, R., & Brito, M. C. (2018). Impact of network layout and time resolution on spatio-temporal solar forecasting. *Solar Energy*, *163*, 329–337. DOI: 10.1016/j.solener.2018.01.095 (Cited on pp. 1, 14, 26, 28, 56, 77, 78).

Amato, F., Guignard, F., Robert, S., & Kanevski, M. (2020). A novel framework for spatio-temporal prediction of environmental data using deep learning. *Scientific Reports*, *10*(1), 1–11. DOI: 10.1038/s41598-020-79148-7 (Cited on p. 2).

Anthopoulos, L. G., & Reddick, C. G. (2016). Smart city and smart government: Synonymous or complementary?, In *Proceedings of the 25th international conference companion on world wide web*. DOI: 10.1145/2872518.2888615. (Cited on p. 9).

Arbizu-Barrena, C., Ruiz-Arias, J. A., Rodríguez-Benítez, F. J., Pozo-Vázquez, D., & Tovar-Pescador, J. (2017). Short-term solar radiation forecasting by advecting and diffusing MSG cloud index. *Solar Energy*, *155*, 1092–1103. DOI: 10.1016/j.solener.2017.07.045 (Cited on pp. 12, 14, 15, 28).

Ayet, A., & Tandeo, P. (2018). Nowcasting solar irradiance using an analog method and geostationary satellite images. *Solar Energy*, *164*, 301–315. DOI: 10.1016/j.solener.2018.02.068 (Cited on pp. 14, 15, 28).

Ayuntamiento de Madrid. (2024). *Calidad del aire. Datos horarios desde 2001 - Portal de datos abiertos* (dataset). https://datos.madrid.es/.../ (last accessed: 2025/03/15). (Cited on pp. 20, 26, 27, 112, 147, 175).

Aznarte, J. L. (2017). Probabilistic forecasting for extreme no2 pollution episodes. *Environmental Pollution*, *229*, 321–328. DOI: 10.1016/j.envpol.2017.05.079 (Cited on p. 133).

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166. DOI: 10.1109/72.279181 (Cited on p. 52).

Berman, J. D., & Ebisu, K. (2020). Changes in U.S. air pollution during the COVID-19 pandemic. *Science of The Total Environment*, *739*, 139864. DOI: 10.1016/j.scitotenv.2020.139864 (Cited on p. 21).

Bike Share Research contributors. (2021). *Open bike share data [β]* (dataset). https://bikeshare-research.org/ (last accessed: 2025/03/15). (Cited on pp. 16, 27)

Boufidis, N., Nikiforiadis, A., Chrysostomou, K., & Aifadopoulou, G. (2020). Development of a station-level demand prediction and visualization tool to support bike-sharing systems' operators. *Transportation Research Procedia*, *47*, 51–58. DOI: 10.1016/j.trpro.2020.03.072 (Cited on p. 28).

Campoy, J., Prado-Rujas, I.-I., Risco-Martin, J. L., Olcoz, K., & Perez, M. S. (2023). Distributed training and inference of deep learning solar energy forecasting models, In *2023 31st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Naples, Italy. DOI: 10.1109/PDP59025.2023.00035. (Cited on p. 6).

Campoy Heredero, J. (2023). *Exploring a deep learning based spatio-temporal forecasting model: Testing for scalability and other features* (Master's thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/75899/. (Cited on p. 7)

Chai, D., Wang, L., & Yang, Q. (2018). Bike flow prediction with multi-graph convolutional networks, In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, New York, NY, USA, Association for Computing Machinery. DOI: 10.1145/3274895.3274896. (Cited on pp. 19, 28, 103, 104, 107).

Chen, Y., Chen, X., Xu, A., Sun, Q., & Peng, X. (2022). A hybrid CNN-Transformer model for ozone concentration prediction. *Air Quality, Atmosphere & Health*, *15*(9), 1533–1546. DOI: 10.1007/s11869-022-01197-w (Cited on p. 152).

Chen, Y., Zhang, S., Zhang, W., Peng, J., & Cai, Y. (2019). Multifactor spatio-temporal correlation model based on a combination of convolutional neural network and long short-term memory neural network for wind speed forecasting. *Energy Conversion and Management*, *185*, 783–799. DOI: 10.1016/j.enconman.2019.02.018 (Cited on p. 1).

Chollet, F. Et al. (2015). Keras. https://keras.io (last accessed: 2025/03/15). (Cited on pp. 10, 74, 99, 126)

City of Chicago. (2021). *E-Scooter Trips at Chicago, USA – 2020* (dataset). https://data.cityofchicago.org/Transportation/E-Scooter-Trips-2020/3rse-fbp6 (last accessed: 2025/03/15). (Cited on pp. 16, 27, 150)

City of Chicago. (2024). *Taxi trips at Chicago, USA* (dataset). https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew (last accessed: 2025/03/15). (Cited on pp. 16, 26, 27, 87, 147, 173)

Comunidad de Madrid. (2023). *Regional holidays - Open data portal of the Region of Madrid, Spain* (dataset). https://datos.comunidad.madrid/catalogo/dataset/festivos_regionales_locales_historico (last accessed: 2025/03/15). (Cited on pp. 125, 175)

Cui, Z., Ke, R., Pu, Z., Ma, X., & Wang, Y. (2020). Learning traffic as a graph: A gated graph wavelet recurrent neural network for network-scale traffic prediction. *Transportation Research Part C: Emerging Technologies*, *115*, 102620. DOI: 10.1016/j.trc.2020.102620 (Cited on p. 86).

Dbouk, T., Visez, N., Ali, S., Shahrour, I., & Drikakis, D. (2022). Risk assessment of pollen allergy in urban environments. *Scientific Reports*, *12*(1), 21076. DOI: 10.1038/s41598-022-24819-w (Cited on pp. 11, 25).

Del Barrio, E., Cuesta-Albertos, J. A., & Matrán, C. (2018). An optimal transportation approach for assessing almost stochastic order. In *The Mathematics of the Uncertain* (pp. 33–44). Springer. DOI: 10.1007/978-3-319-73848-2_3. (Cited on p. 131).

de Medrano, R., de Buen Remiro, V., & Aznarte, J. L. (2021). SOCAIRE: Forecasting and monitoring urban air quality in Madrid. *Environmental Modelling & Software*, *143*, 105084. DOI: 10.1016/j.envsoft.2021.105084 (Cited on pp. 23, 24, 26, 28, 133).

Deri, J. A., Franchetti, F., & Moura, J. M. (2016). Big data computation of taxi movement in New York City, In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. DOI: 10.1109/BigData.2016.7840904. (Cited on pp. 17, 23, 28).

Devasekhar, V., & Natarajan, P. (2023). Prediction of Air Quality and Pollution using Statistical Methods and Machine Learning Techniques. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *14*(4). DOI: 10.14569/IJACSA.2023.01404103 (Cited on pp. 22, 24, 28).

Directorate-General for Environment, European Commission. (2008). *Air quality standards* (website). https://environment.ec.europa.eu/topics/air/air-quality/eu-air-quality-standards_en (last accessed: 2025/03/15). (Cited on p. 111)

Domingo, J. L., & Rovira, J. (2020). Effects of air pollutants on the transmission and severity of respiratory viral infections. *Environmental Research*, *187*, 109650. DOI: 10.1016/j.envres.2020.109650 (Cited on p. 111).

Drewil, G. I., & Al-Bahadili, R. J. (2022). Air pollution prediction using LSTM deep learning and metaheuristics algorithms. *Measurement: Sensors*, *24*, 100546. DOI: 10.1016/j.measen.2022.100546 (Cited on pp. 22, 24, 28).

Dror, R., Shlomov, S., & Reichart, R. (2019). Deep dominance - how to properly compare deep neural models (A. Korhonen, D. R. Traum, & L. Màrquez, Eds.). In A. Korhonen, D. R. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, Association for Computational Linguistics. DOI: 10.18653/v1/p19-1266. (Cited on p. 131).

Dumoulin, V., & Visin, F. (2018). A guide to convolution arithmetic for deep learning. DOI: 10.48550/arXiv.1603.07285. (Cited on p. 53).

Dutheil, F., Baker, J. S., & Navel, V. (2020). COVID-19 as a factor influencing air pollution? *Environmental Pollution*, *263*, 114466. DOI: 10.1016/j.envpol.2020.114466 (Cited on p. 21).

EMT Madrid. (2023). *BiciMAD open data - Bicycle trips in Madrid, Spain* (dataset). https://opendata.emtmadrid.es/Datos-estaticos/Datos-generales-(1) (last accessed: 2025/03/15). (Cited on pp. 16, 27)

Escribá Pina, E. (2021). *Aprendizaje por refuerzo mediante Deep Learning para las ciudades inteligentes* (Master's thesis). E.T.S. de Ingenieros Informáticos (UPM). https://oa.upm.es/68733/. (Cited on pp. 20, 28)

EU Solar Energy Strategy (tech. rep.). (2022). European Commission. European Commission, 1049 Bruxelles/Brussel, Belgium. https://energy.ec.europa.eu/topics/renewable-energy/solar-energy_en (last accessed: 2025/03/15). (Cited on p. 12)

European Environment Agency. (2023). *Air quality and COVID-19* (data visualization). https://www.eea.europa.eu/themes/air/air-quality-and-covid19 (last accessed: 2025/03/15). (Cited on p. 21)

European Environment Agency. (2024a). *Air quality download service* (dataset). https://eeadmz1-downloads-webapp.azurewebsites.net/ (last accessed: 2025/03/15). (Cited on pp. 21, 27)

European Environment Agency. (2024b). *Air quality statistics* (dataset). https://www.eea.europa.eu/data-and-maps/dashboards/air-quality-statistics (last accessed: 2025/03/15). (Cited on p. 21)

European Environment Agency. (2024c). *European air quality index* (data vizualization). https://airindex.eea.europa.eu/AQI/index.html (last accessed: 2025/03/15). (Cited on p. 21)

European Environment Agency. (2024d). *European city air quality viewer* (data visualization). https://www.eea.europa.eu/themes/air/urban-air-quality/european-city-air-quality-viewer (last accessed: 2025/03/15). (Cited on p. 21)

European Environment Agency, González Ortiz, A., Guerreiro, C., & Soares, J. (2020). *Air quality in Europe: 2020 report*. LU, Publications Office of the European Union. DOI: 10.2800/786656. (Cited on pp. 20, 111).

Fairfax, E., Zhu, E., Clinton, N., Maiman, S., Shaikh, A., Macfarlane, W. W., Wheaton, J. M., Ackerstein, D., & Corwin, E. (2023). EEAGER: A Neural Network Model for Finding Beaver Complexes in Satellite and Aerial Imagery. *Journal of Geophysical Research: Biogeosciences*, *128*(6). DOI: 10.1029/2022JG007196 (Cited on p. 22).

Fan, J., Li, Q., Hou, J., Feng, X., Karimian, H., & Lin, S. (2017). A spatiotemporal prediction framework for air pollution based on deep RNN. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *4*, 15. DOI: 10.5194/isprs-annals-IV-4-W2-15-2017 (Cited on p. 1).

Fattorini, D., & Regoli, F. (2020). Role of the chronic air pollution levels in the Covid-19 outbreak risk in Italy. *Environmental Pollution*, *264*, 114732. DOI: 10.1016/j.envpol.2020.114732 (Cited on p. 21).

Fiorello, D., & Zani, L. (2015). EU survey on issues related to transport and mobility. *JRC Science and Policy Report*. DOI: 10.2791/48322 (Cited on p. 85).

Galicia, A., Talavera-Llames, R., Troncoso, A., Koprinska, I., & Martínez-Álvarez, F. (2019). Multi-step forecasting for big data time series based on ensemble learning. *Knowledge-Based Systems*, *163*, 830–841. DOI: 10.1016/j.knosys.2018.10.009 (Cited on pp. 15, 23, 28).

García, M. V., & Aznarte, J. L. (2020). Shapley additive explanations for NO2 forecasting. *Ecological Informatics*, *56*, 101039. DOI: 10.1016/j.ecoinf.2019.101039 (Cited on p. 133).

García Martínez, M. (2021). *Bike sharing demand forecasting using recurrent neural networks* (Bachelor's Thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/66259/. (Cited on p. 7)

Ge, L., Wu, K., Zeng, Y., Chang, F., Wang, Y., & Li, S. (2021). Multi-scale spatiotemporal graph convolution network for air quality prediction. *Applied Intelligence*, *51*, 3491–3505. DOI: 10.1007/s10489-020-02054-y (Cited on p. 152).

Geng, X., Li, Y., Wang, L., Zhang, L., Yang, Q., Ye, J., & Liu, Y. (2019). Spatiotemporal multi-graph convolution network for ride-hailing demand fore-

casting, In *Proceedings of the AAAI conference on artificial intelligence*. DOI: 10.1609/aaai.v33i01.33013656. (Cited on p. 151).

Gibson, D. V., Kozmetsky, G., & Smilor, R. W. (1992). *The technopolis phenomenon: Smart cities, fast systems, global networks*. Rowman & Littlefield. (Cited on p. 9).

Glinz, M. (2007). On non-functional requirements, In *15th IEEE International Requirements Engineering Conference (RE 2007)*. DOI: 10.1109/RE.2007.45. (Cited on p. 23).

Gobierno de Canarias. (2023). *Calidad del aire - Portal de datos abiertos* (dataset). https://datos.canarias.es/catalogos/general/dataset/calidad-del-aire-de-canarias (last accessed: 2025/03/15). (Cited on pp. 20, 27)

González García, D. (2020). *Predicción de la radiación solar mediante el uso de una red neuronal convolucional* (Bachelor's Thesis). ETSI Informática (Universidad Politécnica de Madrid). (not available online). (Cited on p. 7).

Google. (2024). Environmental insights explorer. https://insights.sustainability.google (last accessed: 2025/03/15). (Cited on p. 154)

Grigsby, J., Wang, Z., Nguyen, N., & Qi, Y. (2023). Long-Range Transformers for Dynamic Spatiotemporal Forecasting. DOI: 10.48550/arXiv.2109.12218. (Cited on p. 150).

Grodi, R., Rawat, D. B., & Rios-Gutierrez, F. (2016). Smart parking: Parking occupancy monitoring and visualization system for smart cities, In *Southeastcon 2016*. IEEE. DOI: 10.1109/SECON.2016.7506721. (Cited on pp. 10, 25, 149).

Gros, B. (2016). The design of smart educational environments. *Smart learning environments*, *3*, 1–11. DOI: 10.1186/s40561-016-0039-x (Cited on pp. 11, 25).

Gruber, I., Obersteiner, G., Mayerhofer, J., Ramusch, R., Romain, A., Eriksson, M., Große, J., Nascimento, G. M., Olsen, T. B., de Luca, C., Aranda, P. Z., Kazeroni, M., & Kovács, E. (2017). Compendium of waste management practices in pilot cities and best practices in touristic cities. Zenodo. DOI: 10.5281/zenodo.2651137. (Cited on pp. 11, 25, 149).

Guo, J., Huang, W., & Williams, B. M. (2014). Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies*, *43*, 50–64. DOI: 10.1016/j.trc.2014.02.006 (Cited on p. 90).

Habtemichael, F. G., & Cetin, M. (2016). Short-term traffic flow rate forecasting based on identifying similar traffic patterns. *Transportation Research Part C: Emerging Technologies*, *66*, 61–78. DOI: 10.1016/j.trc.2015.08.017 (Cited on pp. 17, 28).

Haurwitz, B. (1945). Insolation in relation to cloudiness and cloud density. *Journal of meteorology*, *2*(3), 154–166. DOI: 10.1175/1520-0469(1945)002<0154:IIRTCA>2.0.CO;2 (Cited on p. 67).

Haurwitz, B. (1946). Insolation in relation to cloud type. *Journal of Meteorology*, *3*(4), 123–124. DOI: 10.1175/1520-0469(1946)003<0123:IIRTCT>2.0.CO;2 (Cited on p. 67).

Health Effects Institute. (2020). State of global air 2020. Health Effects Institute Boston, MA, USA. (Cited on p. 109).

Heras Aranzana, P. (2020). *Predicción de la radiación mediante el uso de una red neuronal recurrente* (Bachelor's Thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/63345/. (Cited on p. 7)

Hinkelman, L. M. (2013). Differences between along-wind and cross-wind solar irradiance variability on small spatial scales. *Solar Energy*, *88*, 192–203. DOI: 10.1016/j.solener.2012.11.011 (Cited on p. 64).

Hoang, M. X., Zheng, Y., & Singh, A. K. (2016). FCCF: Forecasting citywide crowd flows based on big data, In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. DOI: 10.1145/2996913.2996934. (Cited on pp. 17, 28).

Holmes, N. S., & Morawska, L. (2006). A review of dispersion modelling and its application to the dispersion of particles: An overview of different dispersion models available. *Atmospheric Environment*, *40*(30), 5902–5928. DOI: 10.1016/j.atmosenv.2006.06.003 (Cited on p. 21).

Holmgren, W., Hansen, C., & Mikofski, M. (2018). pvlib Python: A python package for modeling solar energy systems. *Journal of Open Source Software*, *3*(29), 884. DOI: 10.5281/zenodo.3585619 (Cited on p. 67).

Hontoria, L., Aguilera, J., & Zufiria, P. (2002). Generation of hourly irradiation synthetic series using the neural network multilayer perceptron. *Solar Energy*, *72*(5), 441–446. DOI: 10.1016/S0038-092X(02)00010-5 (Cited on pp. 14, 28).

Huang, C.-J., & Kuo, P.-H. (2018). A Deep CNN-LSTM Model for Particulate Matter (PM2.5) Forecasting in Smart Cities. *Sensors*, *18*(7), 2220. DOI: 10.3390/s18072220 (Cited on pp. 22, 24, 28, 149).

Huang, C.-J., Shen, Y., Chen, Y.-H., & Chen, H.-C. (2020). A novel hybrid deep neural network model for short-term electricity price forecasting. *International Journal of Energy Research*. DOI: 10.1002/er.5945 (Cited on pp. 15, 28).

Huertas Tato, J., & Centeno Brito, M. (2019). Using smart persistence and random forests to predict photovoltaic energy production. *Energies*, *12*(1), 100. DOI: 10.3390/en12010100 (Cited on p. 2).

Instituto Tecnológico Agrario de Castilla y León. (2024). *Inforiego API* (dataset). https://www.inforiego.org/opencms/opencms/api_rest/index.html (last accessed: 2025/03/15). (Cited on pp. 13, 27)

Iqbal, R., Maniak, T., & Karyotis, C. (2019). Intelligent Remote Monitoring of Parking Spaces Using Licensed and Unlicensed Wireless Technologies. *IEEE Network*, *33*(4), 23–29. DOI: 10.1109/MNET.2019.1800459 (Cited on pp. 11, 25).

Jamil, S., & Roy, A. M. (2023). An efficient and robust Phonocardiography (PCG)-based Valvular Heart Diseases (VHD) detection framework using Vision Transformer (ViT). *Computers in Biology and Medicine*, *158*, 106734. DOI: 10.1016/j.compbiomed.2023.106734 (Cited on p. 152).

Jiang, B., Chen, S., Wang, B., & Luo, B. (2022). MGLNN: Semi-supervised learning via Multiple Graph Cooperative Learning Neural Networks. *Neural Networks*, *153*, 204–214. DOI: 10.1016/j.neunet.2022.05.024 (Cited on p. 152).

Jiang, W. (2022). Bike sharing usage prediction with deep learning: A survey. *Neural Computing and Applications*. DOI: 10.1007/s00521-022-07380-5 (Cited on pp. 19, 28).

Jiang, Z., Fan, W., Liu, W., Zhu, B., & Gu, J. (2018). Reinforcement learning approach for coordinated passenger inflow control of urban rail transit in peak hours. *Transportation Research Part C: Emerging Technologies*, *88*, 1–16. DOI: 10.1016/j.trc.2018.01.008 (Cited on pp. 20, 28).

Jin, G., Cui, Y., Zeng, L., Tang, H., Feng, Y., & Huang, J. (2020). Urban ride-hailing demand prediction with multiple spatio-temporal information fusion network. *Transportation Research Part C: Emerging Technologies*, *117*, 102665. DOI: 10.1016/j.trc.2020.102665 (Cited on p. 151).

Joint Research Centre. (2024). *PVGIS: Photovoltaic geographical information system* (dataset). https://ec.europa.eu/jrc/en/pvgis (last accessed: 2025/03/15). (Cited on pp. 13, 27, 82)

Khaidem, L., Luca, M., Yang, F., Anand, A., Lepri, B., & Dong, W. (2020). Optimizing transportation dynamics at a city-scale using a reinforcement learning framework. *IEEE Access*, *8*, 171528–171541. DOI: 10.1109/ACCESS.2020.3024979 (Cited on pp. 20, 28).

Kim, D., Chen, Z., Zhou, L.-F., & Huang, S.-X. (2018). Air pollutants and early origins of respiratory diseases. *Chronic Diseases and Translational Medicine*. DOI: 10.1016/j.cdtm.2018.03.003 (Cited on p. 111).

König, P. D. (2021). Citizen-centered data governance in the smart city: From ethics to accountability. *Sustainable Cities and Society*, *75*, 103308. DOI: 10.1016/j.scs.2021.103308 (Cited on pp. 11, 25).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks (F. Pereira, C. J. Burges, L. Bottou, & K. Q. Weinberger, Eds.). In F. Pereira, C. J. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc. https://proceedings.neurips.cc/.../ (last accessed: 2025/03/15). (Cited on p. 53).

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Vinyals, O., Stott, J., Pritzel, A., Mohamed, S., & Battaglia, P. (2023). Learning skillful medium-range global weather forecasting. *Science*, *382*(6677), 1416–1421. DOI: 10.1126/science.adi2336 (Cited on p. 153).

Landrigan, P. J., Fuller, R., Acosta, N. J. R., Adeyi, O., Arnold, R., Basu, N. (, Baldé, A. B., Bertollini, R., Bose-O'Reilly, S., Boufford, J. I., Breysse,

P. N., Chiles, T., Mahidol, C., Coll-Seck, A. M., Cropper, M. L., Fobil, J., Fuster, V., Greenstone, M., Haines, A., ... Zhong, M. (2018). The Lancet Commission on pollution and health. *The Lancet*, *391*(10119), 462–512. DOI: 10.1016/S0140-6736(17)32345-0 (Cited on pp. 109, 153).

Li, P., Pang, Y., & Ren, J. (2024). Spatio-Temporal Graph Convolutional Network Combined Large Language Model: A Deep Learning Framework for Bike Demand Forecasting. *arXiv preprint*. DOI: 10.48550/arXiv.2403.15733 (Cited on pp. 19, 28).

Li, Q., Wu, Z., Ling, R., & Tan, M. (2019). Echo state network-based spatio-temporal model for solar irradiance estimation. *Energy Procedia*, *158*, 3808–3813. DOI: 10.1016/j.egypro.2019.01.868 (Cited on pp. 15, 28).

Li, X., Pan, G., Wu, Z., Qi, G., Li, S., Zhang, D., Zhang, W., & Wang, Z. (2012). Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science*, *6*(1), 111–121. DOI: 10.1007/s11704-011-1192-6 (Cited on pp. 17, 28).

Li, X., Xu, Y., Chen, Q., Wang, L., Zhang, X., & Shi, W. (2021). Short-Term Forecast of Bicycle Usage in Bike Sharing Systems: A Spatial-Temporal Memory Network. *IEEE Transactions on Intelligent Transportation Systems*, 1–12. DOI: 10.1109/TITS.2021.3097240 (Cited on pp. 19, 28).

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, In *International Conference on Learning Representations (ICLR '18)*. DOI: 10.48550/arXiv.1707.01926. (Cited on p. 151).

Lin, L., He, Z., & Peeta, S. (2018). Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. *Transportation Research Part C: Emerging Technologies*, *97*, 258–276. DOI: 10.1016/j.trc.2018.10.011 (Cited on p. 19).

Lippi, M., Bertini, M., & Frasconi, P. (2013). Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, *14*(2), 871–882. DOI: 10.1109/TITS.2013.2247040 (Cited on pp. 18, 28).

Liu, C.-C., Lin, T.-C., Yuan, K.-Y., & Chiueh, P.-T. (2022). Spatio-temporal prediction and factor identification of urban air quality using support vector machine. *Urban Climate*, *41*, 101055. DOI: 10.1016/j.uclim.2021.101055 (Cited on pp. 23, 24, 28).

Liu, J., Li, T., Xie, P., Du, S., Teng, F., & Yang, X. (2020). Urban big data fusion based on deep learning: An overview. *Information Fusion*, *53*, 123–133. DOI: 10.1016/j.inffus.2019.06.016 (Cited on p. 86).

Liu, T., Wu, W., Zhu, Y., & Tong, W. (2020). Predicting taxi demands via an attention-based convolutional recurrent neural network. *Knowledge-Based Systems*, *206*, 106294. DOI: 10.1016/j.knosys.2020.106294 (Cited on pp. 19, 28, 102, 103, 107).

Lopez-Gomez, I., McGovern, A., Agrawal, S., & Hickey, J. (2023). Global extreme heat forecasting using neural weather models. *Artificial Intelligence for the*

Earth Systems, *2*, e220035. DOI: 10.1175/AIES-D-22-0035.1 (Cited on p. 22).

Lyft. (2024a). *Citi Bike Trip Histories at New York City, USA* (dataset). https://ride.citibikenyc.com/system-data (last accessed: 2025/03/15). (Cited on pp. 16, 27)

Lyft. (2024b). *Divvy Bike Trip Histories at Chicago, USA* (dataset). https://divvybikes.com/system-data (last accessed: 2025/03/15). (Cited on pp. 16, 26, 27, 87, 90, 147, 150, 173)

Ma, S., Zheng, Y., & Wolfson, O. (2013). T-share: A large-scale dynamic taxi ridesharing service, In *2013 IEEE 29th international conference on data engineering (ICDE)*. DOI: 10.1109/ICDE.2013.6544843. (Cited on pp. 17, 28).

Ma, X., Zhang, J., Du, B., Ding, C., & Sun, L. (2018). Parallel architecture of convolutional bi-directional LSTM neural networks for network-wide metro ridership prediction. *IEEE Transactions on Intelligent Transportation Systems*, *20*(6), 2278–2288. DOI: 10.1109/TITS.2018.2867042 (Cited on p. 18).

Ma, Y., Lin, T., Cao, Z., Li, C., Wang, F., & Chen, W. (2015). Mobility viewer: An eulerian approach for studying urban crowd flow. *IEEE Transactions on Intelligent Transportation Systems*, *17*(9), 2627–2636. DOI: 10.1109/TITS.2015.2498187 (Cited on pp. 16, 17, 28).

Mao, W., Wang, W., Jiao, L., Zhao, S., & Liu, A. (2021). Modeling air quality prediction using a deep learning approach: Method optimization and evaluation. *Sustainable Cities and Society*, *65*, 102567. DOI: 10.1016/j.scs.2020.102567 (Cited on pp. 21, 22, 24, 28).

Markmann, S. (2023). *Application of graph neural networks for urban mobility demand forecasting* (Master's thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/75995/. (Cited on p. 7)

Marrero, D., Macías, E., Suárez, á., Santana, J. A., & Mena, V. (2019). Energy Saving in Smart City Wireless Backbone Network for Environment Sensors. *Mobile Networks and Applications*, *24*(2), 700–711. DOI: 10.1007/s11036-016-0786-5 (Cited on pp. 11, 23, 25).

Martín Otero, Á. (2018). *Predicción de nubes a corto plazo para una plataforma solar a partir de datos radiométricos* (Master's thesis). Universidad Complutense de Madrid. https://hdl.handle.net/20.500.14352/19959. (last accessed: 2025/03/15). (Cited on pp. 14, 28)

Massachusetts Bay Transportation Authority. (2024). *MBTA Open Data Portal* (dataset). https://mbta-massdot.opendata.arcgis.com/ (last accessed: 2025/03/15). (Cited on p. 150)

Méndez, M., Merayo, M. G., & Núñez, M. (2023). Long-term traffic flow forecasting using a hybrid CNN-BiLSTM model. *Engineering Applications of Artificial Intelligence*, *121*, 106041. DOI: 10.1016/j.engappai.2023.106041 (Cited on p. 152).

Meteosim. (2021). *Sistema de pronóstico y de diagnóstico sobre la calidad del aire en andalucía* (website). https://meteosim.com/meteosim-desarrolla-.../ (last accessed: 2025/03/15). (Cited on pp. 23, 24, 28).

Miao, F., Han, S., Lin, S., Stankovic, J. A., Zhang, D., Munir, S., Huang, H., He, T., & Pappas, G. J. (2016). Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach. *IEEE Transactions on Automation Science and Engineering*, *13*(2), 463–478. DOI: 10.1109/TASE.2016.2529580 (Cited on p. 17).

Ministerio para la Transición Ecológica y el Reto Demográfico. (2023). *Histórico de la calidad del aire en España* (dataset). https://www.miteco.gob.es/.../ (last accessed: 2025/03/15). (Cited on pp. 20, 27).

Ministry of Environment, Taiwan. (2024). *Taiwan air quality monitoring network* (dataset). https://airtw.moenv.gov.tw/CHT/Query/His_Data.aspx (last accessed: 2025/03/15). (Cited on pp. 21, 27)

Mobility Data. (2024). *Mobility database-catalogs* (dataset). https://github.com/MobilityData/mobility-database-catalogs (last accessed: 2025/03/15). (Cited on pp. 16, 27)

Mohanty, S. P., Choppali, U., & Kougianos, E. (2016). Everything you wanted to know about smart cities: The internet of things is the backbone. *IEEE Consumer Electronics Magazine*, *5*(3), 60–70. DOI: 10.1109/MCE.2016.2556879 (Cited on p. 10).

Moreira-Matias, L., Gama, J., Ferreira, M., & Damas, L. (2012). A predictive model for the passenger demand on a taxi network, In *2012 15th International IEEE Conference on Intelligent Transportation Systems*. DOI: 10.1109/ITSC.2012.6338680. (Cited on pp. 17, 28).

Moya Iratxeta, K. (2024). *Predicción de la demanda de movilidad espacio-temporal del transporte público mediante transformers* (Master's thesis). ETSI Informática (Universidad Politécnica de Madrid). https://oa.upm.es/82617/. (Cited on p. 8)

NASA Langley Research Center. (2024). *NASA Prediction Of Worldwide Energy Resources (POWER)* (dataset). https://power.larc.nasa.gov/ (last accessed: 2025/03/15). (Cited on pp. 13, 27)

Nassimiha, S., Dudfield, P., Kelly, J., Deisenroth, M. P., & Takao, S. (2023). Short-term Prediction and Filtering of Solar Power Using State-Space Gaussian Processes. *arXiv preprint*. DOI: 10.48550/arXiv.2302.00388 (Cited on p. 150).

Natural Resources Canada. (2024). *NRCan - High-Resolution Solar Radiation Datasets* (dataset). https://www.nrcan.gc.ca/energy/renewable-electricity/solar-photovoltaic/18409 (last accessed: 2025/03/15). (Cited on pp. 13, 27)

Ni, Q., & Cao, X. (2022). MBGAN: An improved generative adversarial network with multi-head self-attention and bidirectional RNN for time series imputation. *Engineering Applications of Artificial Intelligence*, *115*, 105232. DOI: 10.1016/j.engappai.2022.105232 (Cited on p. 152).

NYC Taxi & Limousine Commission. (2024). *Taxi trips at New York City, USA* (dataset). https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page (last accessed: 2025/03/15). (Cited on pp. 16, 27)

OpenAQ. (2024). *Air quality open data* (dataset). https://openaq.org (last accessed: 2025/03/15). (Cited on pp. 21, 27)

Ouyang, X., Yang, Y., Zhang, Y., Zhou, W., & Guo, D. (2023). Dual-channel spatial–temporal difference graph neural network for PM 2.5 forecasting. *Neural Computing and Applications*, *35*(10), 7475–7494. DOI: 10.1007/s00521-022-08036-0 (Cited on pp. 22, 28).

Piantadosi, G., Dutto, S., Galli, A., De Vito, S., Sansone, C., & Di Francia, G. (2024). Photovoltaic power forecasting: A Transformer based framework. *Energy and AI*, *18*, 100444. DOI: 10.1016/j.egyai.2024.100444 (Cited on p. 150).

Polson, N. G., & Sokolov, V. O. (2017). Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, *79*, 1–17. DOI: 10.1016/j.trc.2017.02.024 (Cited on pp. 18, 28).

Prado-Rujas, I.-I., García-Dopico, A., Serrano, E., Córdoba, M. L., & Pérez, M. S. (2024). A multivariable sensor-agnostic framework for spatio-temporal air quality forecasting based on Deep Learning. *Engineering Applications of Artificial Intelligence*, *127*, 107271. DOI: 10.1016/j.engappai.2023.107271 (Cited on pp. 6, 109).

Prado-Rujas, I.-I., García-Dopico, A., Serrano, E., & Pérez, M. S. (2020). *Results, graphs and code for a Deep Learning system for spatio-temporal solar irradiance forecasting* (repository). https://github.com/iipr/solar-irradiance (last accessed: 2025/03/15). (Cited on pp. 5, 61, 65, 67, 69, 71, 74, 75, 77, 78, 81, 83, 172)

Prado-Rujas, I.-I., García-Dopico, A., Serrano, E., & Pérez, M. S. (2021). A flexible and robust deep learning-based system for solar irradiance forecasting. *IEEE Access*, *9*, 12348–12361. DOI: 10.1109/ACCESS.2021.3051839 (Cited on pp. 5, 61).

Prado-Rujas, I.-I., Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2021a). *Datasets for a Deep Learning system for mobility demand forecasting* (dataset). DOI: 10.5281/zenodo.5166839. (Cited on pp. 6, 85, 88, 89, 94, 173).

Prado-Rujas, I.-I., Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2021b). Predicción espacio-temporal: Más allá del error/precisión, In *Congreso Español De Informática 2021 (CEDI'21)*, Málaga, Spain. DOI: 10.5281/zenodo.5530048. (Cited on pp. 5, 33).

Prado-Rujas, I.-I., Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2021c). *Results, graphs and code for a Deep Learning system for spatio-temporal mobility demand forecasting* (repository). https://github.com/iipr/mobility-demand (last accessed: 2025/03/15). (Cited on pp. 6, 85, 90, 93, 94, 100, 173)

Prado-Rujas, I.-I., Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2023a). Combining heterogeneous data sources for spatio-temporal mobility demand forecasting. *Information Fusion*, *91*, 1–12. DOI: 10.1016/j.inffus.2022.09.028 (Cited on pp. 6, 85).

Prado-Rujas, I.-I., Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2023b). *Datasets for a Deep Learning system for spatio-temporal air quality forecasting* (dataset). DOI: 10.5281/zenodo.7764528. (Cited on pp. 7, 109, 122, 128, 175).

Prado-Rujas, I.-I., Serrano, E., García-Dopico, A., Córdoba, M. L., & Pérez, M. S. (2023c). *Results, graphs and code for a Deep Learning system for spatio-temporal air quality forecasting* (repository). https://github.com/iipr/air-quality (last accessed: 2025/03/15). (Cited on pp. 7, 109, 114, 122, 125, 126, 128, 129, 131, 133–135, 138, 139, 175)

Pressman, R. S. (2005). *Software engineering: A practitioner's approach*. (Cited on p. 23).

Qi, Y., Li, Q., Karimian, H., & Liu, D. (2019). A hybrid model for spatiotemporal forecasting of PM2.5 based on graph convolutional neural network and long short-term memory. *Science of The Total Environment*, *664*, 1–10. DOI: 10.1016/j.scitotenv.2019.01.333 (Cited on pp. 22, 24, 28).

Qian, X., Ukkusuri, S. V., Yang, C., & Yan, F. (2020). Short-term demand forecasting for on-demand mobility service. *IEEE Transactions on Intelligent Transportation Systems*. DOI: 10.1109/TITS.2020.3019509 (Cited on p. 28).

Qing, X., & Niu, Y. (2018). Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM. *Energy*, *148*, 461–468. DOI: 10.1016/j.energy.2018.01.177 (Cited on p. 14).

Rao, S. K., & Prasad, R. (2018). Impact of 5G technologies on smart city implementation. *Wireless Personal Communications*, *100*, 161–176. DOI: 10.1007/s11277-018-5618-4 (Cited on pp. 11, 25).

Regueiro Tuñón, I. (2021). *Simulación en la nube de nodos medidores de radiación solar desplegados en un contexto IoT* (Master's thesis). Universidad Complutense de Madrid. https://hdl.handle.net/20.500.14352/5139. (Cited on p. 81)

Risco-Martin, J. L., Almendras, L., Prado-Rujas, I.-I., & Campoy, J. (2024). *Cloud-based Analysis and Integration for Data Efficiency (CAIDE)* (repository). https://github.com/jlrisco/caide (last accessed: 2025/03/15). (Cited on pp. 83, 172)

Risco-Martín, J. L., Mittal, S., Henares, K., Cardenas, R., & Arroba, P. (2023). xDEVS: A toolkit for interoperable modeling and simulation of formal discrete event systems. *Software: Practice and Experience*, *53*(3), 748–789. DOI: 10.1002/spe.3168 (Cited on p. 81).

Risco-Martín, J. L., Prado-Rujas, I.-I., Campoy, J., Pérez, M. S., & Olcoz, K. (2024). Advanced simulation-based predictive modelling for solar irradiance sensor farms. *Journal of Simulation*, 1–18. DOI: 10.1080/17477778.2024.2333775 (Cited on pp. 5, 81).

Rodrigues, F., Markou, I., & Pereira, F. C. (2019). Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. *Information Fusion*, *49*, 120–129. DOI: 10.1016/j.inffus.2018.07.007 (Cited on pp. 19, 28).

Roos, J., Gavin, G., & Bonnevay, S. (2017). A dynamic bayesian network approach to forecast short-term urban rail passenger flows with incomplete data. *Transportation Research Procedia*, *26*, 53–61. DOI: 10.1016/j.trpro.2017.07.008 (Cited on pp. 17, 28).

Roy, A. M., & Bhaduri, J. (2023). DenseSPH-YOLOv5: An automated damage detection model based on DenseNet and Swin-Transformer prediction head-enabled YOLOv5 with attention mechanism. *Advanced Engineering Informatics*, *56*, 102007. DOI: 10.1016/j.aei.2023.102007 (Cited on p. 22).

Sengupta, M., & Andreas, A. (2010). *Oahu Solar Measurement Grid (1-Year Archive): 1-Second Solar Irradiance; Oahu, Hawaii (Data); NREL Report No. DA-5500-56506* (dataset). DOI: 10.5439/1052451. (Cited on pp. 5, 13, 25, 27, 28, 64, 71, 82, 146, 172).

Sheffield Solar, University of Sheffield. (2024). *Photovoltaic live API* (dataset). https://www.solar.sheffield.ac.uk/api/ (last accessed: 2025/03/15). (Cited on pp. 13, 27)

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-C. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, In *Proceedings of the 28th international conference on neural information processing systems - volume 1*, Montreal, Canada, MIT Press. https://dl.acm.org/doi/abs/10.5555/2969239.2969329 (last accessed: 2025/03/15). (Cited on pp. 15, 26, 28, 53)

Sikder, A. K., Acar, A., Aksu, H., Uluagac, A. S., Akkaya, K., & Conti, M. (2018). IoT-enabled smart lighting systems for smart cities, In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. DOI: 10.1109/CCWC.2018.8301744. (Cited on pp. 11, 25).

Simeunović, J., Schubnel, B., Alet, P.-J., & Carrillo, R. E. (2021). Spatio-temporal graph neural networks for multi-site PV power forecasting. *IEEE Transactions on Sustainable Energy*, *13*(2), 1210–1220. DOI: 10.1109/TSTE.2021.3125200 (Cited on pp. 15, 28).

Taylor, S. J., & Letham, B. (2017). Forecasting at scale. *PeerJ Preprints*, *5*, e3190v2. DOI: 10.7287/peerj.preprints.3190v2 (Cited on p. 82).

Tian, C., Zhu, X., Hu, Z., & Ma, J. (2021). A transfer approach with attention reptile method and long-term generation mechanism for few-shot traffic prediction. *Neurocomputing*, *452*, 15–27. DOI: 10.1016/j.neucom.2021.03.068 (Cited on p. 18).

Travaglio, M., Yu, Y., Popovic, R., Selley, L., Leal, N. S., & Martins, L. M. (2021). Links between air pollution and COVID-19 in England. *Environmental Pollution*, *268*, 115859. DOI: 10.1016/j.envpol.2020.115859 (Cited on p. 21).

Ulmer, D., Hardmeier, C., & Frellsen, J. (2022). deep-significance - easy and meaningful statistical significance testing in the age of neural networks. *arXiv preprint*. DOI: 10.48550/arXiv.2204.06815 (Cited on pp. 131, 133).

Vázquez-Canteli, J. R., Ulyanin, S., Kämpf, J., & Nagy, Z. (2019). Fusing tensorflow with building energy simulation for intelligent energy management in smart cities. *Sustainable cities and society*, *45*, 243–257. DOI: 10.1016/j.scs.2018.11.021 (Cited on pp. 10, 25).

Venter, Z. S., Aunan, K., Chowdhury, S., & Lelieveld, J. (2020a). *Air pollution changes during COVID-19 lockdowns* (data visualization). https://nina.earthengine.app/view/lockdown-pollution (last accessed: 2025/03/15). (Cited on p. 21)

Venter, Z. S., Aunan, K., Chowdhury, S., & Lelieveld, J. (2020b). COVID-19 lockdowns cause global air pollution declines. *Proceedings of the National Academy of Sciences*, *117*(32), 18984–18990. DOI: 10.1073/pnas.2006853117 (Cited on p. 21).

Vlahogianni, E. I., Karlaftis, M. G., & Golias, J. C. (2014). Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, *43*, 3–19. DOI: 10.1016/j.trc.2014.01.005 (Cited on p. 85).

Walraven, E., Spaan, M. T., & Bakker, B. (2016). Traffic flow optimization: A reinforcement learning approach. *Engineering Applications of Artificial Intelligence*, *52*, 203–212. DOI: 10.1016/j.engappai.2016.01.001 (Cited on pp. 2, 20, 28).

Wang, L., Geng, X., Ma, X., Liu, F., & Yang, Q. (2019). Cross-city transfer learning for deep spatio-temporal prediction, In *Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization. DOI: 10.24963/ijcai.2019/262. (Cited on pp. 18, 28).

Wang, Y., Liao, W., & Chang, Y. (2018). Gated recurrent unit network-based short-term photovoltaic forecasting. *Energies*, *11*(8), 2163. DOI: 10.3390/en11082163 (Cited on p. 14).

Wang, Z., Maeda, T., Hayashi, M., Hsiao, L.-F., & Liu, K.-Y. (2001). A Nested Air Quality Prediction Modeling System for Urban and Regional Scales: Application for High-Ozone Episode in Taiwan. *Water, Air, and Soil Pollution*, *130*(1), 391–396. DOI: 10.1023/A:1013833217916 (Cited on p. 21).

Wasicek, A. (2020). The future of 5G smart home network security is micro-segmentation. *Network Security*, *2020*(11), 11–13. DOI: 10.1016/S1353-4858(20)30129-X (Cited on p. 1).

Wei, H., Zheng, G., Yao, H., & Li, Z. (2018). Intellilight: A reinforcement learning approach for intelligent traffic light control, In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Association for Computing Machinery. DOI: 10.1145/3219819.3220096. (Cited on pp. 20, 28).

Wen, C., Liu, S., Yao, X., Peng, L., Li, X., Hu, Y., & Chi, T. (2019). A novel spatiotemporal convolutional long short-term neural network for air pollution prediction. *Science of The Total Environment*, *654*, 1091–1099. DOI: 10.1016/j.scitotenv.2018.11.086 (Cited on pp. 22, 24, 28).

White, G., Zink, A., Codecá, L., & Clarke, S. (2021). A digital twin smart city for citizen feedback. *Cities*, *110*, 103064. DOI: 10.1016/j.cities.2020.103064 (Cited on pp. 10, 25).

World Health Organization. (2021). *WHO global air quality guidelines: Particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide*. World Health Organization. https://apps.who.int/iris/handle/10665/345329. (last accessed: 2025/03/15). (Cited on pp. 20, 111)

World Health Organization. (2022a). *Ambient (outdoor) air pollution* (website). https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health (last accessed: 2025/03/15). (Cited on pp. 109, 111)

World Health Organization. (2022b). *Ambient air pollution attributable deaths* (dataset). https://www.who.int/data/gho/data/indicators/indicator-details/GHO/ambient-air-pollution-attributable-deaths (last accessed: 2025/03/15). (Cited on p. 109)

World Health Organization. (2022c). *Annual mean PM2.5 concentration in urban areas* (data visualization). https://www.who.int/data/gho/data/indicators/indicator-details/GHO/concentrations-of-fine-particulate-matter-(pm2-5) (last accessed: 2025/03/15). (Cited on p. 21)

Xie, P., Li, T., Liu, J., Du, S., Yang, X., & Zhang, J. (2020). Urban flow prediction from spatiotemporal data using machine learning: A survey. *Information Fusion*, *59*, 1–12. DOI: 10.1016/j.inffus.2020.01.002 (Cited on p. 17).

Xu, J., Rahmatizadeh, R., Bölöni, L., & Turgut, D. (2017). Real-time prediction of taxi demand using recurrent neural networks. *IEEE Transactions on Intelligent Transportation Systems*, *19*(8), 2572–2581. DOI: 10.1109/TITS.2017.2755684 (Cited on pp. 18, 28, 102).

Xu, S., Zhang, R., Ma, H., Ekanayake, C., & Cui, Y. (2024). On vision transformer for ultra-short-term forecasting of photovoltaic generation using sky images. *Solar Energy*, *267*, 112203. DOI: 10.1016/j.solener.2023.112203 (Cited on p. 150).

Yang, D. (2019). A guideline to solar forecasting research practice: Reproducible, operational, probabilistic or physically-based, ensemble, and skill (ROPES). *Journal of Renewable and Sustainable Energy*, *11*(2), 022701. DOI: 10.1063/1.5087462 (Cited on pp. 12, 56, 62, 64).

Yang, Y., Liu, Y., Zhang, Y., Shu, S., & Zheng, J. (2025). DEST-GNN: A double-explored spatio-temporal graph neural network for multi-site intra-hour PV power forecasting. *Applied Energy*, *378*, 124744. DOI: 10.1016/j.apenergy.2024.124744 (Cited on pp. 15, 28).

Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., Gong, P., Ye, J., & Li, Z. (2018). Deep multi-view spatial-temporal network for taxi demand prediction,

In *Proceedings of the AAAI Conference on Artificial Intelligence*. DOI: 10.1609/aaai.v32i1.11836. (Cited on pp. 19, 28, 86, 102).

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting, In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. DOI: 10.24963/ijcai.2018/505. (Cited on p. 151).

Zahid, M., Chen, Y., Khan, S., Jamal, A., Ijaz, M., & Ahmed, T. (2020). Predicting risky and aggressive driving behavior among taxi drivers: Do spatio-temporal attributes matter? *International journal of environmental research and public health*, *17*(11), 3937. DOI: 10.3390/ijerph17113937 (Cited on p. 1).

Zaręba, A., Krzemińska, A., Kozik, R., Adynkiewicz-Piragas, M., & Kristiánová, K. (2022). Passive and active solar systems in eco-architecture and eco-urban planning. *Applied Sciences*, *12*(6). DOI: 10.3390/app12063095 (Cited on p. 61).

Zeiler, M. D., Krishnan, D., Taylor, G. W., & Fergus, R. (2010). Deconvolutional networks, In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR.2010.5539957. (Cited on p. 53).

Zhang, J., Pan, X., Li, M., & Philip, S. Y. (2016). Bicycle-sharing system analysis and trip prediction, In *2016 17th IEEE international conference on mobile data management (MDM)*. IEEE. DOI: 10.1109/MDM.2016.35. (Cited on p. 28).

Zhang, J., Zheng, Y., Qi, D., Li, R., & Yi, X. (2016). DNN-based prediction model for spatio-temporal data, In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. DOI: 10.1145/2996913.2997016. (Cited on pp. 18, 28, 86).

Zhang, N., Zhang, Y., & Lu, H. (2011). Seasonal autoregressive integrated moving average and support vector machine models: Prediction of short-term traffic flow on freeways. *Transportation Research Record*, *2215*(1), 85–92. DOI: 10.3141/2215-09 (Cited on pp. 17, 28).

Zhao, L., Wang, J., Liu, J., & Kato, N. (2019). Routing for Crowd Management in Smart Cities: A Deep Reinforcement Learning Perspective. *IEEE Communications Magazine*, *57*(4), 88–93. DOI: 10.1109/MCOM.2019.1800603 (Cited on pp. 11, 25).

Zhou, Y., Chang, F.-J., Chang, L.-C., Kao, I.-F., & Wang, Y.-S. (2019). Explore a deep learning multi-output neural network for regional multi-step-ahead air quality forecasts. *Journal of Cleaner Production*, *209*, 134–145. DOI: 10.1016/j.jclepro.2018.10.243 (Cited on pp. 22, 24, 28).

Zlatev, Z., & Dimov, I. (2006). *Computational and numerical challenges in environmental modelling*. Elsevier. (Cited on p. 22).

Open-source code and data repositories

This annex provides an overview of the code and data repositories associated with this thesis. The code repositories are hosted on GitHub and the data ones are in Zenodo. The materials presented here support data processing and analysis, enabling the reproducibility of results. In the following, we describe each repository with its contents, access details, and relevant metadata. Table A.1 summarizes all repositories and classifies them based on the use case and resource type. We employ Python alongside common data science libraries (e.g., NumPy, Pandas, Matplotlib, Keras, etc.). Each GitHub repository hosts a requirements file for Python 3.8.

Table A.1: Summary of open-source code and data repositories.

Use case	Resource	Contents	URL
Solar irradiance	Code	Scripts to reproduce experiments, graphs, and result tables.	https://github.com/iipr/solar-irradiance
	Code	Scripts to reproduce experiments.	https://github.com/jlrisco/caide/
Mobility demand	Code	Scripts to reproduce experiments, graphs, and result tables.	https://github.com/iipr/mobility-demand
	Data	Mesh-grid and other supporting datasets utilized.	DOI: 10.5281/zenodo.5166839
Air quality	Code	Scripts to reproduce experiments, graphs, and result tables.	https://github.com/iipr/air-quality
	Data	Mesh-grid and other supporting datasets utilized.	DOI: 10.5281/zenodo.7764528

A.1 First use case: Solar irradiance

Prado-Rujas et al. (2020) hosts the code to reproduce the experiments from Chapter 5, except those from Section 5.4.5, which can be found in a different repository (Risco-Martin et al., 2024).

- **Data analysis and transformation**:
  - **Mesh-grid and metadata**: Due to distribution restrictions (Sengupta & Andreas, 2010), we opted not to distribute the data directly. Instead, we prepared a Jupyter Notebook named `etl-data.ipynb` (hosted on Prado-Rujas et al., 2020) guiding through the data analysis and transformation steps of our methodology. This Notebook can be used to obtain the 8×8 and 10×10 mesh-grid datasets of standardized GHI and CSI together with the relevant metadata.
  - **Supplementary files**: The pyranometers geolocations are located in the file `stations.txt`. In addition, we include statistics of the sensor measurements before and after estimating missing values when $\tau = 1$ s.
- **ST-SIF and baselines definition**:
  - **ST-SIF**: The ST-SIF model implementation appears on the `modelUtils.py` file as `convLstm0`.
  - **Baseline models**: The FC, RNN, LSTM, and BiLSTM baselines are `fc2`, `rnn0`, `lstm0`, and `biLstm0` of the same file, respectively.
- **Experiments implementation**:
  - **Error metric**: The training and testing of the ST-SIF and baseline models is accomplished with the scripts `learner.py`, `trainUtils.py`, `modelUtils.py`, and `plotUtils.py`. Additionally, `deep_playground.py` allows the user to choose model parameters and launch the training interactively.
  - **Robustness test**: The function `__robustness_test()` in `learner.py` implements Algorithm 1, which evaluates the robustness of the models.
  - **Integration into CAIDE**: The experiments of the integration of the ST-SIF into CAIDE are available on Risco-Martin et al. (2024).
- **Results and graphs**:
  - **Error metrics**: The file `results.ods` lists all the studied models, their parameters, skill, and RMSE.
  - **Irradiance maps**: The folder `irradiance-map` presents examples of

animated solar irradiance mesh-grids evolving throughout the day.

- **Model graphs**: The folder `model-graph` collects the Keras-exported graphs of the studied models.
- **Sample predictions**: The folder `prediction-sample` includes plots of truth versus predicted values of three baseline models, similar to Figure 5.5.
- **Skill boxplots**: The folder `skill-boxplot` contains the skill boxplots of the studied models, similar to Figure 5.6.
- **Skill heatmaps**: The folder `skill-map` presents the skill of the studied models as heatmaps, similar to Figure 5.7.
- **Robustness test**: The folder `robustness-test` hosts graphs similar to Figure 5.8 for variations of the ST-SIF and other related models. The folder with the same name under `tables` gathers the execution results of the robustness tests, i.e., for each repetition: mean skill, skill per horizon, RMSE per horizon, and randomly excluded sensors.

A.2 Second use case: Mobility demand

The code to reproduce the experiments from Chapter 6 is hosted on GitHub (Prado-Rujas, Serrano, et al., 2021c). The datasets from this use case are also available at Zenodo (Prado-Rujas, Serrano, et al., 2021a).

- **Data analysis and transformation**:
  - **Mesh-grid and metadata**: The original raw data comprises taxi trips (City of Chicago, 2024) and bicycle rides (Lyft, 2024b). In this case, we also prepared a Jupyter Notebook named `data-etl.ipynb` (hosted on Prado-Rujas, Serrano, et al., 2021c) for the data analysis and transformation steps. The Notebook `data-eda.ipynb` explores and visualizes the data prior to its transformation. In addition, the 90×60 mesh-grid datasets of normalized taxi and bike rides are available on Prado-Rujas, Serrano, et al. (2021a).
  - **Supplementary files**: The coordinates of the taxi zone centroids and bicycle stations are also available on Prado-Rujas, Serrano, et al. (2021a). In addition, we include the holiday dataset of Chicago for the studied period (2013 –2020). Weather data is not shared due to distribution restrictions imposed by Meteosim. Statistics of the sensor measurements are located in the folder `results` (Prado-Rujas, Serrano, et al., 2021c) for $\tau = 15$ min and 1 day.

- **ST-MDF and baselines definition**:
  - **ST-MDF**: The ST-MDF model implementation and its variants appear on the `modelUtils.py` file as `ST-MDF`.
  - **Baseline models**: The LSTM, BiLSTM, persistence, and naive baselines are `lstm`, `biLstm`, `persistence`, and `naive` of the same file, respectively.
- **Experiments implementation**:
  - **Error metric**: The training and testing of the ST-MDF and baseline models is accomplished with the scripts `learner.py`, `trainUtils.py`, `modelUtils.py`, and `plotUtils.py`. The script `launcher.py` enables users to define multiple hyperparameter combinations. It can also automate training and testing by running the previous scripts as a background job, as follows: `CUDA_VISIBLE_DEVICES='X' nohup python launcher.py > /dev/null 2> ../tmp/exp_Y.err < /dev/null &`, where `X` and `Y` represent the GPU and experiment identifier, respectively.
  - **Robustness test**: The function `__robustness_test()` of the `learner.py` script implements the robustness experiment similarly as for the previous use case.
  - **Extensibility and integrability**: These experiments are also implemented by training and testing the adequate models of `modelUtils.py`, varying the model modules or input shape.
- **Results**:
  - **Error metrics**: The file `results.ods` lists the studied models, their parameters, and aggregated errors. In the table, *MMWC* stands for "Mean across taxi zones/bike racks of the MAE Weighted of trip Counts, per forecast horizon". Similarly, *MRWC* is the "Mean across taxi zones/bike racks of the RMSE Weighted of trip Counts, per forecast horizon. In addition, a null learning rate implies that the default value for that Keras optimizer was employed.
  - **Robustness, integrability, and n_x**: The `results-analysis.ipynb` Notebook prepares the results of the experiments corresponding to robustness, integrability, and impact of temporal window width.
- **Graphs**:
  - **Mobility demand animated heatmap**: The video file located in the `graphs` folder is an example of an animated taxi pick-up mesh-grids evolving throughout two days.

  - **Chicago maps**: The folder `graphs` hosts several figures depicting the Chicago metropolitan area with the mesh-grid, sensor locations, census tracts, and convex hulls.

A.3 Third use case: Air quality

Lastly, another GitHub repository gathers the code to reproduce the experiments from Chapter 7 (Prado-Rujas et al., 2023c). As for the second use case, the dataset used in the experiments can be found at Zenodo (Prado-Rujas et al., 2023b).

- **Data analysis and transformation**:
  - **Mesh-grid and metadata**: The original raw data comprises measurements of eleven pollutants (Ayuntamiento de Madrid, 2024). In this case, we also prepared a Jupyter Notebook named `data-etl.ipynb` (hosted on Prado-Rujas et al., 2023c) to guide data analysis and transformation. The Notebook `data-eda.ipynb` explores and visualizes the data before its transformation. In addition, the 35×30 mesh-grid datasets of raw, normalized, and standardized air pollutants for linear and nearest-neighbor interpolation are available on Prado-Rujas et al. (2023b).
  - **Supplementary files**: The coordinates of the air quality sensors are available on Prado-Rujas et al. (2023c). The holiday dataset is not included in our repository since it is available at Comunidad de Madrid (2023). Due to distribution restrictions, we do not share weather data. Statistics of the sensor measurements are located in the folder `variables-stats` (Prado-Rujas et al., 2023c).
- **ST-AQF and baselines definition**:
  - **ST-AQF**: The ST-AQF model implementation and its variants appear on the `modelUtils.py` file as `ST-AQF`.
  - **Baseline models**: The LSTM, Conv3D, and persistence baselines are `lstm`, `conv3d`, and `persistence` of the same file, respectively.
- **Experiments implementation**:
  - **Error metric**: The training and testing of the ST-AQF and baseline models is accomplished with the scripts `learner.py`, `trainUtils.py`, `modelUtils.py`, and `plotUtils.py`. Additionally, `launcher.py` allows for the definition of multiple combinations of hyperparameters and for deploying the training and testing of those models as background jobs, using the previous scripts.

- **Robustness test**: The function `__robustness_test()` of the `learner.py` script implements the robustness experiment in a similar fashion as for the first use case.
- **Extensibility and integrability**: These experiments are also implemented by training and testing the adequate models of `modelUtils.py`, varying the model modules or input shape.

- **Results**:
  - **Error metrics**: The file `model-results.xlsx` lists the studied models, their parameters, and aggregated errors (nRMSE and MAE).
  - **Robustness, integrability, extensibility, and n_x**: The Notebooks `results-analysis.ipynb` and `model-comparison.ipynb` prepare and analyze the results of the presented experiments.
- **Graphs**:
  - **Model graphs**: The folder `graphs` collects the Keras-exported graphs of the ST-AQF model variants.
  - **Robustness test**: The folder `experiment-robustness` hosts graphs similar to Figure 7.7 for the ST-AQF model.
  - **Integrability**: The folder `experiment-integrability` includes graphs similar to Figure 7.10 for variations of the ST-AQF model.
  - **Extensibility**: The folder `experiment-extensibility` collects graphs similar to Figures 7.8 and 7.9 for variations of the ST-AQF model.
  - **Impact of temporal window width**: The folder `experiment-n_x` hosts graphs similar to Figure 7.6 for variations of the ST-AQF model.
  - **Madrid maps**: The folder `graphs` hosts several figures depicting Madrid together with the mesh-grid, sensor locations, and convex hulls.

Alphabetical index