UNIVERSITAT POLITÈCNICA DE CATALUNYA

DOCTORAL THESIS

# On the data quality improvement of air pollution monitoring low-cost sensor networks using data-driven techniques

*Author:*
Pau Ferrer Cid

*Supervisors:*
Dr. Jorge García Vidal
Dr. José M. Barceló Ordinas

*A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy*

*in the*

Statistical Analysis of Networks and Systems
Computer Architecture Department

March 10, 2023

*Science is a wonderful thing*
*if one does not have to earn one's living at it.*

Albert Einstein

# Abstract

Nowadays, authorities monitor the concentrations of regulated air pollutants in order to assist in decision-making processes, e.g., for the implementation of traffic restrictions, and mitigate the effects of air pollution. For this purpose, they deploy high-precision instrumentation, the cost of which makes the number of sensors deployed over a region very low. The advent of air pollution low-cost sensors (LCSs) has opened up the possibility of complementing the authorities' instruments with more measurement points. Unfortunately, LCSs present inaccuracies, which makes it difficult to include them in a regulated way for decision-making processes of authorities.

In recent years, enabling technologies such as the internet of things (IoT) and machine learning (ML) have allowed the improvement of the data quality of LCSs. Therefore, this thesis is devoted to the improvement of the *data quality* of air pollution monitoring LCS networks focusing on two aspects; *i)* the improvement of data quality at node level using *ML-based sensor calibration*, and *ii)* the improvement of the sensor network data quality by using measurements from the network sensors with a *graph-based* approach.

In the first part of the thesis, the improvement of the data quality of individual sensors is investigated. First, it is evaluated how the *sensor sampling* affects the representativeness of the samples. Then, the use of ML techniques, both linear and nonlinear, for the *in-situ calibration* of LCSs is analyzed. The *in-situ* sensor calibration task can be seen as a supervised ML learning problem, so techniques such as multiple linear regression (MLR) or support vector regression (SVR) are evaluated. The evaluation shows how nonlinear techniques improve the quality of pollution estimates significantly. In addition, given the inaccuracies present in LCSs and the difference that exists from one sensor to another of the same manufacturer, the inclusion in the calibration of multiple sensors measuring the same pollutant is investigated. Thereby, the proposed *multisensor calibration* approach based on ML results in increased calibration accuracy.

The second part of the thesis focuses on the quality of the data reported by a sensor network once deployed over an area. A *graph-based* approach is proposed to describe the existing relationships between sensors using a *graph topology* and represent the network measurements as signals defined on the graph, as realized in the graph signal processing (GSP) field. First, different techniques have been evaluated to correctly learn the relationships between sensors in a network that can contain both LCSs and high-precision nodes. The most suitable option has proven to be the data-driven GSP model based on signal smoothness. Then, different *signal reconstruction* techniques coupled with the graph have been studied in order to reconstruct pollution measurements reported by different sensors in a network. Kernel-based techniques and those based on the weights of the Laplacian have been the most effective ones. Once these main components have been studied, a *graph-based data reconstruction framework* has been proposed for different post-processing applications that appear in LCS networks, e.g., missing value imputation and virtual sensing. The results have shown how this framework allows for dealing with a wide variety of applications and scenarios that can occur in this context with precision. Finally, another important aspect of this type of network has been addressed, which is the detection of outliers. The *Volterra graph-based outlier de-*

*tection* (VGOD) has been proposed, using a graph learned from the data and a signal reconstruction model based on the Volterra series, to detect and locate outliers. Therefore, the proposed algorithm has been proven to improve the monitoring and maintenance of heterogeneous air pollution sensor networks by identifying abnormal measurements and malfunctioning sensors. All in all, this graph-based approach can be seen as a monitoring tool to evaluate and maintain the sensor network data quality over the lifetime of a deployment.

# Resumen

Hoy en día, las autoridades vigilan las concentraciones de contaminantes atmosféricos regulados para ayudar en los procesos de toma de decisiones, por ejemplo, en la aplicación de restricciones de tráfico, y mitigar los efectos de la contaminación atmosférica. Para ello, despliegan instrumentación de alta precisión, cuyo coste hace que el número de sensores desplegados en una región sea muy reducido. La aparición de sensores de contaminación atmosférica de bajo coste (LCS) ha abierto la posibilidad de complementar los instrumentos de las autoridades con más puntos de medición. Desafortunadamente, los LCS presentan imprecisiones, lo que dificulta su inclusión de forma regulada en los procesos de toma de decisiones de las autoridades.

En los últimos años, tecnologías como el internet de las cosas (IoT) y el aprendizaje automático (ML) han permitido mejorar la calidad de los datos de los LCSs. Por lo tanto, esta tesis está dedicada a la mejora de la *calidad de los datos* de las redes de LCS de contaminación atmosférica, centrándose en dos aspectos: *i)* la mejora de la calidad de los datos a nivel de nodo utilizando *calibración de sensores basada en ML*, y *ii)* la mejora de la calidad de los datos de la red de sensores utilizando mediciones de los sensores de la propia red mediante un *enfoque basado en grafos*.

En la primera parte de la tesis se investiga la mejora de la calidad de los datos de los sensores de forma individual. En primer lugar, se evalúa cómo afecta el *muestreo de los sensores* a la representatividad de las muestras. A continuación, se analiza el uso de técnicas ML, tanto lineales como no lineales, para la *calibración in-situ* de LCSs. La tarea de calibración de sensores in-situ puede considerarse un problema de aprendizaje de ML supervisado, por ello se evalúan técnicas como la *multiple linear regression* (MLR) o *support vector regression* (SVR). La evaluación muestra cómo las técnicas no lineales mejoran significativamente la calidad de las estimaciones de contaminación. Además, dadas las imprecisiones presentes en los LCS y la diferencia que existe de un sensor a otro del mismo fabricante, se investiga la inclusión en la calibración de múltiples sensores que miden el mismo contaminante. De este modo, el enfoque propuesto de *calibración multisensor* basado en ML permite aumentar la precisión de la calibración.

La segunda parte de la tesis se centra en la calidad de los datos medidos por la red de sensores una vez desplegada en un área. Se propone un enfoque *basado en grafos* para describir las relaciones existentes entre los sensores mediante la *topología del grafo* y representar las medidas de la red como señales definidas en el grafo, tal y como se realiza en el campo del *graph signal processing* (GSP). En primer lugar, se han evaluado diferentes técnicas para aprender correctamente las relaciones entre sensores de una red que puede contener tanto LCSs como nodos de alta precisión. La opción más adecuada ha resultado ser el modelo de GSP basado en el *smoothness* de la señal. A continuación, se han estudiado distintas técnicas de *reconstrucción de señal* acopladas al grafo con el fin de reconstruir las medidas de contaminación obtenidas por los distintos sensores de la red. Las técnicas basadas en kernel y las basadas en los pesos del Laplaciano han sido las más efectivas. Una vez estudiados estos componentes, se ha propuesto un *framework de reconstrucción de datos basado en grafos* para diferentes aplicaciones de post-procesado que aparecen en las redes de LCSs, por ejemplo, la imputación de valores perdidos y los sensores virtuales. Los resultados han mostrado cómo este *framework* permite abordar con precisión una amplia variedad de aplicaciones y escenarios que pueden darse en este contexto. Por último, se ha investigado otro aspecto importante de este tipo de redes, la detección de valores atípicos. Se ha propuesto el algoritmo *Volterra graph-based outlier detection* (VGOD), que utiliza un grafo aprendido a partir de los datos y un modelo de reconstrucción de señal basado en las series de Volterra, para detectar y localizar valores atípicos. Se ha demostrado que el algoritmo propuesto mejora la monitorización y el mantenimiento de redes heterogéneas de sensores de contaminación atmosférica al identificar medidas anómalas y sensores que funcionan mal. En definitiva, este enfoque basado en grafos puede considerarse una herramienta de supervisión para evaluar y mantener la calidad de los datos de una red de sensores durante un despliegue.

# Acknowledgements

I would like to highlight the importance of all those who have made the development of this doctoral thesis possible. A doctoral thesis is both a technical and an emotional journey, so different types of support are necessary.

First of all, I would like to thank my supervisors, both Prof. Jorge García Vidal and Prof. Jose María Barceló Ordinas, for having helped me during all these years and for giving me the opportunity to work with them. It has always been a joy to learn with you and to brainstorm ideas.

Secondly, I would like to thank the unconditional support of my partner, it has been years of emotions, both frustration and joy to see the different works published and the progress made. So, it would not have been possible to overcome the most delicate moments without the amount of encouragement given. Last but not least, I would like to thank the continuous support received from my family during all these years. For all those who support me day by day and for those who, in spite of not being present, accompany me day by day.

# Contents

# List of Figures

# List of Tables

# Nomenclature

**List of Acronyms and Abbreviations**

| | |
|---|---|
| **ANN** | Artificial Neural Network |
| **CO** | Carbon Monoxide |
| **CRMSE** | Centered-Root-Mean-Squared Error |
| **CV** | Cross-Validation |
| **DC** | Duty Cycle |
| **EC** | Electrochemical |
| **EPA** | Environmental Protection Agency |
| **EU** | European Union |
| **FPR** | False Positive Rate |
| **GDFT** | Graph Discrete Fourier Transform |
| **GMRF** | Gaussian Markov Random Field |
| **GSO** | Graph Shit Operator |
| **GSP** | Graph Signal Processing |
| **GSR** | Graph Signal Reconstruction |
| **IGDFT** | Inverse Graph Discrete Fourier Transform |
| **IoT** | Internet of Things |
| **KNN** | K-Nearest Neighbors |
| **KRR** | Kernel Ridge Regression |
| **Lap.Int.** | Laplacian Interpolation |
| **LCS** | Low-Cost Sensor |
| **LGF** | Linear Graph Filter |
| **LMMSE** | Linear Minimum Mean Squared Error |
| **LOF** | Local Outlier Factor |
| **MAE** | Mean Absolute Error |
| **MBias** | Mean Bias |
| **MLR** | Multiple Linear Regression |
| **ML** | Machine Learning |
| **MOX** | Metal-Oxide |
| **NO** | Nitrogen Monoxide |
| $\mathbf{NO_2}$ | Nitrogen Dioxide |
| **NPGF** | Nonlinear Polynomial Graph Filter |
| $\mathbf{O_3}$ | Tropospheric Ozone |
| **PLS** | Partial Least Squares |
| $\mathbf{PM_{10}}$ | Particulate Matter 10$\mu$m |
| **RF** | Random Forest |
| **RH** | Relative Humidity |
| **RMSE** | Root-Mean-Squared Error |
| **RSS** | Residual Sum of Squares |
| $\mathbf{R^2}$ | Coefficient of Determination |
| **SVR** | Support Vector Regression |
| **TPR** | True Positive Rate |
| **TV** | Total Variation |
| **T** | Temperature |
| **VGOD** | Volterra Graph-Based Outlier Detection |
| **VIF** | Variance Inflation Factor |
| **VOC** | Volatile Organic Compound |
| **WHO** | World Health Organization |

**Notation**

| | |
|---|---|
| $x$ | Scalar variable |
| $\mathbf{x}$ | Vector variable |
| $\mathbf{X}$ | Matrix variable |
| $x_i$ | Vector i-th value |
| $X_{ij}$ | Matrix (i,j)-th value |
| $\mathbf{1}\|\mathbf{0}$ | Vector of ones \| zeros |
| $\mathbf{I}_N$ | Identity matrix of size N×N |
| $\|x\|$ | Absolute value of a scalar |
| $\\|\mathbf{x}\\|_2$ | Vector $l_2$-norm |
| $\\|\mathbf{x}\\|_1$ | Vector $l_1$-norm |
| $\mathbf{x}^\mathsf{T}\|\mathbf{X}^\mathsf{T}$ | Vector \| Matrix transpose |
| $\mathbf{X}^{-1}\|\mathbf{X}^\dagger$ | Matrix inverse \| Moore-Penrose pseudoinverse |
| $\mathbf{X}\odot\mathbf{Y}$ | Hadamard product of matrices $\mathbf{X}$ and $\mathbf{Y}$ |
| $\\|\mathbf{X}\\|_\mathsf{F}$ | Frobenius norm of a matrix |
| $\\|\mathbf{X}\\|_1$ | $L_{1,1}$-norm of a matrix |
| $diag(\mathbf{x})$ | Square matrix whose diagonal elements are $\mathbf{x}$ |
| $tr(\mathbf{X})$ | Trace of a matrix |
| $cond(\mathbf{X})$ | Condition number of a matrix |
| $det(\mathbf{X})$ | Determinant of a matrix |
| $vec(\mathbf{X})$ | Column-wise vectorization of a matrix |
| $\mathscr{X}$ | Set variable |
| $\|\mathscr{X}\|$ | Cardinality of a set |
| $\mathscr{G}$ | Graph |
| $\mathscr{V}\|\mathscr{E}$ | Graph set of nodes \| edges |
| $\mathbf{S}$ | Graph Shift Operator |
| $\mathbf{L}\|\mathbf{A}$ | Laplacian \| Adjacency matrix |
| $\mathbf{W}\|\mathbf{D}$ | Weight \| Degree matrix |
| $\boldsymbol{\Theta}$ | Precision matrix |
| $\mathscr{N}(x_i)$ | Set of neighboring nodes of node $x_i$ |

# 1
# INTRODUCTION

*We do not inherit the earth from our ancestors.*
*We borrow it from our children.*

Native American Proverb

Nature is suffering severe damage during the last years and climate change is already a reality. The enormous development of industries, as well as the high use of fossil fuels, makes air pollution a concerning problem. Nevertheless, air pollution not only affects nature but also the citizens of large and small cities and animals, causing respiratory problems and cardiovascular diseases among others. Nowadays, technological advances are providing new tools to reduce pollution and mitigate its effects. In fact, the Internet of Things (IoT) technologies in conjunction with air pollution *low-cost sensors* (LCSs) can enable the efficient measurement of pollution levels at different scales, both at city and national levels. This monitoring is a core element for government agencies to take action and raise citizen awareness of pollution.

This thesis provides the knowledge to improve the low-cost sensor calibration via machine learning (ML) techniques, and thus the accuracy of these sensors which is the weakness of this technology. It shows how a branch of artificial intelligence, *machine learning*, can enhance the calibration of such sensors. In addition, this thesis explains the findings on how to analyze sensor networks from the point of view of *graph theory* and *graph signal processing* (GSP). The findings of this work show how graphs can be a good tool to describe highly complex networks such as air pollution sensor networks. Thus, showing how the graph can be inferred, and how it can help in conjunction with signal reconstruction techniques to maintain the data quality of the network data, as well as help in the detection of outlying measurements. This chapter introduces the classical approach towards air pollution monitoring, then it explains the main challenges of air pollution monitoring with LCSs and the issues related to sensor networks for air pollution monitoring. This chapter poses the different research questions that are answered throughout the thesis and finally, the contributions of this thesis are listed.

## 1.1. OUTDOOR AIR POLLUTION

OUTDOOR air pollution is becoming a serious problem for the environment and public health, with various studies pointing out the different harmful consequences for human beings, such as cardiovascular problems, respiratory problems, or neuronal problems [1, 2]. In fact, the World Health Organization (WHO[1]) reports that about 4.2 million people die each year from exposure to air pollution. The WHO states that 9 out of 10 people periodically breathe air that exceeds the limits established by organizations, being the least developed countries the most affected. The impact of pollution not only affects people's health but also contributes to climate change. According to NASA[2], different record figures have been reached during the last years, e.g., carbon dioxide ($CO_2$) levels are at their highest since 650000 years ago, nineteen of the planet's hottest years have occurred since 2000, among others. One of the main causes is the greenhouse effect, being carbon dioxide one of the main gas contributors, as well as methane ($CH_4$) and tropospheric ozone ($O_3$). The effects of climate change are undeniable and many nations have agreed on objectives for the coming years (2030 agenda[3]) to reduce pollution and slow or even reverse the effects of climate change. These data are very alarming, given the impact on health and the clear impact on climate change. Therefore, the main objective of the governing authorities is to undertake policies to reduce the levels of air pollution in order to mitigate the negative effects on health and nature. Thus, different criteria and thresholds have been defined to evaluate air quality and to be able to implement the necessary measures in each case.

There are many air pollutants, but the ones most studied and most present in our daily life are; particulate matter ($PM_x$), tropospheric ozone, nitrogen dioxide ($NO_2$), and sulfur dioxide ($SO_2$). The $PM_x$ corresponds to different types of fine particles in suspension in the air, they can be classified in particles of less than 10 microns $PM_{10}$ and particles of 2.5 microns or less ($PM_{2.5}$). $O_3$ is an indirect pollutant formed by chemical reactions of gaseous precursors such as volatile organic compounds (VOC) and nitrogen oxides in conjunction with sunlight. $NO_2$ is basically generated in combustion processes, which is why its levels can be higher in cities and large avenues with heavy traffic. In order to mitigate the effects of these pollutants, it is first important to be able to monitor the levels of pollution. As a reference, the WHO defines different thresholds (Table 1.1) that should not be exceeded, and if they are exceeded, authorities should take action to mitigate pollution levels.

**Table 1.1:** Air pollution thresholds defined by the WHO in its latest guidelines [3].

| | | |
|---|---|---|
| $PM_{10}$ | 15 $\mu$gr/m$^3$ (annual mean) | 45 $\mu$gr/m$^3$ (daily mean) |
| $PM_{2.5}$ | 5 $\mu$gr/m$^3$ (annual mean) | 15 $\mu$gr/m$^3$ (daily mean) |
| $O_3$ | 100 $\mu$gr/m$^3$ (8-hour mean) | |
| $NO_2$ | 10 $\mu$gr/m$^3$ (annual mean) | 200 $\mu$gr/m$^3$ (hourly mean) |
| $SO_2$ | 40 $\mu$gr/m$^3$ (daily mean) | |

Since different organizations such as the European Union (EU), the U.S. Environmental Protection Agency (EPA), or the WHO have established thresholds above which pollution levels are considered harmful, the different countries' authorities have built

[1]https://www.who.int
[2]https://climate.nasa.gov/
[3]https://sdgs.un.org/2030agenda

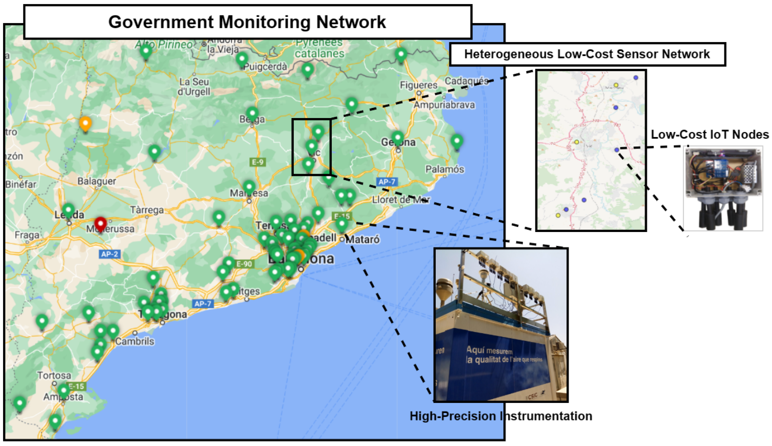

Government Monitoring Network
Heterogeneous Low-Cost Sensor Network
Low-Cost IoT Nodes
High-Precision Instrumentation

**Figure 1.1:** Catalonia map with the air quality monitoring network. On the bottom is an example of government-deployed high-precision instrumentation, a reference station. On the right, an IoT node mounting LCSs.

air quality monitoring networks to monitor the concentration levels at different scales. These networks consist of very precise but at the same time very expensive instrumentation[4], in the order of 100k euros. Given their high economic cost, the number of instruments deployed by the government over the different regions is scarce, and although they may cover some major streets in large cities, there are no pollution measurements on a finer-grain scale. This can be a problem given that the governments need more fine-grained air pollution measurements in order to put in action measures to effectively mitigate the effects of air pollution events, detect air pollution hotspots, raise citizen awareness, or perform better air pollution forecasting among others. As an example, the following Figure 1.1 shows the different reference stations (government instrumentation) deployed by the Spanish government to monitor air pollution in the area of Catalonia. As it can be seen, many reference stations are concentrated in the larger cities, yet the number of stations is too few to properly monitor pollution levels. Another clear example of the need for pollution monitoring in order to take measures are several European cities such as Paris[5], where the new energy laws of 2015 allow the local authorities to place restrictions on the circulation of vehicles in different parts of the city, as a consequence of pollution episodes.

**Definition 1** ***Reference stations*** *are stations equipped with high-precision instrumentation, capable of measuring ground-truth*[6] *levels of pollutants with high accuracy. They are usually assembled by governments and used in a regulated manner according to EU guidelines, in the case of European countries, and the number of deployed stations is usually scarce due to their high price.*

[4]Throughout the thesis we refer to this government instrumentation as high-precision instrumentation, reference instrumentation, or reference stations interchangeably.

[5]http://www.airuse.eu/

[6]We refer to the pollution concentrations measured by reference stations as true concentrations or ground-truth concentrations interchangeably.

***Approach to overcome reference station limitations:*** An alternative approach proposed years ago that allows an increase in the resolution of pollution measurements is the use of LCSs in conjunction with IoT-enabling technologies, explained in the next section 1.2. Therefore, LCSs can be deployed in conjunction with government instruments to form a more extensive sensor network and provide more fine-grained measurements.

## 1.2. LOW-COST SENSORS AND THE INTERNET OF THINGS

LOW-COST sensors provide a cheaper alternative to high-precision instrumentation since these sensors can be mounted on nodes featuring IoT technologies, unfortunately, at the cost of questionable data quality. Because of that, the quality of the data provided by such sensors has been in the spotlight during the last years [4], and many studies have been focused on evaluating the quality of these sensors to check whether they can meet the data quality requirements to include them in a regulated manner [5]. Nevertheless, LCSs seem to be the most feasible technology that can help increase the resolution of air pollution monitoring networks [6]. Thus, in conjunction with reference stations, LCSs can be deployed to obtain measurements at a finer scale to obtain more specific restriction policies.

**Definition 2** *An air pollution **low-cost sensor** is a cheap sensing device designed to measure air pollution concentration levels. However, given their low-cost nature, these sensors suffer from imprecision, leading to data quality problems. Although there is a wide variety of sensors, with different purposes, generally they have been designed for consumer or industrial applications so they may not meet the necessary government data quality requirements.*

There are different LCS technologies, among the most important are; metal-oxide sensors (MOX), electrochemical sensors (EC), and optical sensors. MOX sensors measure gas concentrations by measuring the change in resistance due to the effect of the pollutant. The EC sensors work by gas diffusion, when the gas reaches the working electrode of the sensor an electrochemical reaction is produced. Finally, optical sensors are widely used for measuring particulate matter concentrations and proceed by measuring how light scatters due to the presence of particles.

In recent years, there has been a growing trend and interest in the deployment of sensing nodes equipped with LCSs for various applications such as air quality monitoring [5]. This is due to the rise of IoT-enabling technologies, so it has become possible to create nodes capable of capturing, processing, and transmitting data to databases using microcontrollers and internet connection. Therefore, both technologies can enable the capture, processing, transmission, and storage of air pollution measures. The collected air pollution data can be used for dissemination to increase citizen awareness of pollution [7, 8], it can also be used to place gas emission restrictions to mitigate pollution effects, forecast future concentrations, detect air pollution hotspots, for early warning purposes in cases of extreme contamination, and it can even be used for research purposes. Table 1.2 explains different use cases for air pollution monitoring networks. Figure 1.2 shows different possible scenarios for the applications described, where sensor networks can be deployed into a city to deal with these applications. An example

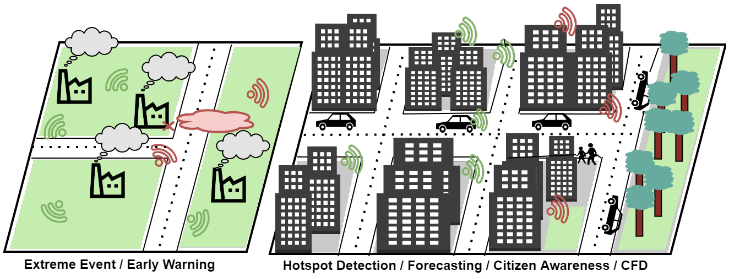

Extreme Event / Early Warning
Hotspot Detection / Forecasting / Citizen Awareness / CFD

**Figure 1.2:** Example of monitoring sensor networks' final applications, such as early warning, citizen awareness, or air pollution forecasting.

of an IoT node that incorporates low-cost sensing technology to capture air pollution measurements is the Captor node [9]. Figure 1.3 shows a version of the *Captor* node, its main architecture, where MOX sensors measure $O_3$ concentrations, an Arduino works as processing unit to process and transmit the data, and a modem 3G enables internet connection and data transmission to a central database.

**Table 1.2:** Examples of different final applications for an air pollution monitoring sensor network.

| **Applications** | **Goal** |
|---|---|
| ★*Air pollution forecasting* | Air pollution monitoring and forecasting is important to carry out measures to prevent and mitigate pollution effects, as well as to foresee future pollution episodes. |
| ★*Detection of air pollution hotspots and early warning* | Monitoring is also important to detect air pollution hotspots, where thresholds defined by different organizations are frequently exceeded, so that actions can be taken to reduce air pollution events or to alert citizens. It is also important to detect extreme pollution events in order to warn first-responders and citizens to take effective measures in time. |
| ★*Creation of air pollution maps* | Measurements can be used to extrapolate the observed values to unobserved locations, obtaining an air pollution map for an area. |
| ★*Citizen Awareness* | It is also important for citizens to be aware of the most polluted sites in order to modify their routine habits to avoid possible harmful effects. |
| ★*Enhancement of CFD models* | As a novel application, the introduction of sensor network information can help refine computational fluid dynamics (CFD) models for the creation of very accurate simulation models. |

***Limitation of LCSs:*** Although LCSs present an economically feasible alternative to high-cost instrumentation, their accuracy is limited and has been the subject of study in recent years. Therefore, in order to implement the various use cases mentioned above using low-cost heterogeneous sensor networks for air quality monitoring, it is essential to address the problem of *data quality* in LCSs.

LCSs present several data quality problems. Firstly, they can be inaccurate as the manufacturer usually provides the sensors calibrated in chambers, which is a controlled environment, and secondly, this calibration may not be correct depending on the sensor's deployment location. In fact, several studies support that sensors should be cali-

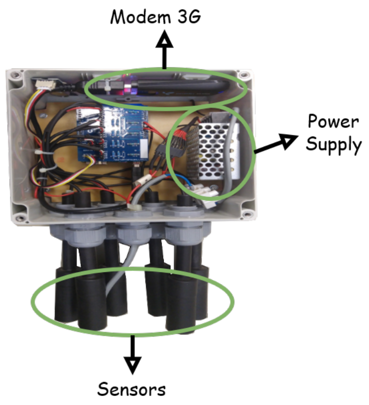


**Figure 1.3:** Captor node architecture; an IoT sensing node mounting several low-cost air pollution sensors and a processing unit capable of capturing and transmitting data.

brated in conditions similar to those of its deployment, so requiring an *in-situ* calibration [10, 11]. There are other factors that influence the quality of the sensors, such as possible aging or drift problems. Given the dependencies of the sensors on the environment they are deployed in, and the varying nature of the environmental conditions, it is advisable to recalibrate the sensors to avoid drift or bias problems, either periodically or opportunistically [12].

To improve the accuracy of LCSs for air pollution monitoring, it is common to reformulate the sensor calibration problem as a supervised ML problem, where sensor values are compared with ground-truth values from a reference station where they are collocated.

**Definition 3** *A **machine learning-based in-situ sensor calibration** is based on the fact that given a set of sensor measurements* $\mathbf{x} \in \mathbb{R}^N$*, where* $N$ *is the number of measurements, and a set of ground-truth values* $\mathbf{y} \in \mathbb{R}^N$*, a function* $f : \mathbb{R} \to \mathbb{R}$ *is learned so that* $y_i \approx f(x_i); i = 1, .., N$*. Different machine learning algorithms assume different characteristics for the regression function.*

Before sensor calibration can be formulated as a supervised ML problem, representative measurements need to be obtained from LCSs at a temporal granularity equal to that of the reference station in order to synchronize their values. In addition, these measurements may go through different *pre-processing* stages, such as sensor sampling, filtering and subsequent aggregation to a desired temporal granularity. Some works indicate that the sensor sampling frequency can impact the power consumption of the node or even the quality of the measures. This way, we can pose the first research question of this thesis:

**(R.Q.1.1):** *Which pre-processing steps are required prior to the calibration of a low-cost sensor? What effect does sensor sampling have on the calibration quality and energy consumption of the sensing node?*

In recent years, several authors have proposed and evaluated the use of different machine learning techniques for *in-situ* LCS calibration, improving the sensors' data quality [13]. Both linear and nonlinear models have been tested [14–16], but none of the methods has been proven to be superior. In this specific field, we can already define the next research question we wish to answer:

**(R.Q.1.2):** *How much do nonlinear methods improve the calibration of ozone sensors compared to linear methods? How much data do they need ? How do they work in the long term?*

This question poses the second goal of the thesis, which is evaluating the performance of different *supervised ML* algorithms (linear and nonlinear) for LCS calibration. This way, the results show which is the best calibration method depending on the calibration time available and can be useful for future practitioners.

LCSs are not only inaccurate but also exhibit variability from unit to unit [8]. Thus, there has been some attempt to use different sensors of the same pollutant to improve pollution estimates [17]. Since different sensors may not contain exactly the same information, and sometimes replicated sensors are mounted in sensing devices to improve the devices' robustness, sensor fusion is plausible. Here is where the third question we want to answer arises:

**(R.Q.1.3):** *Does having replicated sensors on a node improve the calibration accuracy? Is it possible to perform a calibration based on data fusion and machine learning?*

This question raises the third goal of the thesis, which is to perform *data fusion-based sensor calibration* by applying a ML method over an array of sensors. Indeed, since a sensing node can include more than a sensor, given their low price, the calibration can be performed taking advantage of all of them.

All the efforts of this thesis are based on improving the accuracy and data quality of LCSs, both in their *in-situ* calibration and their subsequent deployment in sensor networks to obtain more reliable measurements and to be able to carry out the different use cases of this type of network.

## 1.3. SENSOR NETWORKS AND GRAPH SIGNAL PROCESSING

THE main purpose of calibrating the sensors is their subsequent deployment to monitor the pollution over an area of interest. These nodes, which mount LCSs, are equipped with IoT technology so that the sensors can form a sensor network and the captured data can be transmitted from node to node or to a centralized database. The low economic cost of this type of sensor can lead to the deployment of large amounts of sensors, thus dramatically increasing the spatial resolution of the official air pollution monitoring network. Therefore, a sensor network may be able to obtain measurements at street level, building blocks, and other finer or coarser scales. This finer level of data resolution may allow the use of more customized policies depending on the characteristics of each pollution zone. Following this line, Motlagh *et al.* [6] explain how the future of air pollution monitoring is based on the formation of sensor networks, where LCSs and high-precision instrumentation coexist. Since LCSs are deployed to complement and benefit from the reference stations available in an area, the study of this kind of heterogeneous network -high-precision and low-precision nodes- can be quite complex. For instance, as reference stations always measure the ground-truth levels of pollution, they can provide assistance in detecting if a LCS is malfunctioning or in correcting sensor drifts [18].

There are a variety of studies that make use of geospatial models in order to extrapolate the values measured by the sensing nodes to obtain pollution maps [19]. Geostatistical models such as Kriging or Inverse distance weighting (IDW) are models that operate over a continuous field, and therefore have been used to create pollution maps and other applications such as correction or recalibration of sensors deployed in a network [18]. All these models have as a common feature the assumption of spatial correlation, or that the geographical distance between observations governs the correlation between measurements. However, this can be a limitation for heterogeneous networks, where LCSs of different accuracy coexist with high-accuracy nodes.

It may not only be of interest to create contamination maps from sensors, but the sensors may be distributed in critical points, where contamination may have a great impact, and it is not necessary to extrapolate those values but to know the contamination as accurately as possible in those points. Therefore, a very important aspect in the deployment and use of sensor networks is the *quality of the data* they provide in order to carry out the different use cases seen in the previous section 1.2. For instance, we can calibrate *in-situ* a few LCSs, and then deploy them in schools, avenues, and places where we have a clear interest in measuring air pollution. Indeed, we can use the sensor network (information from other sensors) to mitigate common sensor errors such as; lack of data due to sensor malfunction, sensor precision loss, or estimating air pollution concentrations in places where there is no physical sensor (i.e., virtual sensor). Thus, sensors can benefit from the other sensors jointly deployed in order to impute missing data, recalibrate sensors or create virtual sensors. To this end, given the heterogeneity of these networks, a data-driven approach to model these networks and carry out the above-mentioned applications seems to be a good approach.

The recent emergence of the GSP field translates the ideas and methods from classical signal processing to signals defined over graphs [20, 21]. Therefore, defining some operations such as signal filtering, signal reconstruction, signal convolution, or clustering over the vertices of a graph. In this way, a graph signal is defined as the map $x : \mathcal{V} \to \mathbb{R}$ [22], where $\mathcal{V}$ is the set of vertices of the graph, and $x_i$ represents the signal value at the $i$-th node. Some operations like the graph discrete Fourier transform (GDFT) or low-pass filtering can be applied using the network topology. Given the high flexibility that graphs exhibit in modeling highly complex structures, the conjunction of graphs, machine learning, and graph signal processing techniques seems to be a very good option for air pollution sensor networks. In this way, a graph can define the existing relationships between nodes, which need not be based on the distance between them, and benefit from machine learning techniques that make use of the graph topology to perform some tasks (e.g. signal reconstruction) to carry out different applications and maintain the quality of the network data.

**Definition 4** *A graph $\mathcal{G}$ is a mathematical structure defined by the triplet $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of vertices of the graph, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges defined*

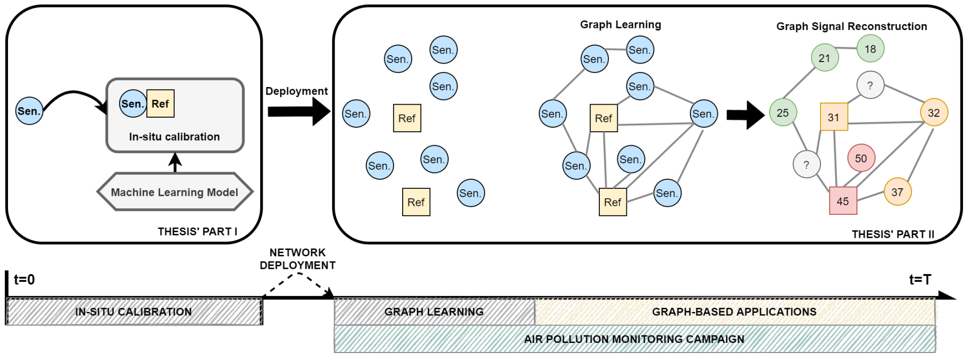


**Figure 1.4:** The different chronological steps in a sensor network deployment analyzed throughout the thesis. Ultimately, the goal is to increase the sensors' accuracy via *in-situ* calibration, afterwards, when the sensor network is deployed for different purposes (e.g., air pollution forecasting or citizen awareness) the goal is to maintain the network data quality and thus its reliability.

*as* $\mathcal{E} = \{e_{ij} : i, j \in \mathcal{V} \wedge W_{ij} \neq 0\}$*, and the weight matrix* $\mathbf{W} \in \mathbb{R}^{N\times N}$ *defines the edges' weights between vertices. So, a graph describes a set of vertices that are somehow related.*

Maintaining data quality through ML techniques and signal processing over the graph representing a sensor network is investigated in this thesis to provide solutions to sensor network data quality issues during the deployment. This second part of the thesis unravels several research questions that open the door to the joint use of pollution sensing networks, graphs, and ML. Figure 1.4 shows the chronological order of the different steps that are the object of study in this thesis. The first step is the *in-situ* calibration of the sensors, then the sensors are deployed over an area of interest forming a sensor network in order to start the air pollution monitoring campaign. Then, the graph $\mathcal{G}$ describing the network can be inferred from the data or using a distance-based function. Once the underlying relationships between the sensors are learned, the neighboring information of the sensors (described by the graph) can be used to perform different applications to maintain the quality of the air pollution measures.

The first step in the use of graph-based techniques is to obtain the graph $\mathcal{G}$ that describes the sensor network. There are several techniques to learn the underlying relationships between the different nodes of a network [23, 24]. Statistical techniques such as graphical lasso, which aims to estimate the precision matrix of the data, techniques based on the geographical distances between sensors using a distance-based similarity function, or GSP techniques that assume the smoothness of the data over the graph. A clear example of learning the graph from the data is when we have three sensors, for instance, two placed in schools and one in an avenue. Clearly, even if a school is nearby the avenue the data coming from the two schools can follow similar patterns since they have similar conditions.

**(R.Q.2.1):** *What different techniques are best suited to infer a graph for a sensor network? Can neighboring nodes be used to reconstruct sensor measurements using the graph and signal reconstruction techniques?*

These questions related to how to effectively infer a graph for air pollution monitoring and the signal reconstruction performance provide the necessary information to decide which algorithm to use when learning a graph. The goal here is to evaluate the best method for graph inference, as well as the ability to reconstruct a node's signal given its neighboring information. Then, given the relationship learned between the different nodes of the network, solutions to common applications of sensor deployments can be provided by reconstructing the signal from the different nodes, applications such as; imputing missing values, creating virtual sensors or sensor fusion. This graph-based approach allows to use information from neighboring nodes, Figure 1.5, which may contain useful information for the signal reconstruction task.

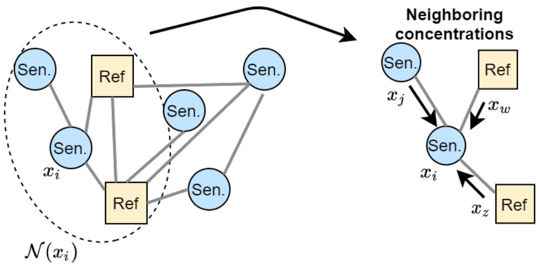


**Figure 1.5:** Notion of node neighborhood. This picture depicts the idea of using the graph topology to flow information from similar nodes to reconstruct sensor measurements.

In fact, the main objective of using a graph-based approach is to use the information provided by the network topology and the relationships between sensors. That is, a node $x_i$[7] will benefit from the measurements of the sensors in its neighborhood $\mathcal{N}(x_i)$. Thus, we can define the neighborhood of a sensor $x_i$ as the set of nodes connected to it $\mathcal{N}(x_i) = \{j : e_{ij} \in \mathcal{E}\}$. Figure 1.5 shows the neighborhood definition for a sensor $x_i$ that includes nodes $x_j$, $x_z$, $x_w$ and the idea that the information from these neighboring nodes can be used to reconstruct the signal from node $x_i$.

**Definition 5** *Graph signal reconstruction (GSR), also known as graph signal recovery, is the task of finding the regression function $f : \mathcal{V} \to \mathbb{R}$ that interpolates the graph signal at a set of unobserved/missing nodes $\{x_u : u \in \mathcal{U} \wedge \mathcal{U} \subset \mathcal{V}\}$ given a set of observed nodes $\{x_m : m \in \mathcal{M} \wedge \mathcal{M} \subset \mathcal{V}\}$.*

GSR is a key tool in this graph-based setting, therefore, we can evaluate the use of different signal reconstruction techniques and check how they perform on air pollution networks.

**(R.Q.2.2):** *How do different graph signal reconstruction techniques perform in air pollution monitoring networks? What is specific about them? What problems can they pose?*

[7]Throughout the thesis $x_i$ is used to denote a sensor/node as well as its measurement (or the value of a graph signal at that node) interchangeably.

This study provides the necessary information to select a GSR model to be used in conjunction with the graph learning process to reconstruct air pollution sensor measurements in this type of network.

Once we know how to infer a graph for a sensor network and how to use neighboring information for signal reconstruction, the third graph-related question rises:

(**R.Q.2.3**): *For which applications can the signal reconstruction using the sensor network data be used?*

The aim of this question is to apply the graph methodology to solve some common sensor post-processing applications mentioned above; missing data imputation, virtual sensing, or data fusion, to help maintain and enhance the quality of the network data. Hence, the thesis provides knowledge about graph inference, GSR techniques, and applications to solve using the proposed graph-based data reconstruction framework.

Finally, this thesis addresses one of the most important problems in a sensor network, which is the detection of outliers or malfunctioning sensors [25]. The data need to be reliable since it can be consumed for applications, e.g., to put in action prevention measures, so the detection of sensors with measurements that deviate from normality is very important. Furthermore, since the signal reconstruction of a node uses the signals from neighboring nodes, it is important to detect which sensors may be malfunctioning or which are unreliable measurements in order not to be involved in the reconstruction of a node's signal. Besides, for monitoring and maintenance operations, a methodology is necessary to detect malfunctioning sensors.

(**R.Q.2.4**): *Can we approach one of the most important data quality problems, such as the detection and location of outliers or erroneous sensors, using the graph information?*

This research question leads directly to the development of an outlier detection algorithm to improve the monitoring of heterogeneous sensor networks by detecting anomalous measurements and faulty sensors.

Summarizing, while the first part of the thesis is focused on improving the sensor data quality via *in-situ* sensor calibration through machine learning, the second part of the thesis is focused on enhancing the sensor network deployment stage and maintaining the data quality using graphs and machine learning.

## 1.4. THESIS STRUCTURE, CHALLENGES, AND CONTRIBUTIONS

THIS section describes the structure of the thesis and the resulting publications that answer the different questions posed in the previous section. The different sections are arranged chronologically, i.e., pre-processing, sensor calibration, and sensor network modeling, and a direct relationship between the different publications and the research question they answer is shown. In order to make the document understandable to the reader, the different parts that address the different questions to be solved contain the technical background necessary to understand the different results obtained and the models used. This way, an expert reader can refer directly to the chapter of interest and skip the technical background if necessary.

### 1.4.1. THESIS STRUCTURE

This thesis is structured in a very simple way, as it follows the chronological steps of the deployment of an air pollution measurement network; pre-processing, *in-situ* sensor calibration, and assessment of the sensor network data quality using graphs. Figure 1.4 summarizes the different studied aspects in the chronological order of a sensor network deployment.

- ***Introduction:*** The introduction gives a brief explanation of the existing problems with air pollution LCSs and sensor networks. Different questions are posed to advance the state-of-the-art and the different publications derived from these questions are described.

- ***PART I: Machine learning-based in-situ sensor calibration:*** The first part comprises the investigation of the first task in the deployment of a sensor network, which is the *in-situ* calibration of sensors. First, the different pre-processing stages are studied, as well as their impact on the sensor calibration. Then, a calibration based on machine learning is studied and different techniques, both linear and nonlinear, are evaluated in terms of calibration accuracy. Thirdly, a multisensor data fusion calibration using more than one sensor measuring the same phenomenon is studied. Machine learning is used to merge the sensor measurements and the proposed procedure shows an increase in calibration accuracy compared to using only one sensor. In this way, having more than one sensor would not only increase robustness but also accuracy.

- ***PART II: Graph-based analysis of air pollution sensor networks:*** The second part of the thesis focuses on the study of sensor networks to maintain the network data quality. This is the second natural step after the *in-situ* calibration, deploying the sensors forming a network and using the network measurements to detect events and maintain the quality of the data provided by the different sensors. For the analysis of this type of network, the use of a discrete model such as graphs is proposed. Thus, graph signal processing is used to learn the graph and apply machine learning techniques on top of the graph topology to perform signal reconstruction tasks. These techniques are shown to be useful for maintaining network data quality in different scenarios, e.g., missing value imputation or outlier detection.

### 1.4.2. SUMMARY OF CHALLENGES

Table 1.3 shows a brief summary of the different challenges posed in both the *in-situ* sensor calibration and the graph-based sensor network analysis. In addition, the proposals and the investigations that answer each of the formalized research questions are shown, as well as the technologies used to address these questions.

### 1.4.3. LIST OF PUBLICATIONS

This section shows different published articles that have resulted directly from the results of the Ph.D. and answer the different questions posed above. In addition, articles indirectly related to the thesis in which we have collaborated are also detailed.

**Table 1.3:** Summary of challenges and proposals addressed in this thesis.

| Area | Challenge | | Proposal | Technology |
|---|---|---|---|---|
| In-situ Calibration | Low-cost sensor pre-processing and impact of sensor sampling scheme | R.Q.1.1 | Study the sensor pre-processing pipeline and evaluate the impact of the sensor sampling | ML |
| | Low-cost sensor calibration | R.Q.1.2 | Compare linear and nonlinear machine learning algorithms for calibration | ML |
| | Use of replicated sensors in the sensor calibration | R.Q.1.3 | Machine learning-based sensor fusion via linear and nonlinear supervised algorithms | ML |
| Graph-Based Analysis of sensor networks | Can we represent an air pollution sensor network as a graph ? | R.Q.2.1 | Evaluation of graph learning techniques (data-driven and distance-based) for sensor networks | GSP & ML |
| | Which signal reconstruction techniques can be used in an air pollution paradigm ? | R.Q.2.2 | Compare the performance of three graph signal reconstruction techniques for air pollution networks | GSP & ML |
| | Can network data be used to solve common applications that arise in air pollution networks ? | R.Q.2.3 | Propose a graph-based data reconstruction framework for missing imputation, virtual sensing and data fusion | GSP & ML |
| | Since data quality is of major concern, how can we detect outliers ? | R.Q.2.4 | Propose an unsupervised graph-based decision process for outlier detection | GSP & ML |

## JOURNAL PAPERS

### Machine Learning-Based *In-Situ* Sensor Calibration

1. Ferrer-Cid, P., Barcelo-Ordinas, J. M., & Garcia-Vidal, J. (2022). **Raw Data Collected From $NO_2$, $O_3$ And NO Air Pollution Electrochemical Low-Cost Sensors**. *Data in Brief*, 45, 108586. [First JIF in June 2023]

   This paper describes the high-frequency electrochemical low-cost sensor data obtained during the January-April of 2021. The data set consists of two sensing nodes measuring $O_3$, $NO_2$, and NO at a sampling frequency of 0.50 Hz using Alphasense electrochemical sensors. This data set allows for exploring the impact of the sampling frequency on the sensor calibration. This data set has been used in chapter 2.

2. Barcelo-Ordinas, J. M., Ferrer-Cid, P., Garcia-Vidal, J., Viana, M., & Ripoll, A. (2021). **H2020 project CAPTOR dataset: Raw data collected by low-cost MOX ozone sensors in a real air pollution monitoring network**. *Data in Brief*, 36, 107127. [First JIF in June 2023]

   This article describes in detail all the data collected for the H2020 Captor project during the 2017 tropospheric ozone campaign. Thus, open data is encouraged by making public all the data necessary for other researchers and institutions to use the published data and support their investigations. Part of this data set has been used for the experiments of chapters 3, 4, 5, 6, 7, and 8.

3. Ferrer-Cid, P., Paredes-Ahumada, J.A, Allka, Xhensilda, Guerrero-Zapata, M., Barcelo-Ordinas, J.M., & Garcia-Vidal. J. (2023). **A Data-Driven Framework For air Quality Sensor Networks**. Submitted to *IEEE Communications Magazine*.

   This article shows the need to improve the quality of data from LCSs in order to feed user applications with reliable data, chapter 1. It is proposed a layered data-driven framework to describe different tasks that need to be performed at different levels to ensure data quality. This article includes several of the techniques described and elaborated throughout the thesis.

4. Ferrer-Cid, P., Garcia-Calvete J., Main-Nadal A., Ye Z., Barcelo-Ordinas, J. M., & Garcia-Vidal J. (2022). **Sampling Trade-Offs in Duty-Cycled Systems for Air Quality Low-Cost Sensors**. *Sensors MDPI.* [Q2, IF=3.847]

   This article studies one of the most important aspects of pre-processing for low-cost air pollution sensors, sensor sampling (R.Q.1.1). The effect of sensor sampling on both calibration quality and energy consumption in terms of duty cycle is studied. The results are explained in chapter 2.

5. Ferrer-Cid, P., Barcelo-Ordinas, J. M., Garcia-Vidal, J., Ripoll, A., & Viana, M. (2019). **A comparative study of calibration methods for low-cost ozone sensors in IoT platforms**. *IEEE Internet of Things Journal*, 6(6), 9563-9571. [Q1, IF=9.936]

   This paper answers question (R.Q.1.2) showing the performance of linear (multiple linear regression) and nonlinear machine learning methods (K-nearest neighbors, random forest, and support vector regression) for the *in-situ* calibration of air pollution low-cost sensors. The experiments and results are explained in chapter 3.

6. Ferrer-Cid, P., Barcelo-Ordinas, J. M., Garcia-Vidal, J., Ripoll, A., & Viana, M. (2020). **Multisensor data fusion calibration in IoT air pollution platforms**. *IEEE Internet of Things Journal*, 7(4), 3124-3132. [Q1, IF=9.471]

   This investigation goes one step further by showing the possibility of using replicated sensors by means of sensor fusion using machine learning for sensor calibration. It answers the question (R.Q.1.3) and shows the potential improvements of using several sensors for the prediction of ozone concentration levels. The methodology and results derived from the multisensor data fusion calibration are explained in chapter 4.

## Graph-Based Analysis of Air Pollution Sensor Networks

7. Ferrer-Cid, P., Barcelo-Ordinas, J. M., & Garcia-Vidal, J. (2021). **Graph learning techniques using structured data for IoT air pollution monitoring platforms**. *IEEE Internet of Things Journal.* [Q1, IF=10.238]

   Question (R.Q.2.1) is answered in this publication by showing the use of different graph learning techniques for homogeneous and heterogeneous low-cost sensor networks. Graph signal processing-based techniques are shown to be the best for graph inference and posterior graph signal reconstruction. The methodological aspects as well as the results of applying this graph-based approach are detailed in chapter 5.

8. Ferrer-Cid, P., Barcelo-Ordinas, J. M., & Garcia-Vidal, J. (2022). **Graph Signal Reconstruction Techniques for IoT Air Pollution Monitoring Platforms**. *IEEE Internet of Things Journal.* [Q1, IF=10.238]

   This publication presents different graph signal reconstruction techniques that can be applied to IoT air pollution platforms for signal reconstruction and data quality maintenance purposes (R.Q.2.2). This paper evaluates the graph-based framework coupled with different graph signal reconstruction techniques as well as shows a clustering-based methodology to overcome scalability issues and shows a drift compensation application of the graph signal reconstruction. The results are explained partially in chapter 6 and 5.

9. Ferrer-Cid, P., Barcelo-Ordinas, J. M., & Garcia-Vidal, J. (2022). **Data reconstruction applications for IoT Air Pollution sensor networks using graph signal processing**. *Elsevier, Journal of Network and Computer Applications.* [Q1, IF=7.574]

   This publication presents how different applications such as missing value imputation, virtual sensing, and data fusion, can be approached using the graph-based methodology shown earlier, answering the question (R.Q.2.3). This paper investigates how the graph-based framework allows for dealing with a wide variety of online post-processing applications that may appear in this type of network. The results are explained in chapter 7.

10. Ferrer-Cid, P., Barcelo-Ordinas, J. M., & Garcia-Vidal, J. (2022). **Volterra Graph-Based Outlier Detection for Air Pollution Sensor Networks**. *IEEE Transactions on Network Science and Engineering.* [Q1, IF=5.033]

    Question (R.Q.2.4) is answered in this publication by proposing a graph-based unsupervised outlier detection method with localization capabilities for air pollution sensor networks. The different components of the outlier detection process and the different experiments conducted on air pollution sensor networks are explained in chapter 8.

## RELATED PAPERS

- Barcelo-Ordinas, J. M., Ferrer-Cid, P., Garcia-Vidal, J., Ripoll, A., & Viana, M. (2019). **Distributed multi-scale calibration of low-cost ozone sensors in wireless sensor networks**. Sensors, 19(11), 2503.

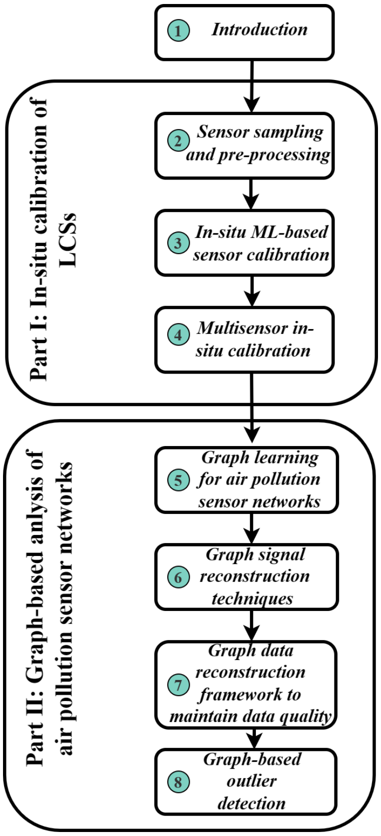
  This previous study analyzes the *in-situ* calibration of a set of low-cost metal-oxide ozone sensors using multiple linear regression. In addition, the effect of the changing climatic conditions during sensor deployment is analyzed. Finally, a technique based on geostatistical Kriging is developed to mitigate sensor drift using information from nearby reference stations.

- Ferrer-Cid, P., Barcelo-Ordinas, J. M., & Garcia-Vidal, J. (2022). **Regularized Multidomain Data-Driven Signal Reconstruction**. Submitted to Pattern Recognition Letters.

  This article opens a related branch of research where the use of orthonormal transform matrices to reconstruct signals, in this case, signals from air pollution sensors, is studied. A regularized solution is elaborated that takes into account two transforms that can provide complementary information.

# I

# Machine Learning-Based In-Situ Sensor Calibration


1 Introduction
Part I: In-situ calibration of LCSs
2 Sensor sampling and pre-processing
3 In-situ ML-based sensor calibration
4 Multisensor in-situ calibration
Part II: Graph-based anlysis of air pollution sensor networks
5 Graph learning for air pollution sensor networks
6 Graph signal reconstruction techniques
7 Graph data reconstruction framework to maintain data quality
8 Graph-based outlier detection

# 2 LOW-COST SENSOR PRE-PROCESSING

*Everything is theoretical impossible,*
*until it is done.*

Robert A. Heinlein

The main purpose of the *in-situ* calibration of sensors is to improve their data quality as well as to translate raw measurements into air pollution concentrations. Nevertheless, before machine learning (ML) techniques can be applied, there are different aspects of sensor signal pre-processing that can affect the quality of the subsequent calibration as well as there exist other metrics of interest in the deployment of internet of things (IoT) nodes that mount low-cost sensors (LCSs). LCSs are often mounted on IoT nodes that participate in a sensor network to measure pollution concentrations in an area of interest. In order to obtain the measurements, LCSs need to be calibrated *in-situ* and for this purpose, their data need to be pre-processed by first taking measurements from the sensor, filtering them, and aggregating them to the periodicity of the reference instrumentation to perform the *in-situ* calibration. In addition, attention has recently been paid to the power consumption of the sensing nodes that mount this type of sensor, either to reduce consumption or to maximize the lifetime of the monitoring campaign in the case of battery-powered nodes. Thus, duty cycle strategies, which are defined by the sampling frequency of the LCSs, can be applied to these nodes to maximize the lifetime of the nodes.

The purpose of this chapter is to describe the different pre-processing steps commonly used for sensor calibration and measurement acquisition, as well as to study how sensor sampling frequency affects both the calibration quality and the sensing node power consumption. The chapter is organized as follows; section 2.1 introduces the main aspects of LCS pre-processing for air pollution, section 2.2 describes the different pre-processing stages for air pollution estimation using LCSs. Section 2.3 describes the data sets used and the experiments carried out. Finally, section 2.4 concludes the chapter. This chapter presents the findings made in the article "*Sampling Trade-Offs in Duty-Cycled Systems for Air Quality Low-Cost Sensors*", MDPI Sensors, [26].

## 2.1. Low-cost sensor pre-processing for air pollution estimation

Governments measure air pollution in order to introduce prevention and mitigation measures. These measurements are obtained by means of high-precision instrumentation, called reference stations, whose high price makes the number of active stations in certain regions scarce. LCSs have revolutionized the paradigm of air pollution monitoring because of their low price, but at the expense of the quality of their data. Therefore, most research activities have focused on the evaluation and improvement of their data quality [4, 8]. Given the progress in improving the quality of data from these sensors, it is considered that the future of air quality monitoring lies in heterogeneous sensor networks where reference stations and LCSs coexist [6]. Moreover, studies have been carried out to verify whether LCSs can obtain accurate measurements so that they can be included in a regulated way for air quality monitoring [4, 8, 14].

LCSs often come uncalibrated, or the manufacturer has calibrated them in a chamber with specific characteristics, making in-field calibration necessary. In the field of LCSs, calibration is performed in uncontrolled environments [10, 11]. This calibration is called *in-situ*, as the LCSs are placed next to a reference station (government instrumentation) for a calibration period in order to train a calibration model that improves the data quality of these sensors and provides air pollution estimates [14, 15]. The most applied and studied techniques for *in-situ* calibration have relied on supervised ML models, ranging from linear to nonlinear models [15, 16, 27]. Nevertheless, in order to apply a supervised ML model the sensor signal pre-processing steps are very important as the reference stations usually report hourly data that are the result of an aggregation of samples during this period[1] [14, 28, 29]. Therefore, in order to obtain representative sensor measurements in the same time granularity as the reference station, the sensor must be sampled by the IoT node, these samples must be filtered to avoid outliers and erroneous measurements, and aggregated at the time granularity of the reference instrumentation. Concas *et al.* [12] describe the most critical pre-processing steps for LCSs for air pollution monitoring, including the sensor sampling and aggregation stages.

LCSs are mounted on IoT nodes participating in a sensor network to perform an air quality measurement campaign. Therefore, depending on the needs and characteristics of the node deployment, the nodes may be required to be battery-powered. Indeed, the fact of having battery-powered nodes allows increasing the degree of network heterogeneity, being able to have mobile nodes carried by pedestrians or cars. This plays an important role as it implies the need to implement a duty-cycled sensing system to maximize the lifetime of the nodes and the measurement campaign. The selection of the duty cycle is not trivial given the response times that the different sensors may have before being able to correctly sample the sensor as well as the effect that this sampling may have on the quality of the calibration and the subsequent estimation of concentrations. Usually, the sensor response time is specified by the manufacturer. In fact, most sensor calibration studies assume a high sampling frequency without taking into account energy consumption constraints [29–31]. In addition, it is worth studying whether differ-

[1]European's reference station sample collection and its validity is described in the European directive 2008/50/EC.

ent air pollution phenomena require different sampling frequencies. Some works have mentioned the existence of the trade-off between the sensor sampling frequency and the power consumption[2] in air pollution IoT nodes [32–37].

The problem of data aggregation and duty-cycle in air pollution monitoring nodes is different from the problem of data aggregation in sensor networks [38]. In these air pollution sensing nodes, sensors are calibrated individually and the data aggregation is performed in the nodes' sensing system by aggregating measured samples every time interval, so the aggregation is not performed in the network. For instance, Table 2.1 shows different sampling periods for air pollution LCSs. These samples need to be further aggregated into the time resolution of the reference instrumentation to perform the calibration and to compare the air pollution estimates with the actual air pollution concentrations. Thus, the goal is to minimize the duty cycle of the sensing system while maintaining the sensor calibration quality. Therefore, the sensor sampling strategy, i.e., how many samples and how often are taken, is crucial to obtain good aggregated data.

In the context of air pollution LCSs, the trade-off between the sensor sampling strategy and the node's power consumption has been studied taking into account the quality of the estimations of sensors already calibrated [32, 34–37, 39]. Becnel *et al.* [33] propose a low-cost pollution monitoring station for airborne PM, temperature, relative humidity, light intensity, carbon monoxide, and nitrogen oxide, that performs a duty cycle scheme when the node operates battery-based. However, the calibration analysis is performed only for the case when the node is connected to an unlimited power supply.

**Table 2.1:** Sensor sampling periods used in the literature.

| Work | Pollutants | Sampling Period ($T_s$) |
|---|---|---|
| Mijling *et al.* [30] | $NO_2$ | 1 min |
| Sahu *et al.* [40] | $O_3$, $NO_2$ | 1 min |
| Ali *et al.* [32] | CO,$NO_2$,PM | 1 min |
| Becnel *et al.* [33] | CO,$NO_2$,$PM_1$, $PM_{2.5}$, $PM_{10}$ | 1 min |
| Nowack *et al.* [41] | $NO_2$, $PM_{10}$ | 30 s |
| Bigi *et al.* [16] | NO, $NO_2$ | 20 s |
| De Vito *et al.* [42] | $NO_2$, $O_3$, NO | 20 s |
| Si *et al.* [28] | $PM_{2.5}$ | 6 s |
| Mead *et al.* [43] | NO | 5 s |
| Han *et al.* [29] | $O_3$, $NO_2$, CO, $SO_2$ | 2 s |
| Mead *et al.* [43] | CO, $NO_2$ | 1 s |
| Astudillo *et al.* [31] | $O_3$, CO | 1 s |

Summarizing, in this chapter, we describe the different steps of the sensor data processing pipeline, before going deeper into the calibration techniques using ML (next chapter 3). In addition to explaining these steps, we analyze the trade-offs between sensor sampling, a critical step in battery-powered nodes, the duty cycle, and the quality of the data resulting from sensor calibration.

[2]The terms power consumption and energy consumption are used interchangeably throughout the chapter.

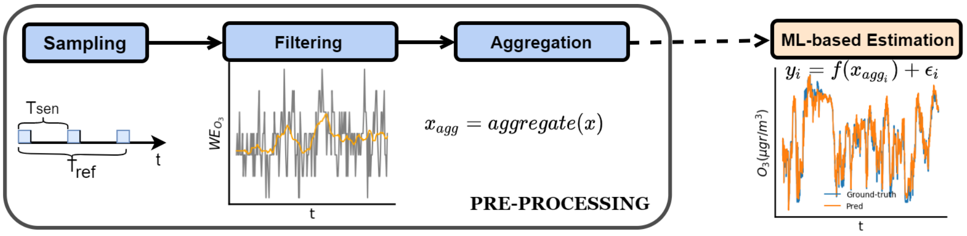


**Figure 2.1:** Sensor data processing pipeline; from sensor sampling to ML estimation. First, sensors are sampled, then the samples collected during this period are filtered to eliminate possible outliers. The resulting samples are aggregated to be sent to the central server. There, the air pollution concentrations are estimated using the calibration models previously trained during the *in-situ* calibration. The trained ML calibration model could also be deployed at the node, obtaining air pollution estimates at the edge.

## 2.2. SENSOR DATA PROCESSING STAGES

IN this section, we explain the different data processing steps needed to produce air pollution estimates from LCS measurements. We focus on the case where LCSs are mounted on IoT nodes that have a power supply, storage capacity, computing resources, and transmission capabilities to transmit measures to a central server where the data is further processed. Therefore, during the different stages, we analyze the operations that can be performed at the *edge*[3] (at the node) and the operations performed at the central server.

The pre-processing steps can be split into; sensor sampling, data filtering, and data aggregation. In this way, a few measurements are sampled from the LCSs, then the measurements are filtered in order to remove outliers, and these measurements are aggregated into the desired time resolution for monitoring or sensor calibration purposes. The aggregation stage is required during the calibration period in order to synchronize the measurement obtained by the LCS and the reference station so an ML model can be trained. Figure 2.1 shows the usual sensor data processing pipeline where the different stages are identified.

In this specific paradigm, the value of the reference station produced every hour is the aggregation of different samples taken during that hour at a frequency that we assume to be higher than the Nyquist frequency corresponding to the time variation of the measured phenomenon. If the sampling frequency of the LCS is also higher than the Nyquist frequency, we can expect the errors in the calibration process to be essentially independent of the sampling frequency of the LCS However, if the frequency sampling of the LCS falls below the Nyquist frequency, we can expect this undersampling to introduce an additional source of error in the calibration process that can have a large impact on the accuracy of the measured values during sensor operation.

Data pre-processing has a big impact on the subsequent representation of the data, it can affect the sensor calibration and the subsequent estimates. As mentioned above, having the data synchronized with reference stations, in the environment where the node will be deployed, allows for calibrating the sensors, and detecting drifts, aging

[3]We emphasize which are the operations that can be performed on the node taking into account possible computation resources constraints.

or outliers [44, 45]. Table 2.2 shows the notation and the sampling parameters used throughout the chapter.

**Table 2.2:** Sensor sampling parameters and their definition.

| Parameter | Definition |
|---|---|
| $T_s$ | Required time to take a sensor measure |
| $T_{sen}$ | Sensing node sampling period |
| $N_s$ | Number of samples taken every sampling period |
| $T_r$ | Sensor response time before valid measurements |
| $T_{ref}$ | Reference data period |
| DC | Sampling strategy duty cycle |
| $T_{on}$ | Time the microcontroller is switched on to collect sensor samples |

The process for obtaining measurements in air pollution sensors is as follows (Figure 2.1):

1. **Sensor sampling:** The sensing node samples the sensor every $T_{sen}$ seconds. For this value to be representative, it may be necessary to wait for a sensor response time $T_r$, and take a sequence of $N_s$ measurements.

2. **Filtering:** Apply a filtering algorithm to remove outliers and smooth the measurements

3. **Aggregation:** Aggregate the filtered measurements into a representative sample every $T_{sen}$ seconds. This step is performed to obtain air pollution measurements at a certain time resolution during the deployment. Besides, measurements can later be synchronized with the reference data for calibration purposes.

The microcontroller can go into sleep mode until it has to collect samples again and switch off the sensing board if necessary. A sensing node can manage an array of sensors, each with its own electronic board, whereby the node reports a vector, each $T_{sen}$, containing the air pollution sensor values (e.g. $NO_2$, $O_3$, NO) and environmental values (e.g. temperature and relative humidity). Two strategies are possible: *i)* a packet is generated with the sample vector every $T_{sen}$, or *ii)* if energy savings are desired in the communication subsystem and the application only needs values every $T_{ref}$, a second aggregation is carried out and transmitted every $T_{ref}$.

Once the sensor measurements are aggregated, these are transmitted to the central server for calibration or air pollution concentration estimation purposes. The reference data and the sensor data need to be synchronized to apply supervised ML models for calibration [14–16]. There exist linear and nonlinear models, such as multiple linear regression (MLR), k-nearest neighbors (KNN), support vector regression (SVR), and artificial neural networks among others. In the case of having the calibration model trained on the cloud, the aggregated samples are used to obtain the final air pollution estimates. Alternatively, an edge computing approach can be followed where the calibration model already trained is stored on the nodes in order to transmit the final air pollution estimates to the cloud. Therefore, in this chapter, we use the MLR, KNN, and SVR to evaluate the effect of the sensor sampling on the calibration and data quality without giving too many details about the calibration models, as this is the main topic of the next chapter 3.

### 2.2.1. Sensor sampling

At this stage, the $N_s$ samples that are part of the representative sample $T_{sen}$ are taken. We focus on taking samples from a single sensor. In the case that the sensing node is responsible for an array of sensors, the microcontroller can run the sampling process in parallel, activating all the sensor boards simultaneously and polling them using a round-robin strategy. Moreover, the sensing node's architecture, several sensors attached to different controller boards, may permit the implementation of specific sampling frequencies for each sensor. To obtain the representative measurement at instant $T_{sen}$ the microcon-

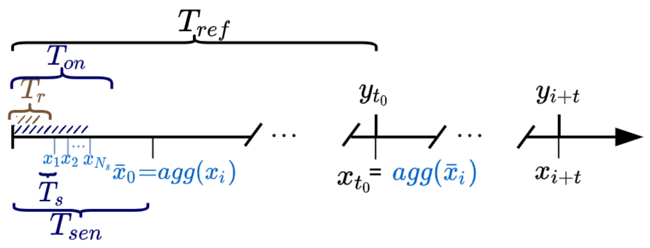


**Figure 2.2:** Assumed sensor sampling scheme. $x_i$ are the sensor measurements while $y_i$ are the reference instrument measurements. Every $T_{sen}$, the controller waits for the sensor response time $T_r$ and collects the $N_s$ samples to be aggregated. Then, these $N_s$ samples are aggregated into measurements every $T_{sen}$ seconds, which can be further aggregated into $T_{ref}$ seconds to synchronize them with the reference values.

troller wakes up the sensor board and takes $N_s$ consecutive samples (Figure 2.2). In this case, the duty cycle is $(N_s \cdot T_s)/T_{sen}$. However, there are air pollution sensors that have a response time of $T_r$, so it is necessary to wait for $T_r$ before collecting valid measurements. Indeed, this response time may vary from one sensing technology to another, and it can be seen as a user-defined parameter specifying the amount of time to wait before collecting a measure to prevent the collection of incorrect measurements. In this case, the duty cycle DC is computed as:

$$\begin{aligned} DC &= \frac{T_{on}}{T_{sen}} \\ T_{on} &= T_r + (N_s \cdot T_s) \\ T_{on} &\le T_{sen} \end{aligned} \tag{2.1}$$

The number of samples $N_s$ that make up the value generated every $T_{sen}$ seconds impact the duty cycle of the sensing node and the quality of the data estimated by the ML algorithm. The value of $T_{sen}$ has an impact on; *i)* the number of packets to transmit, *ii)* the duty cycle of the sensing system (and consequently the node's power consumption), *and iii)* the quality of the values estimated by the ML algorithm. In the experiments section 2.3, we evaluate the impact that the sampling scheme has on the sensor calibration quality and the duty cycle.

### 2.2.2. DATA FILTERING

Once $N_s$ sensor samples have been collected for each sensing node's sampling period $T_{sen}$, these must be filtered in order to remove outliers and smooth the data. The *z-score* is a well-known technique for removing outliers and extreme values [29]:

$$z_i = \frac{|x_i - \bar{\mathbf{x}}|}{s_{\mathbf{x}}} \tag{2.2}$$

Where the measurements are assumed to follow a Gaussian distribution and $\bar{\mathbf{x}}$ is the sample average of the set of measurements $\mathbf{x}$ and $s_{\mathbf{x}}$ is their sample standard deviation. Then, this *z-score* is thresholded to eliminate outliers, a common threshold is 2 standard deviations. Other signal filtering techniques (e.g., moving average or Chebyshev filtering) can be useful to eliminate abrupt changes in the signal and smooth the data trend. For instance, Mijling *et al.* [30] eliminate samples that deviate a given percentage from the sample mean. The filtering process is necessary given that signals sampled from LCSs are noisy and may present outliers, in which case, the subsequent aggregation would be affected and, consequently, the quality of the estimated data would be degraded.

This filtering can be performed at the node since these techniques only involve some basic operations, then, the filtered measurements can be further aggregated. The *z-score* is used as a filtering technique to remove extreme values in the experiments section 2.3.

### 2.2.3. DATA AGGREGATION

The filtered $N_s$ measurements are aggregated into a single measurement every period $T_{sen}$. Different statistics can be used for this aggregation, the most common are the sample mean and median. Nevertheless, these statistics may be affected by the number of measurements aggregated. A small number of samples may affect the representativeness of the aggregated sample and produce biased results. Hence, the number of samples collected in the sampling stage can impact the quality of the aggregation and the subsequent air pollution estimation or sensor calibration. This aggregation can be performed at the node to minimize the number of measurements to be transmitted. In the experiments section 2.3, we use the mean as aggregation technique.

The aggregated measurement can be included in a vector of measurements from all sensors on the node, and can be transmitted to the central server where the ML algorithm can estimate the pollution value with granularity $T_{sen}$. The reference stations, being connected to the power supply, usually take continuous $T_{sen}$ values and aggregate those values into hourly values ($T_{ref}$ = 1h), which are the ones displayed in the applications. If we want to save energy in the communications system, we can do a second aggregation with the $T_{sen}$ values to match the values of the reference stations $T_{ref}$ values. This allows having a heterogeneous network of reference stations and LCS nodes that can spatially measure a pollutant in an area as the two types of nodes have the same time granularity. For sensor calibration purposes the $T_{sen}$ measures need to be aggregated into the $T_{ref}$ granularity to train a supervised ML model.

Nonetheless, nodes with LCSs that have a response time of more than a minute, and that also use batteries and implement a duty cycle to save energy in the sensing subsystem, will not be able to produce $T_{sen}$ values in the same way as reference stations.

## 2.3. EXPERIMENTAL EVALUATION

IN this section, we evaluate the impact of the sampling parameters on the quality of the sensor calibration. We study the impact of the $T_{sen}$ and the number of samples $N_s$ on the sensor calibration quality. We also evaluate how the duty cycle and the quality of the data evolve in the case where the sensor has response time and in the case where it does not have response time $T_r$. The different sampling strategies are simulated by subsampling the raw sensors' signals ($T_{sen} = 2$ s). Since reference values are available hourly, the periods $T_{sen}$ tested are less than or equal to one hour $T_{sen} \leq 1$ h. To do so, we evaluate two different sampling strategies:

(A) Intensive strategy that tries to mimic the reference data gathering process as much as possible by taking $N_s$ samples uniformly every $T_{sen}$ seconds.

(B) Flexible strategy that allows duty cycle implementation where every period $T_{sen}$ a sensor response time $T_r$ is waited and $N_s$ consecutive samples are taken.

In order to evaluate the different strategies, data obtained from the experimental *Captor 4* IoT node, whose sampling frequency allows to simulate different sampling strategies, are used. The characteristics of the node and the data sets used for the experiments are explained in section 2.3.1. The comparison of the two sampling strategies is done by testing different sampling periods for the sensing node $T_{sen}$, as well as different number of samples collected in these periods $N_s$. To carry out this experiment, we use raw two-second signals from the sensors and simulate the different sampling settings by subsampling these raw signals. Once the sensor data have been pre-processed, a 10-fold cross-validation procedure is performed to evaluate the sensor calibration quality, so discussing the resulting goodness-of-fit metrics, duty cycles, and power consumption implications.

The data sets are randomly split into 75% of the data for training and the remaining 25% is used as testing set. The randomized selection is introduced so that the training conditions are representative of the testing, avoiding the implicit effect of out-of-date and inaccurate calibration models [11, 12]. Further explained in next chapter 3.

### 2.3.1. CAPTOR NODE & DATA SETS

For this chapter, we use the data obtained with an experimental air pollution IoT node called *Captor 4*. The Captor is made up of a central processing unit based on a Raspberry PI and different gas monitoring shields attached via an I2C communication bus. The gas monitoring shield integrates two Alphasense sensors controlled by an Arduino Nano microcontroller unit (MCU) responsible for collecting samples from the LCSs attached. Each gas sensor is supplied by the manufacturer with an individual sensor board [46]. The output of the individual sensor board is further amplified by a factor of x2, in order to reduce quantifying errors, and sampled by the analog-to-digital converter of the Arduino Nano MCU.

Two Captor nodes have been deployed for four months at a reference station in Palau Reial, Barcelona (Spain). Captor node labeled as *20001* mounted one Alphasense OX-B431 $O_3$ sensor, one Alphasense NO2-B43F $NO_2$ sensor, one Alphasense NO-B4 NO sensor, and a DHT-22 temperature and relative humidity sensor. Captor node labeled as

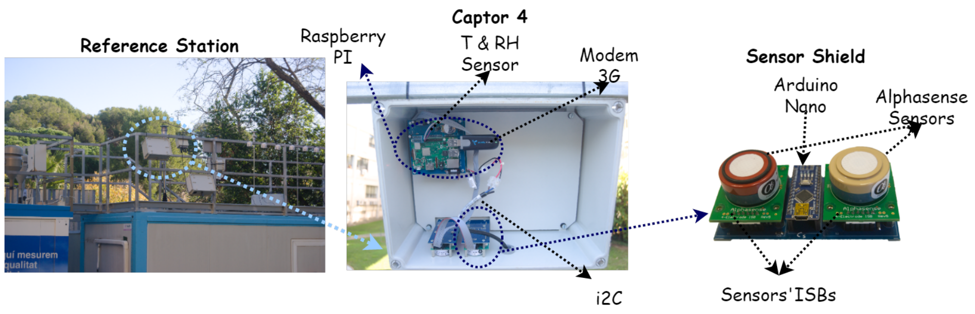


**Figure 2.3:** On the left, a *Captor 4* collocated at Palau Reial reference station. On the middle, a *Captor 4* image with the different components annotated and its deployment site. On the right, the sensing shield that the *Captor 4* mounts with its different components.

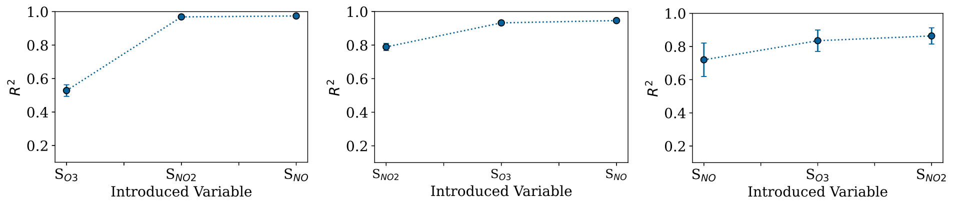
*20002* mounted one Alphasense OX-B431 $O_3$ sensor, one Alphasense NO2-B43F $NO_2$ sensor, and a DHT-22 temperature and relative humidity sensor. Given the availability of high-frequency measurements (0.5 Hz), different sensor sampling policies can be simulated by subsampling these data sets[4].

**Table 2.3:** Description of the data sets used for the experiments. $T_s$ is the sensor sampling period.

| Node Label | Sensor | Deployment Period | $T_s$ |
|---|---|---|---|
| *20001* | $O_3$ | 2021/01/15 - 2021/05/15 | 2 s |
| | $NO_2$ | 2021/01/15 - 2021/05/15 | 2 s |
| | NO | 2021/01/15 - 2021/05/15 | 2 s |
| *20002* | $O_3$ | 2021/01/15 - 2021/05/15 | 2 s |
| | $NO_2$ | 2021/01/15 - 2021/05/15 | 2 s |

Each of the low-cost electrochemical sensors provides measurements for the working electrode and auxiliary electrode in analog-to-digital converter units. The temperature sensor collects measurements in degrees Celsius ($^\circ C$), and the relative humidity sensor collects measurements in percent humidity (%). Table 2.3 summarizes the different sensor data used in the experiments. The two nodes were placed at a reference station for 4 months, from 2021/01/15 to 2021/05/15. The average concentrations measured by the reference station at Palau Reial (Barcelona) from 2021/01/15 to 2021/05/15 are 57.46, 19.87, and 4.28 $\mu$gr/m$^3$ for $O_3$, $NO_2$ and NO, with standard deviations of 23.79, 15.31 and 11.74 $\mu$gr/m$^3$ respectively. The $NO_2$ and NO present important concentration peaks above 100 $\mu$gr/m$^3$. Reference station's values are available hourly, so the reference data period $T_{ref}$ is equal to one hour. The reference station's data can be downloaded from the government's open data web [47], while the raw Captor sensory data have been made public on Zenodo's website [48].

[4]We emphasize that the sensing boards were not put to sleep, and therefore, the effect that turning the sensor on and off may have on aging or sample quality has not been studied.



(a) Sensor 20001 $O_3$ forward feature selection CV $R^2$.

(b) Sensor 20001 $NO_2$ forward feature selection CV $R^2$.

(c) Sensor 20001 NO forward feature selection CV $R^2$.

**Figure 2.4:** Forward stepwise feature selection for the different sensors of node 20001 using the MLR as estimation method. The horizontal axis indicates the variables added to the model from left to right. Bands indicate average CV $R^2$ 95% confidence interval.

## 2.3.2. Defining the calibration models: Feature Selection

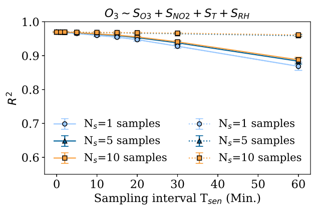
In order to analyze the quality of the calibration, it is first necessary to define the sensor array to be used in the calibration, since the node in question may have measurements of $O_3$, $NO_2$, NO, temperature, and relative humidity. The machine learning-based sensor calibration is studied in more detail in the following chapter 3. At the moment, we use the MLR as calibration method and we select the best subset of sensors for calibration using a forward stepwise feature selection procedure. The forward stepwise feature selection is a wrapper method (model dependent) that proceeds by adding the feature that improves the model's performance the most at a time until all features are included. This procedure is done to take into account the different correlations and cross-sensitivities present between LCSs and it is dependent of the predictive model. Figures 2.4.a), b), and c) show the results for the forward feature selection for the MLR and Captor 20001 sensors[5]. The results for the $O_3$, Figure 2.4.a), show that including the $NO_2$ sensor the $R^2$ increases about 0.44 (from $R^2$=0.53 to $R^2$=0.97), while the NO introduces no further improvement. This is reasonable since the electrochemical Alphasense $O_3$ sensor measures both $O_3$ and $NO_2$ so the $NO_2$ sensor needs to be introduced to compensate for its effect. Regarding the $NO_2$ calibration, Figure 2.4.b), the $NO_2$ achieves an $R^2$ of 0.79 by its own and the introduction of the $O_3$ sensor slightly improves the calibration by more than 0.14 $R^2$. Finally, the NO sensor benefits from introducing the $O_3$ sensor to the calibration, improving the CV $R^2$ from 0.72 to 0.84.

**Table 2.4:** Best subset of sensors found via forward stepwise feature selection for the MLR.

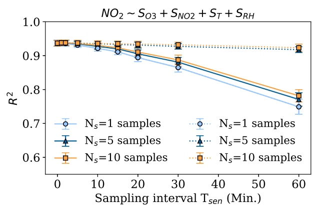
| Target Sensor | Best Subset |
|---|---|
| $O_3$ | $O_3$, $NO_2$, T, and RH |
| $NO_2$ | $NO_2$, $O_3$, T, and RH |
| NO | NO, $O_3$, T, and RH |

Table 2.4 shows the resulting arrays of sensors used for the calibration of each one of

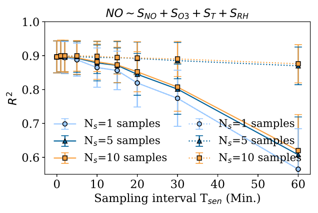
[5]Temperature and relative humidity sensors are always included in sensor calibration since these are important correctors for environmental conditions [18].



(a) Captor 20001 $O_3$ CV $R^2$ for different $T_{sen}$ and $N_s$.



(b) Captor 20001 $NO_2$ CV $R^2$ for different $T_{sen}$ and $N_s$.



(c) Captor 20001 NO CV $R^2$ for different $T_{sen}$ and $N_s$.

**Figure 2.5:** Average CV $R^2$ and 95% confidence intervals for different sampling settings using MLR. Solid lines denote the strategy that the $N_s$ samples are taken consecutively after a sensor response period (strategy *A*), and the dotted lines denote the strategy that the $N_s$ samples are taken uniformly at $T_{sen}$ (strategy *B*).

the sensors. For instance, for the $O_3$ sensor calibration, it is necessary to use the array of sensors $\{O_3, NO_2, T, RH\}$. We take these sensor arrays as the baseline for all calibration models (linear and nonlinear), even though, feature selection could be done for each one of the models and sensors.

### 2.3.3. SENSOR SAMPLING IMPACT: $T_r \approx 0$

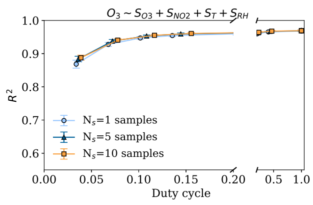
In this section, we explore the ideal case where the sensor has negligible response time $T_r \approx 0$, allowing to compare strategies *A* and *B*. The experiment performed compares the calibration quality of the different sensors assuming different sampling periods $T_{sen}$, from 2 s and 1 min to 60 min, and different number of samples measured per period $N_s = \{1, 5, 10\}$. Sampling strategy *A* (uniform sampling) is only possible when the sensor's response time is small or negligible as in this case.

Figure 2.5.a) shows the CV $R^2$s and their 95% confidence intervals for different sampling strategies for the $O_3$ calibration. Solid lines denote sampling strategy *B* while dashed lines denote sampling strategy *A*. As for the uniform sampling strategy (*A*), it is observed that for $N_s > 1$ any period is sufficient to keep the quality of the calibration constant, only in the case $N_s = 1$ (where strategies *A* and *B* coincide) a deterioration of the calibration is observed, with the $R^2$ is reduced by 0.1. This trend for strategy *A* can also be seen in the calibration of $NO_2$ and NO, 2.5.b) and c), where for $N_s = \{5, 10\}$ the calibration quality is maintained for different $T_{sen}$, although in these cases for $N_s = 5$ we see a slight reduction of $R^2$ as $T_{sen}$ increases. Regarding consecutive sampling *B*, it can be seen that for $T_{sen} \leq 10$min there is little worse in terms of $R^2$. From this point, the $R^2$ starts to decrease until it reaches a CV $R^2$ of 0.87. Sampling strategies $T_{sen} = \{30\text{min}, 60\text{min}\}$ and $N_s = 1$ implies that the sensor is sampled one or twice, so the data aggregation and filtering stages may be affected and the resulting measurement biased. The difference between taking one, five, or ten samples is not significant until we sample every 30 to 60 min. When we take few samples, $N_s = 1$, the confidence intervals are larger since the calibration quality may exhibit larger variability due to the low number of samples. As an example, for $T_{sen} = 5$ min and $N_s = 1$, the aggregation to obtain the measurement every $T_{ref}$ is obtained averaging 12 samples, so in this case and for lower sampling periods one sample is enough to produce a good aggregation estimate. Nevertheless, when the sampling period is larger $T_{sen} \geq 5$ min the data quality may not be maintained even when taking more than one measurement per sampling period. For instance, if $T_{sen} = 30$ min and $N_s = 10$, there are 20 samples participating in the aggregation at instant $T_{ref}$ (more than the 12 samples with $T_{sen} = 5$ min and $N_s = 1$) but they are less representative. In other words, it is better to sample fewer measurements more distributed over the period $T_{ref}$, than to sample more measurements consecutively but fewer times at $T_{ref}$.

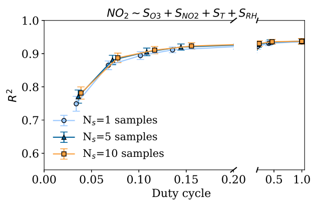
Figure 2.5.b) shows the results for the $NO_2$ calibration. In this case, for sampling strategy *B*, it is observed how the $T_{sen}$ has a larger impact on the $R^2$. Actually, the $R^2$ is maintained for $T_{sen} \leq 10$ min, with $R^2$ around 0.94. However, the $NO_2$ calibration is observed to worsen more for large sampling periods than the $O_3$ calibration since at $T_{sen} = 30$ min the $R^2$ is reduced to 0.86, and at $T_{sen} = 60$ min to 0.75. Moreover, in this case, the number of samples taken every sampling period also has a larger impact on the $R^2$, where for $T_{sen} = 60$ min taking one or five samples may result in a more than 0.03 $R^2$ difference. In addition, since $NO_2$ is a less smooth signal than $O_3$, with few samples per $T_{sen}$ interval, there is greater variability, which explains the higher confidence interval values for $N_s = 1$.

Finally, for the NO calibration, Figure 2.5.c), it is observed an even larger impact of the sampling strategy on the goodness-of-fit of the calibration. The NO is a phenomenon that naturally presents more abrupt changes in the measurements which causes the different calibrations to have large confidence intervals. A similar decreasing trend as for $O_3$ and $NO_2$ $R^2$ is observed, but in this case, the calibration quality starts to worsen for $T_{sen} \geq 5$ min the $R^2$ meaning that the NO phenomena may present larger frequencies so a more intensive sampling scheme is needed. Moreover, the same happens with the number of samples, the gap in performance between taking one sample and five is the largest of all three pollutants. The NO signal contains many peaks so even sampling $N_s$ samples in a row for an instant that does not pick up such peaks may be unrepresentative. Regarding the confidence intervals, both sampling schemes suffer from large confidence intervals due to the variability of the phenomenon. Thus, even when taking samples uniformly, pollution peaks or important events may not be captured. Therefore, in the case of signals with high variability, and high bandwidth, it is logical to sample at more points.

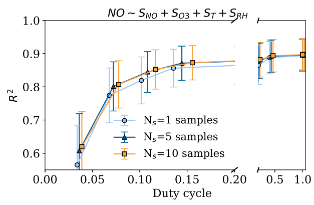
To sum up, we can conclude that it is better to take more than one sample $N_s > 1$ if the sampling period is large ($T_{sen} \geq 10$ min), so that the aggregation is more representative and the data filtering more effective. However, in the case of having a lower sampling period, fewer samples are enough to obtain a high $R^2$ since the aggregation at each $T_{ref}$ will be more representative and the data filtering more effective. Moreover, as observed with the different decreasing trends of the $R^2$ with the $T_{sen}$, different sensors may require different sampling schemes to achieve a certain data quality if a duty cycle system is implemented. For instance, the performance gap between sampling schemes *A* and *B* for $T_{sen} = 30$ min and $N_s = 5$ is of 0.02 $R^2$ in the $O_3$ case, 0.05 $R^2$ in the case of the $NO_2$, and 0.08 $R^2$ in the NO case.



(a) Captor 20001 $O_3$ CV $R^2$ for different duty cycles.



(b) Captor 20001 $NO_2$ CV $R^2$ for different duty cycles.



(c) Captor 20001 NO CV $R^2$ for different duty cycles.

**Figure 2.6:** Average CV $R^2$ and 95% confidence intervals for strategy B and different duty cycles with a sensor response time equal to 2 min ($T_r$=2 min) using MLR.

## 2.3.4. SENSOR SAMPLING IMPACT: $T_r$≈2min

The previous case where the sensor response time $T_r$ was negligible is not common in air pollution LCSs. Therefore, in this section, we take into account a response time of about 2 minutes $T_r$=2 min since the response time of some sensors can reach up to 80 seconds, see section 2.3. In the case of wanting to save energy by implementing a duty cycle scheme, the sampling strategy *A* would not be longer feasible since a response time of 2 minutes does not allow uniform sampling in $T_{sen}$ periods and would imply having the sensing system always powered on. Thus, in an energy-constrained sensing system, strategy *B* is the most feasible and the one studied in this section.

The results of the duty cycle for a 2 min sensor response time ($T_r$=2 min) and therefore with a $T_{sen}$>2 min are shown in Figure 2.6. Recall that low duty cycles correspond to large $T_{sen}$ and fewer samples taken in the interval $T_{ref}$ resulting in a shorter time that the node is measuring. The effect of a low duty cycle is observed in Figures 2.6.a), b) and c), where low $R^2$ are obtained for low duty cycles. In all three cases, $O_3$, $NO_2$ and NO, the quality of the calibration stabilizes from duty cycle equal to DC=0.20 ($T_{sen}$= 10 min). Therefore, a sampling period of about five or ten minutes guarantees the representativeness of the sampled data. When the physical phenomenon presents large variability, as in the case of NO, the confidence intervals are larger. However, large $T_{sen}$ periods with one single sample introduce more variability, as observed in the confidence intervals of sampling strategies with $N_s$= 1. In this case, it is better to take more samples, slightly increasing the duty cycle, since the sensor response time is the one that dominates the duty cycle. It is also observed, as seen previously, that the CV $R^2$ for the $O_3$ sensor stabilizes before that of $NO_2$, and that of $NO_2$ before that of NO, showing how different sensors may require different duty cycles in order to maintain good data quality.

It is important to mention how *in-situ* calibration using a sensor array forces the sensors included in the sensor array to have aggregated data with the same time granularity. Therefore, the sampling of the sensor that requires the highest sampling frequency sets a lower bond for the rest of the sensors used in the sensor array.

### 2.3.5. POWER CONSUMPTION: $T_r \approx 2$min

The difference in power consumption between the two sampling strategies is proportional to the difference in their corresponding duty cycles. Thus, the different duty cycles are used to compare the power consumption of the different configurations for sampling strategy *B*. The duty cycle corresponds to the ratio between the time the subsystem is turned on to measure samples ($T_{on}$) and the total sampling period ($T_{sen}$). Moreover, since the consumption of the different components of the sensing system may be different, one can decide to send only the microcontroller to sleep mode or send the microcontroller to sleep and switch off the sensors[6].

**Table 2.5:** Average CV $R^2$ for sampling strategy *B* ($T_r = 2$ min) obtained with the multiple linear regression, k-nearest neighbors, and support vector regression models for the different sensors and duty cycles.

| **Sensor** | **DC= 1.00** | | | **DC= 0.10** | | | **DC= 0.03** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **MLR** | **KNN** | **SVR** | **MLR** | **KNN** | **SVR** | **MLR** | **KNN** | **SVR** |
| *20001* $O_3$ | 0.97 | 0.96 | 0.98 | 0.95 | 0.94 | 0.96 | 0.87 | 0.86 | 0.88 |
| *20001* $NO_2$ | 0.94 | 0.94 | 0.96 | 0.89 | 0.90 | 0.92 | 0.75 | 0.77 | 0.78 |
| *20001* NO | 0.90 | 0.95 | 0.97 | 0.82 | 0.89 | 0.90 | 0.56 | 0.66 | 0.60 |
| *20002* $O_3$ | 0.97 | 0.96 | 0.98 | 0.94 | 0.93 | 0.95 | 0.84 | 0.84 | 0.85 |
| *20002* $NO_2$ | 0.92 | 0.92 | 0.94 | 0.88 | 0.89 | 0.91 | 0.74 | 0.76 | 0.78 |

The results for the duty cycles, Figures 2.6.a), b) and c), showed that the $O_3$ and $NO_2$ sensors were able to achieve a good calibration accuracy with a duty DC≈0.10, introducing very little improvement at higher duty cycles rates, whilst, in the case of the NO sensor, a slightly higher duty cycle was required (DC= 0.15). Thus, the duty cycle can be reduced about seven or ten times (DC= 0.15 and DC= 0.10) while maintaining a calibration quality as good as in the continuous powered case (DC= 1) so reducing the power consumption considerably and allowing the use of battery-powered systems.

Table 2.5 compares the average CV $R^2$ obtained for the different sensors, the three different ML algorithms, and different duty cycles (with $T_r \approx 2$min); duty cycles equal to 1.0 ({$T_{sen} = 2$ s, $N_s = 1$}), 0.10 ({$T_{sen} = 20$ min, $N_s = 1$}), and 0.03 ({$T_{sen} = 60$ min, $N_s = 1$}). For the studied case so far (MLR calibration), for duty cycles of 0.10 the sensor calibration worsens by about 0.02-0.08 $R^2$, in the worst case, the NO sensor drops from 0.90 to 0.82 $R^2$. The NO sensor is seen to need a higher duty cycle, about 0.15, this confirms what was observed above with the sampling period of the NO, so that its data quality is not reduced so much. On the other hand, in the extreme case where the node is only powered on once, with a resulting duty cycle of 0.03, the $R^2$ worsens approximately by 0.10 $R^2$ in the case of the 20001 $O_3$ sensor, and in the case of the 20001 NO sensor by 0.34 $R^2$. Thus, the power consumption can be reduced up to ten times (DC=0.10) by slightly reducing the quality of sensor calibration.

Regarding the nonlinear sensor calibration models, Table 2.5, it can be seen that these nonlinear models do not improve the MLR performance very much for the $O_3$ and $NO_2$ sensors since the sensors' responses are very linear, except for the NO sensor where these nonlinear models are able to improve the calibration around 0.07 $R^2$. Nevertheless, in terms of calibration quality with respect to the duty cycle, the nonlinear models show

[6]This option has not been evaluated since depending on the sensing technology this switch-off may incur in the need to wait a longer time for the sensor to measure correctly.

the same decreasing trend of the $R^2$ with the duty cycle since the impact of the duty cycle is on the representativeness of the data and does not depend on the ML model used.

To sum up, the experiments show how for the gas sensors analyzed and different sensor response times the duty cycles required to obtain good data quality may vary. Moreover, lower sensor response times may result in more efficient duty cycles, so the hardware components limit the power saving achieved in the sensing system. In these precise experiments, assuming response times in the order of two minutes, duty cycles of 0.1 can be achieved, calibrating with hourly reference values, and maintaining the quality of the calibrated data. Otherwise, duty cycles between 0.01 and 0.02 can be achieved if sensor response times are negligible.

## 2.4. CONCLUDING REMARKS & FUTURE WORK

IN this chapter, we have reviewed the pre-processing steps of LCSs; from sensor sampling to air pollution estimation. We have reviewed the stages of sensor sampling, filtering to remove outliers, data aggregation (for synchronization with reference instrumentation or obtaining data at the desired granularity), and the production of air pollution estimates by machine learning. In this way we have answered the first research item raised in the introduction chapter 1:

**(R.Q.1.1):** *Which pre-processing steps are required prior to the calibration of a low-cost sensor? What effect does sensor sampling have on the calibration quality and energy consumption of the sensing node?*

Within the pre-processing stages, we have investigated in detail the effect of the sensor sampling strategy on calibration quality and duty cycle (and consequently power consumption). The sampling strategy has an important effect on the calibration quality since it determines the amount of collected samples to be used for filtering and subsequent aggregation, thus affecting the representativeness of the aggregated data produced by the sensor. The results of the experiments carried out show how in the common case of having a sensor response time a duty cycle can be applied by sampling samples consecutively in each measurement period. In this way, the calibration quality can be maintained ($\pm 0.03$ $R^2$) using a duty cycle of 0.15, thus reducing the energy consumption about seven times compared to having the node always plugged in. Moreover, depending on the bandwidth of the measured phenomenon different sampling frequencies may be required, resulting in specific sampling strategies for sensor phenomena. Finally, the different calibration models have suffered the same loss of quality due to the sampling strategy, which means that sensor sampling affects the representativeness of the samples and that regardless of the calibration model used there is a loss of quality in the calibration.

In conclusion, in the practical case of a node's design where different sensing technologies, with different sensor response times to the ones shown in this work, are used, it would be necessary to characterize the most appropriate duty cycle based on the designer's needs, the sensors used, and how the sensors are calibrated.

More concisely we can summarize the conclusions as:

**Duty-Cycle: Sampling strategy impact**

- The sensor sampling strategy directly affects the quality of the sensor calibration and the duty cycle.
- Using a flexible sampling strategy, duty cycles can be reduced up to seven times while maintaining the quality of the sensor calibration.
- Sensors measuring different phenomena may require different sampling frequencies.
- The sensor calibration model used does not affect the loss of data quality due to sampling.
- Different sensor technologies and hardware designs may implement more efficient duty cycles.
- Prior to a sensor deployment, a fast-sampling experiment such as the one shown in the chapter can be used to design the duty cycle necessary to meet the needs of the particular deployment sensor.

In future research, it may be interesting to use signal reconstruction or compressed sensing techniques to elaborate more efficient and tailored duty-cycle strategies. In this way, a duty cycle could be designed depending on the correlations present in the measured data. Finally, it is not only interesting to study the duty cycle at the node level but also at the sensor network level where, depending on the measured phenomenon, strategies can be developed to reduce the overall consumption of a sensor network.

**Practical Tip !**

In the case of wanting to deploy a node with a duty-cycle system, a previous study can be done, while the node is collocated in a reference station, to determine the exact duty cycle according to the needs and required data quality.

☐ *From now on, we assume that the sensor data has been pre-processed and that we only have available the final aggregated measurement and, consequently, we focus on the sensor calibration stage.*

# 3

# Low-Cost Sensor Calibration For Air Pollution Monitoring

*An expert is a person who has made all the mistakes that can be made in a very narrow field.*

Niels Bohr

Low-cost sensors (LCSs) are known for their low accuracy compared to high-precision instrumentation. In addition, they are known to suffer from various problems that cast doubt on whether their data quality is good enough for regulated air quality monitoring applications [4, 49]. Lately, a trend to improve their accuracy consists in the application of supervised machine learning (ML) techniques to calibrate the LCSs while placed next to high-precision instrumentation that provides ground-truth values of pollution [15, 27].

The objective of this chapter is to show the effectiveness of different supervised ML methods for the calibration of LCSs. The use of linear models such as multiple linear regression (MLR) and nonlinear models such as support vector regression (SVR) are studied. These models are applied to real data sets of an IoT node, the Captor node, which contains low-cost tropospheric ozone sensors [9]. In addition, their long-term accuracy as environmental conditions change is studied.

The chapter is organized as follows; section 3.1 shows describes the state-of-the-art methodologies used for sensor calibration and section 3.2 describes the state-of-the-art machine learning models for sensor calibration. Then, section 3.3 describes the different machine learning techniques evaluated for calibration, and section 3.4 shows experimental results. Finally, section 3.5 concludes the chapter. This chapter presents the findings made in the article "*A comparative study of calibration methods for low-cost ozone sensors in IoT platforms*", IEEE IoT-J, [50].

## 3.1. Calibration of air pollution low-cost sensors

Air quality monitoring has become a key task in order to take action and mitigate the possible effects of pollution. Recent advances in both low-cost sensing technologies and the internet of things allow the possibility of using nodes spread over an area to capture pollution measurements [51]. But, one of the most important points is the accuracy of these sensors [4, 49]. Several studies argue that the accuracy of LCSs, as well as their variability, do not meet the accuracy levels required by agencies to be used in decision-making processes [7]. Therefore, one of the most important challenges that exist with this low-cost sensing technology is the data quality.

Although the accuracy of these sensors is one of the most important challenges, it has also been highlighted the fact that LCSs can be the key piece in the measurement of pollution in cities, helping to increase the resolution provided by reference stations[1]. Thus, given the limited number of deployed reference stations, LCSs can be the tool that complements the measurements of regulatory stations according to Snyder *et al.* [49]. For instance, Rajasegarar *et al.* [52] propose the joint use of LCSs for measuring PM and reference stations, showing the improvement in the estimation given a large density of LCSs. Hence, even though the LCSs may not be enough accurate to provide reliable atmospheric estimates, they can provide good coarse estimations and their data can be used for several applications such as awareness purposes [5, 53]. One example of application is the H2020 Captor project, where sensing nodes were placed in volunteers' homes to capture $O_3$ measures [9]. Despite of the accuracy of these measures, the monitoring campaign could provide measures to raise citizen awareness about pollution levels in three different areas (Spain, Italy, and Austria).

The methodology that has gained importance during the last years to increase the reliability and accuracy of LCSs is the *in-situ* sensor calibration. Penza *et al.* [54] show the use of calibrated LCSs deployed in conjunction with the official monitoring network to calculate air quality indexes (AQI). Similarly, Spinelle *et al.* [55] evaluate the performance of different calibrated sensors to check whether they meet the uncertainty required by governments. There are several issues that hinder the deployment of LCSs, including the need to recalibrate the sensors as the environmental conditions of the deployment location change, as well as taking into account sensor aging or manufacturing variability [7]. In order to improve the quality of the data reported by LCSs, supervised ML techniques are used to calibrate these sensors [14, 42, 56]. Zimmerman *et al.* [15] calibrate $NO_2$ LCSs using the random forest (RF) algorithm, achieving an important improvement with respect to linear calibration models.

To mitigate sensor data quality issues, different calibration methodologies have been studied [10, 11]. These strategies depend on what resources are available (e.g., the availability of reference instrumentation) and the type of network to be deployed. More precisely, Table 3.1 shows a brief summary of the different sensor calibration approaches used in the literature. As it can be seen, there are many different methodologies for sensor calibration, and the exact configuration always depends on the characteristics of the sensor network to be used and the resources available in the area of interest. For example, in the case of a mobile sensor network [57], where the sensors are mounted on public transport, apart from the *pre/post* calibration, an *opportunistic* calibration can also be performed when a sensor is sufficiently close to a reference station during a certain time window.

Apart from *pre/post* calibration, other studies make use of recalibration techniques to

[1]Throughout this thesis the terms *reference station*, *reference instrumentation*, and *high-precision instrumentation* are used interchangeably to denote the instrumentation that provides air pollution ground-truth concentrations.

**Table 3.1:** Calibration approaches summarized from [10].

| | | **Description** |
|---|---|---|
| **Area of interest** | *Micro* | The sensor is optimized for a specific location. |
| | *Macro* | The sensor is optimized for an area of interest. |
| **Number of sensors** | *Single sensor* | Only one sensor involved in the calibration. |
| | *Several sensors* | More than one sensor involved in the calibration. |
| **Available ground-truth data** | *Non-blind* | Ground-truth values available. |
| | *Semi-blind* | Indirect access to ground-truth values. |
| | *Blind* | Ground-truth values not available. |
| **Position reference station** | *Collocated/In-situ* | Sensor placed close to a reference station for calibration. |
| | *Multi-hop* | Sensor calibrated using nodes already calibrated. |
| | *Model-based* | Sensor calibrated using reference model at a location. |
| **Calibration frequency** | *Pre/Post* | Sensor calibrated before and after deployment. |
| | *Periodic* | Sensor is periodically calibrated. |
| | *Opportunistic* | Sensor is calibrated when ground-truth values are available. |
| **Mode of operation** | *Offline* | Sensor is calibrated when it is not operating. |
| | *Online* | Sensor is calibrated while in operating mode. |
| **Mode of calibration processing** | *Centralized* | Calibration model optimized in a central server. |
| | *Distributed* | Calibration model optimized using sensor node collaboration. |

deal with long-term quality problems [58, 59]. For example, Wei *et al.* [59] study the effect of drift and changing environmental conditions on sensors. Saukh *et al.* [58] mounted nodes on buses to opportunistically recalibrate sensors when a bus passed near a reference station, thus mitigating long-term calibration problems. Among all possible calibration settings, in this thesis we focus on the following one:

$$\textit{Micro} \rightarrow \textit{Single/Several sensors} \rightarrow \textit{Non-blind} \rightarrow \textit{In-situ} \rightarrow \text{Pre/Post} \rightarrow \textit{Offline} \rightarrow \textit{Centralized}$$

This corresponds to the most common approach where a sensor is used to sense the pollution concentrations at a specific location (*micro*), one or more sensors are involved in the calibration (*single/several sensors*), the sensors are collocated at a reference station to be calibrated using an ML algorithm (*in-situ* and *non-blind*), the calibration model optimization is performed at a centralized server before the sensor deployment (*offline* and *centralized*). Finally, the model is transferred to the sensing node to be deployed or the sensors' measures are sent to the central server to apply the previously trained models (*pre/post*, *offline*, and *centralized*).

The most recent work has addressed the calibration of LCSs using ML techniques. Hence, these powerful modeling techniques can help to improve the quality of data provided by the sensors. Therefore, the rest of this chapter is focused on the study of different ML techniques, both linear and nonlinear, to investigate how much they can improve sensor calibration.

## 3.2. Machine learning-based in-situ calibration

Recent studies regarding the LCS *in-situ* calibration are based on the application of supervised ML techniques [16, 60]. Given the increasing interest in ML and its proven ability to learn highly complex nonlinear functions, these mathematical models are an ideal candidate for obtaining calibration models. For instance, Spinelle *et al.* [27] use from linear techniques such as multiple linear regression to nonlinear techniques such as artificial neural networks to calibrate $NO_2$ and $O_3$ sensors. Similarly, other techniques such as random forest have been used on LCSs [15]. Bigi *et al.* [16] compare the performance of three ML models (multiple linear regression, support vector regression, and random forest) applied to NO and $NO_2$ LCSs. The results showed a better performance of the nonlinear techniques as well as the benefit of introducing different correlated sensors in the calibration, where $NO_2$ and NO are introduced to benefit from each other measurements. Other research, use ML techniques to derive a generalist sensor calibration model, but more importantly, conclude that each sensor needs to be calibrated individually given the intrinsic variability that exists from sensor to sensor from the same manufacturer [56]. In general, the different studies show the existence of cross-sensitivities and correlations between sensors, so that, for the prediction of a pollutant there are sensors of other phenomena that can influence the calibration. Therefore, one of the most common configurations in calibration is the use of several sensors, to take into account other influential factors in the target sensor response.

**Definition 6** ***Cross-sensitivity*** *is an effect that occurs when a sensor has interference in its readings caused by the presence of other pollutants different than its target gas. As an example, an $O_3$ sensor may react to other gaseous species, such as $NO_2$, so its readings may be influenced by another pollutant.*

Several studies mention the importance of using an array of sensors to calibrate sensors such as; $O_3$, $NO_2$, CO, or $CH_4$ [58, 60]. Another example of sensor calibration using an array of sensors shows the inclusion of correlated pollutants in the calibration [14]. Thus, taking benefit from cross-correlations and cross-sensitivities present between different gas pollutants. Wei *et al.* [59] show the dependence of temperature and relative humidity on the evolution of low-cost electrochemical sensors. Moreover, in order to calibrate an electrochemical $NO_2$ sensor, an $O_3$ sensor, temperature and relative humidity are also required. However, the performance of different ML techniques is still to be tested, as well as different technical aspects such as the number of samples required to train the model, or the performance in the long term. The required number of samples per model is important since it imposes a calibration period and most of the applications rely on a pre/post sensor calibration. These aspects are of great interest in this field due to the specific calibration methodology used for low-cost sensing technologies.

**Definition 7** ***Sensor array:*** *in the context of sensor calibration, a sensor array is defined as the use of several P sensors in the calibration of a sensor or in the estimation of a pollutant. In this way, a sensor array is introduced into a sensor calibration mechanism to take advantage of the cross-sensitivities and cross-correlations present, improving the overall estimation.*

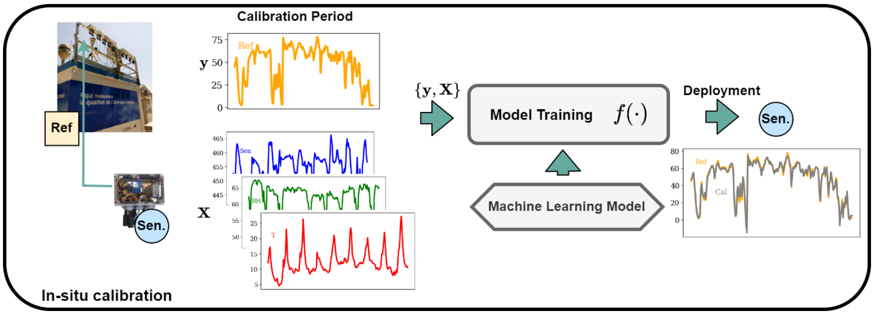


**Figure 3.1:** Machine learning-based *in-situ* LCS calibration. LCSs are placed next to reference instrumentation for a calibration period to train the ML-based calibration model.

Recalling from chapter 1, the *in-situ* calibration can be seen as a supervised learning task. Indeed, given a training set obtained during a sensor calibration period $\mathbf{X}\in\mathbb{R}^{N\times P}$, where $N$ is the number of collected samples and $P$ is the sensor array size[2], and the corresponding ground-truth values provided by a reference instrument $\mathbf{y}\in\mathbb{R}^{N}$, the goal is to find the function $f:\mathbb{R}^{P}\rightarrow\mathbb{R}$ that approximates the ground-truth concentrations given the sensors readings, i.e., $y_i\approx f(\mathbf{x}_i)$. Different ML techniques assume different characteristics for the regression function $f(\cdot)$. For instance, the multiple linear regression assumes that the regression function is linear in its covariates $\mathbf{x}_i$. Another example is random forests, which assume that the function is the average output of an ensemble of decision trees, which have been decorrelated.

Figure 3.1 shows the overall process for the *in-situ* LCS calibration. First, the sensor is collocated near a reference station for a calibration period to obtain the set of tuples $\{(\mathbf{x}_1, y_1),\ldots,(\mathbf{x}_N, y_N)\}$ so that the regression function $f(\cdot)$ can be learned using a ML model. Once the pre-deployment calibration is done, the trained ML model is saved and the sensor is deployed at a specific location of interest to capture air pollution concentration levels during a monitoring campaign. The object of study of this part of the thesis are the ML models to be used during the sensor calibration and air pollution estimation.

## 3.3. MACHINE LEARNING MODELS

THIS subsection briefly introduces the different ML techniques studied for sensor calibration. The multiple linear regression (MLR), the K-nearest neighbors (KNN), the random forest (RF), and the support vector regression (SVR). For a more detailed discussion of these methods refer to [61].

### 3.3.1. MULTIPLE LINEAR REGRESSION

The multiple linear regression is the extension of the classical linear regression for the case where more than one feature is used ($P\geq1$). $\mathbf{X}\in\mathbb{R}^{N\times P}$ corresponds to the set of sensor observations, where $N$ is the number of observations and $P$ is the number of features per observation. Usually, in the sensor calibration paradigm, more than one

[2]Throughout the thesis we use the terms sensor array, features, and predictors interchangeably to refer to the sensor measures introduced to the calibration model.

sensor measurement is involved in the calibration, for instance, the ozone can be calibrated using the ozone sensor, the temperature sensor, and the relative humidity sensor $\mathbf{x}_i=[x_{O_{3_i}}, x_{temp_i}, x_{rh_i}]$. Now, given the set of tuples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ the multiple linear regression assumes that the response variable $y$ can be explained using a linear combination of the features $\mathbf{x}$:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \;\; i = 1, \dots, N \tag{3.1}$$

$$y_i = \beta_0 + \sum_{j=1}^{P} \beta_j x_{ij} + \epsilon_i, \;\; i = 1, \dots, N \tag{3.2}$$

Where $\boldsymbol{\beta}$ are the model's coefficients to be learned and $\epsilon$ is the error term. These coefficients are obtained by least squares, minimizing the residual sum of squares (RSS). Now, using matrix notation, the RSS and the coefficients $\boldsymbol{\beta}$ can be obtained as follows:

$$\mathrm{RSS}(\boldsymbol{\beta}, \mathbf{y}, \mathbf{X}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \tag{3.3}$$

$$\frac{\partial \mathrm{RSS}}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^\mathsf{T}\mathbf{y} + 2\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\beta} \tag{3.4}$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y} \tag{3.5}$$

The resulting equation for the coefficients $\hat{\boldsymbol{\beta}}$ is known as the normal equation. Once the coefficients are learned from the data, the model can be used for future predictions. The linear model is well-known for its interpretability since if the data is standardized the learned coefficients can be interpreted as the feature importance.

### 3.3.2. K-nearest neighbors

The k-nearest neighbors is a method that belongs to a different family of methods, the dictionary-based. These models are characterized by the training data being the model itself. In this case, the model's output $\hat{y}_i$ is obtained by averaging the response values of the $K$ nearest points in the training. So, the model's prediction can be written as:

$$\hat{y}_i = \frac{1}{K} \sum_{j \in \mathcal{N}(i)} y_j, \;\; i = 1, \dots, N \tag{3.6}$$

Where $\mathcal{N}(i)$ denotes the neighborhood of data point i. As it may be noticed, a distance metric $d(x_i, x_j)$ is needed to define the notion of the closeness of two different data samples $\mathbf{x}_i$ and $\mathbf{x}_j$. The Minkowski distance is a distance metric, which has a governing parameter $\rho \in \mathbb{R}$, that generalizes other well-known distance metrics such as the euclidean distance and the Manhattan distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \left( \sum_{k=1}^{P} |x_{ik} - x_{jk}|^\rho \right)^{\frac{1}{\rho}} \tag{3.7}$$

Given the distance function, the model is learned by obtaining the hyperparameters minimizing the cross-validation mean squared error (MSE). Indeed, the model has two different hyperparameters that are not learned and need to be supplied to the model, the

$K \in \mathbb{N}^+$ and $\rho$. Cross-validation strategies are required to find the most suitable values for these parameters.

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.8}$$

### 3.3.3. RANDOM FOREST

The random forest is a ML method that belongs to the group of ensemble models. An ensemble method is a model formed by a combination of more than one individual model, whose outputs are aggregated to form a unique response. Indeed, a random forest is an ensemble of decision trees, which have been decorrelated, and whose output is the average of its decision trees:

$$\hat{y}_i = \frac{1}{N_{tree}} \sum_{t=1}^{N_{tree}} \hat{y}_{it}; \quad i = 1, \ldots, N \tag{3.9}$$

Where $N$ is the number of samples, $N_{tree}$ is the number of trees, and $\hat{y}_{it}$ is the output of $t$-th tree for the $i$-th sample. Let's give a brief introduction to decision trees before we delve into the random forests. A decision tree divides the output space into regions $\mathscr{R}$, so the algorithm needs to find the splitting points and splitting variables. Each node of a decision tree takes a feature and splits a region according to the feature value. Once a leaf is reached the output is the average output of the region $\hat{y}_{\mathscr{R}_i} = \frac{1}{|\mathscr{R}_i|} \sum_{x_j \in \mathscr{R}_i} y_{x_j}$. For all this, the depth of a tree is an important parameter, pruning techniques exist to avoid constructing extremely complex trees. The key relies on the selection of splitting points, which can be obtained using a greedy algorithm finding the best variable $x_j$ and the best splitting value $S$ at each point, and a pruning methodology to avoid overfitting.

The idea behind an ensemble model is to obtain noisy unbiased models whose average can improve the variance of the estimation. In order to increase the noise and thus improve the performance of the overall ensemble, two randomization steps are introduced to decorrelate the decision trees; *i)* each one of the decision trees is trained with a bootstrap sample of the training set, and *ii)* at each decision node of a tree a random subset of features are taken into account as candidate splitting variables.

All in all, the random forest model has three important hyperparameters; *i)* $N_{tree}$ the number of trees of the ensemble, *ii)* $D$ the maximum depth for the decision trees, and *iii)* $P_{tree}$ the number of random features to take into account at each decision node.

### 3.3.4. SUPPORT VECTOR REGRESSION

The support vector regression falls into the class of kernel methods. The aim of kernel models is to implicitly transform the data into a large high-dimensional feature space where data points are predicted or classified better. The core element is the kernel map $k : \mathbb{R}^P \times \mathbb{R}^P \to \mathbb{R}$ that given two data points $\mathbf{x}_i$, $\mathbf{x}_j$ computes a similarity metric between them. Then, the goal is to transform the data points to a larger dimensional space using a feature map $\phi : \mathbb{R}^P \to \mathbb{R}^Z$, where $Z > d$, and then compute the similarity between data points in that enlarged feature space. However, these operations in such a large space can be computationally intractable, this is where the *kernel trick* appears. The kernel trick avoids the explicit computation of the feature maps and states that inner products of mapped points can be computed implicitly using a kernel function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = < \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) > \tag{3.10}$$

Where $< \cdot, \cdot >$ denotes the inner product of two vectors and $k(\cdot, \cdot)$ is a valid kernel function. In conjunction with the kernel trick, the *representer theorem* states that there exists a function $\hat{y}$ that minimizes the empirical risk and belongs to a kernel Hilbert space $\mathscr{H}$ and can be expressed as a linear combination of the kernel evaluated at the different training data points, $\hat{y}_i = \sum_{j=0}^{N} \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)$. More precisely, the support vector regression is analogous to the support vector machine for regression where the points that are far from the regression plane are penalized. This is achieved by using the $\epsilon$-insensitive error function, where those points with residual smaller than the $\epsilon$ are not taken into account:

$$E_\epsilon(y, \hat{y}) = \begin{cases} 0 & , \text{if } |y - \hat{y}| < \epsilon \\ |y - \hat{y}| - \epsilon & , \text{else} \end{cases} \tag{3.11}$$

The output of the support vector regression can be defined as:

$$\hat{y}(\mathbf{x}) = \sum_{i=0}^{N} (\hat{\alpha}_i^* - \hat{\alpha}_i) k(\mathbf{x}, \mathbf{x}_i) + b \tag{3.12}$$

Where the parameters $\hat{\alpha}_i^*$ and $\hat{\alpha}_i$ are found by optimizing a convex quadratic optimization problem. Ultimately, several design choices are left; *i)* the value for the $\epsilon$, *ii)* the kernel map to use, for instance, the radial basis function (RBF) kernel is a well-known choice that produces an infinite dimensional feature space, *iii)* the possible parameters of the kernel function, and *iv)* a constant $C$ present in the objective function that penalizes the samples not well predicted.

## 3.3.5. Nonlinear models' hyperparameters

The performance of nonlinear models using a specific set of data is linked to their hyperparameters' values. These hyperparameters can not be set a priori, but a validation procedure is needed to select the hyperparameters' values that generalize to the test set, and therefore best fit the problem. There exist different techniques for model selection, such as train-validation-test split and cross-validation (CV). Throughout this chapter, the 5-fold cross-validation is used because it is one of the most widely used methods and it is applied in the training, thus taking advantage of the whole data set. The different hyperparameters mentioned throughout the section are listed in Table 3.2.

**Table 3.2:** Nonlinear models' hyperparameters.

| | **Hyperparameters** |
|---|---|
| **MLR** | None |
| **KNN** | $k$: number of neighbors, $p$: Minkowski distance power |
| **RF** | $N_{tree}$: number of trees, $D$: maximum depth, $P_{tree}$: number of features |
| **SVR** | $C$: penalization, $\gamma$: scale RBF kernel,$\epsilon$: width $\epsilon$-insensitive error |

Algorithm 1 depicts the ML-based *in-situ* LCS calibration using an array of sensors. For this purpose, the set of sensors' measures and reference values during the calibration period are needed, $\mathbf{X} \in \mathbb{R}^{N \times P}$, $\mathbf{y}_{tr} \in \mathbb{R}^N$, as well as the ML model $f(\cdot)$ and a range of values for its hyperparameters $\boldsymbol{\Omega}$. First, data is standardized, then the model's parameters are selected by means of cross-validation. Finally, the definitive model is trained using the best hyperparameters found during cross-validation and it is further used for air pollution estimation.

**Algorithm 1** Sensor array calibration.

```
Input: $\{\mathbf{X}_{tr}, \mathbf{y}_{tr}, f(\cdot), \boldsymbol{\Omega}\}$
 1: $\bar{\mathbf{X}}_{tr} \leftarrow$ Standardization($\mathbf{X}_{tr}$)
 2: $\bar{\mathbf{y}}_{tr} \leftarrow$ Standardization($\mathbf{y}_{tr}$)
 3: hyp $\leftarrow$ Cross_Validation($f(\cdot), \bar{\mathbf{X}}_{tr}, \bar{\mathbf{y}}_{tr}, \boldsymbol{\Omega}$)      ◁ Model Selection
 4: $\hat{f} \leftarrow$ Train_Model($f(\cdot), \bar{\mathbf{X}}_{tr}, \bar{\mathbf{y}}_{tr}$, hyp)            ◁ Model Training
 5: while $\mathbf{x}_{new}$ do                                          ◁ Prediction Phase
 6:     $\bar{\mathbf{x}}_{new} \leftarrow$ Standardization($\mathbf{x}_{new}$)
 7:     $\hat{\bar{y}}_{new} \leftarrow \hat{f}(\bar{\mathbf{x}}_{new})$
 8:     $\hat{y}_{new} \leftarrow$ Unstandardization($\hat{\bar{y}}_{new}$)
 9:     RETURN $\hat{y}_{new}$
10: end while
```

## 3.4. EXPERIMENTAL EVALUATION

THIS section evaluates the different ML models described in the previous section in the framework of *in-situ* LCS calibration. First, the different data sets used for the evaluation are detailed. And secondly, different experiments are performed to effectively compare the different ML models in the sensor calibration setting[3].

The different experiments elaborate on the performance of the different ML techniques applied to the data sets obtained in the H2020 Captor campaign. Three different experiments are performed:

(A) **Short-term performance**: data is randomly split into training set $\mathbf{X}_{tr}$ and test set $\mathbf{X}_{ts}$, 75% of the data and 25% of the data respectively. Then, a 5-fold CV procedure is performed using the training set in order to obtain the best-performing hyperparameters for each model. Finally, the performance of every model is compared using the test set. This procedure evaluates the performance of the sensor calibration short-term, i.e., in the posterior weeks of the *in-situ* calibration, where the environmental conditions are similar to those seen during the calibration. This is achieved by randomly sampling the training and test set.

75 %Train/ 25%Test Split → 5-fold CV over Train → Compare models over Test

(B) **Calibration period size impact**: a similar procedure to the short-term experiment is performed but after the train/test split, training subsets of increasing size are selected for sensor calibration. For instance, one week of data for training, two weeks for training, three weeks, etc.

X weeks Train/ 7 weeks Test → 5-fold CV over Train → Compare models over Test

[3]A python implementation of the mentioned ML methods for in-situ calibration is available at https://bitbucket.org/sans-rg/iot-calibration-software.

(C) **Long-term performance**: is a common evaluation of the sensor calibration. The sensor is calibrated using a few weeks of data, then the sensor is deployed at the desired location during the monitoring campaign. The goal is to observe the models' performance as the environmental conditions of the deployment location change with respect to the training conditions. Here the train/test split is not randomized.

4 weeks Train/Rest for Test → 5-fold CV over Train → Compare models over Test day by day

### 3.4.1. Data sets

The data used in this chapter for the study of the various ML techniques for calibration mentioned above consist of data captured by the H2020 Captor project nodes during the summer of 2017. The different IoT nodes were deployed in different testbeds in Spain, Austria, and Italy. In fact, the whole platform was made up of sixty IoT nodes. Two types of nodes were deployed; the Captor and the Raptor. The Captors are nodes containing an Arduino Yun as processing unit, together with a 3G modem to send the data to a centralized database, four ozone MOX sensors SGX Sensortech MICS 2614, a temperature sensor, and a relative humidity sensor, all connected to an external power supply. The different nodes provide measures every 30 minutes. In order to obtain a representative half-hour value a hundred samples are taken during this interval and the top and bottom ten percentiles are removed and the rest is averaged to obtain the final measurement. MOX sensors SGX Sensortech MICS 2614 have a load resistor and a variable resistor. The variable resistor changes with the ozone concentrations, and the sensor values $s_{raw}$ are obtained by measuring the load resistor voltage ($V_L$):

$$s_{raw} = R_L(1 - \frac{V_{cc}}{V_L}) \tag{3.13}$$

Where $R_L$ is the load resistor and $V_{cc}$ is the input voltage. The captor nodes assemble tuples of the form {Timestamp, $x_s, x_T, x_{RH}$} and send them via a 3G connection to an IoT repository using a REST web service.

On the other hand, the Raptor node was built at the Universite Clermont Auvergne (UCA) and is equipped with a Raspberry Pi as a processing unit, an Alphasense OX-B431 electrochemical ozone sensor, an Alphasense NO2-B43F electrochemical nitrogen dioxide sensor, a temperature sensor, and a relative humidity sensor. The sampling interval is also half an hour. The Raptors nodes mount both ozone and nitrogen dioxide sensors since the Alphasense OX-B431 sensors measure both phenomena simultaneously. Therefore, nitrogen dioxide measures are required to be subtracted from the ozone electrochemical sensor, then the raw ozone sensor output $s_{raw_{O_3}}$ for the electrochemical sensors is defined as:

$$s_{raw_{O_3}} = (s_{WE_{O_3}} - s_{AE_{O_3}}) - (s_{WE_{NO_2}} - s_{AE_{NO_2}}) \tag{3.14}$$

Where $s_{WE}$ corresponds to the working electrode (WE) reading and $s_{AE}$ corresponds to the auxiliary electrode (AE) reading. The deployment of the nodes for ozone monitoring was carried out using a pre/post calibration strategy where the nodes were placed at reference stations in locations close to where they were to be deployed for calibration. Then, mainly at the end of July and August 2017, the calibrated nodes were deployed at the volunteer's house to perform the ozone measurement campaign. Some of the nodes stayed longer at the reference stations to perform research tasks. In this study, the data from the nodes that remained collocated next to the reference instruments for a long period of time are used because the ground-truth ozone values are available and the models can be evaluated correctly. More specifically, Table 3.3 summarizes the data used in the study, separating the data from metal-oxide sensors from the nodes with electrochemical sensors.

**Table 3.3:** Summary of the different data sets used for the *in-situ* calibration evaluation.

| Node Name | Sensor Labels | Sensor Type | Calibration Place | Period | # Samples |
|---|---|---|---|---|---|
| Captor C17013 | s1,s2,s3,s4 | MICS 2614 | Manlleu (Spain) | 08/05/2017-04/10/2017 | 6745 |
| Captor C17016 | s1,s2,s3,s4 | MICS 2614 | Vic (Spain) | 26/05/2017-05/10/2017 | 6149 |
| Captor C17017 | s1,s2,s3,s4 | MICS 2614 | Tona (Spain) | 08/05/2017-05/10/2017 | 6944 |
| Raptor R69-17 | s1 | OX-B431 | MonteCucco (Italy) | 06/07/2017-11/10/2017 | 1797 |
| Raptor R308-17 | s1 | OX-B431 | Weiz Bahnhof (Austria) | 07/06/2017-27/09/2017 | 1439 |
| Raptor R69-18 | s1 | OX-B431 | MonteCucco (Italy) | 20/06/2018-26/09/2018 | 2295 |
| Raptor R202-18 | s1 | OX-B431 | Colli Euganei (Italy) | 18/06/2018-30/09/2018 | 2254 |
| Raptor R212-18 | s1 | OX-B431 | Osio Sotto (Italy) | 26/06/2018-25/09/2018 | 2148 |

All in all, the experiments are performed over twelve metal-oxide ozone sensors and five electrochemical ozone sensors. Captor nodes are named; C17013, C17016, and C17017. Raptor nodes are named; R69-17, R308-17, R69-18, R202-18 and R212-18. The captured data are comprised between the months of May and October 2017 and 2018. Further explanation of the data set of the whole H2020 Captor monitoring campaign can be found in [9].

### 3.4.2. SHORT-TERM PERFORMANCE

In this section, we show the calibration performance in the period close to the calibration. This is done by randomly splitting the data set into 75% for training and the 25% left for testing. In fact, the design matrix **X** is defined as:

$$\mathbf{X} = \begin{bmatrix} x_{O_{3_1}} & x_{T_1} & x_{RH_1} \\ x_{O_{3_2}} & x_{T_2} & x_{RH_2} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x_{O_{3_N}} & x_{T_N} & x_{RH_N} \end{bmatrix} \tag{3.15}$$

So, $\mathbf{X} \in \mathbb{R}^{N \times P}$, where $P = 3$ is the number of regressors, $x_{O_3}$ are the values of the ozone sensor, the $x_T$ are the temperature sensor values, and the $x_{RH}$ are the relative humidity sensor measures. This way, the samples of the training set become $\{(y_i, [x_{O_{3_i}}, x_{T_i}, x_{RH_i}])\}_{i=1}^{N}$, so to calibrate the real ozone concentrations $y_i$ the sensor measurements mentioned above are used. The relative humidity and temperature are correction factors commonly used in the literature [10, 11].

Four different error metrics are used to compare the performance of the different models. The *root-mean-squared error* (RMSE), the *centered-root-mean-squared error*

(CRMSE), the *mean bias* (MBias) and the *coefficient of determination* ($R^2$):

$$\text{RMSE}(\mathbf{y},\hat{\mathbf{y}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}$$

$$\text{CRMSE}(\mathbf{y},\hat{\mathbf{y}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[(y_i-\bar{y})-(\hat{y}_i-\bar{\hat{y}})]^2}$$

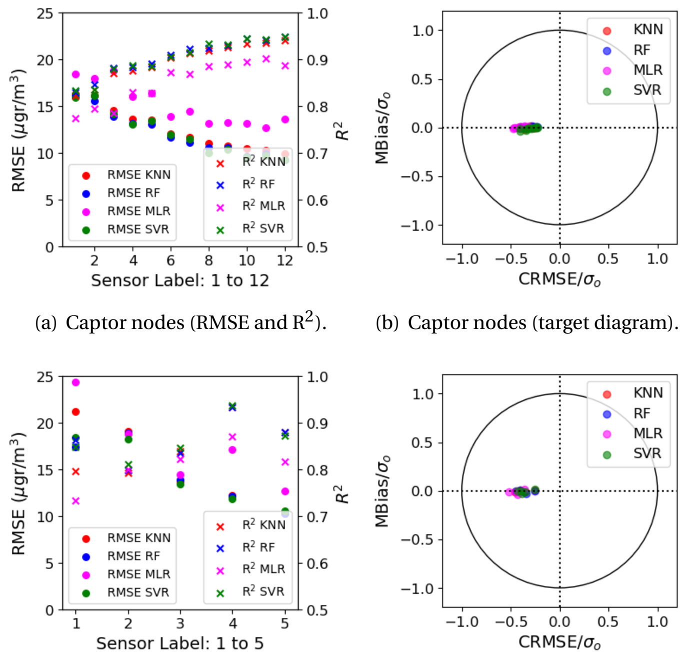
$$\text{MBias}(\mathbf{y},\hat{\mathbf{y}}) = \frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)$$

$$R^2(\mathbf{y},\hat{\mathbf{y}}) = 1-\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y})^2}$$

Where $\hat{y}_i$ is the prediction of a ML model for a sample $\mathbf{x}_i$. Figure 3.2.a) shows the test RMSE and $R^2$ obtained for the twelve MOX sensors present in the three Captor nodes that are the object of study. First, it can be observed that there is a large variability in performance between sensors, the RMSEs range between 9.50-19.00 $\mu$gr/m$^3$ and the $R^2$s range between 0.75 and 0.95. Thus, results show how LCSs are often affected by variability resulting from the manufacturing process. Indeed, lower values of RMSE correspond to larger $R^2$ values, meaning that the ML models are able to explain a large proportion of the variability of the ozone concentrations. Now, looking at the difference in performance between the linear (MLR) and nonlinear models, it is observed that the nonlinear models are able to reduce the RMSEs around 2.00-4.00 $\mu$gr/m$^3$ and the $R^2$ is increased by approximately by 0.05 with respect to the MLR. This shows the superiority of the nonlinear methods with respect to the linear model.

Figure 3.2.b) shows the target diagram for the same methods, this diagram has the normalized mean bias (MBias) as the y-axis and the normalized centered-root-mean-squared error (CRMSE) as the x-axis. This representation allows observing the decomposition of the error in terms of MBias and CRMSE. $\sigma_o$ is the standard deviation of the reference values $\mathbf{y}$, CRMSE<0 denote $\sigma_o > \sigma_{\hat{y}}$, i.e., reference values standard deviation larger than the standard deviation of the predicted values. The target diagram shows that the term dominating the error is the CRMSE since all four models have a normalized mean bias near to zero. This means that the error is mainly composed of the variance component. Therefore, the models exhibit almost no biases due to the representativeness of the training set, with representative data of different environmental conditions. Again, the MLR is the method that shows the largest CRMSE/$\sigma_o$ with near -0.50, while the nonlinear models are able to reach a normalized CRMSE of -0.20. Since the RMSE can be decomposed as $\text{RMSE}^2 = \text{CRMSE}^2 + \text{MBias}^2$[4], the CRMSE is the component that dominates the error. Now, let's take a look at the results of the five electrochemical sensors of the Raptor nodes in Figure 3.2.c). The same pattern is observed as with the MOX sensors, the test RMSEs range between 10.00-25.00 $\mu$gr/m$^3$ while the test $R^2$ range between 0.70-0.94. Apart from the same pattern in the error, the same variation from sensor to sensor is also observed. The MLR is the worst performing method and the nonlinear

[4]The decomposition $\text{RMSE}^2 = \text{CRMSE}^2 + \text{MBias}^2$ is related to the variance-bias trade-off present in the error. The ideal case corresponds to an unbiased estimator with small variance.



(a) Captor nodes (RMSE and R$^2$). (b) Captor nodes (target diagram).

(c) Raptor nodes (RMSE and R$^2$). (d) Raptor nodes (target diagram).

**Figure 3.2:** Test RMSE, R$^2$, and target diagram for Captor nodes C17013, C17016, and C17017 (twelve MOX sensors) and Raptor nodes R69-17, R308-17, R69-18, R202-18, and R212-18 (five EC sensors).

ones are able to improve the RMSE around 2.00-4.00 $\mu$gr/m$^3$, although in the specific case of the sensor with label 1, it can be observed how the SVR and the RF outperform the KNN. The target diagram for the raptor nodes, Figure 3.2.d), is similar to the case of the captor nodes, where the error is mainly composed by the normalized CRMSE and the normalized mean bias of the methods is almost zero.

All in all, we have observed; *i)* there is a large variability in performance from sensor to sensor, probably due to the manufacturing process, *ii)* the nonlinear methods are able to reduce the RMSE 2.00-4.00 $\mu$gr/m$^3$ with respect to the linear model, *iii)* there is no large difference in performance between the three nonlinear methods (KNN, RF, and SVR) compared in the experiment. However, the SVR obtains the best performance for a very small difference, and *iv)* the target diagrams show the contribution of the normalized CRMSE to the normalized RMSE, whereas the models show an almost negligible normalized mean bias. This means that having large data sets with representative data for different environmental conditions produces models with almost no bias.

### 3.4.3. Calibration period size

Sensor calibration corresponds to the use of ML techniques for a special case, where *in-situ* sensor data to carry out the calibration are scarce. Thus, for an air pollution monitoring campaign, sensors are first placed for a short calibration period at a reference station to be calibrated and then deployed. However, nonlinear models usually demand a higher amount of data than linear models. In order to compare the performance of different calibration models with respect to the calibration period size Figure 3.3 depicts the experiment performed. The test set is set to the last seven weeks of the data sets,

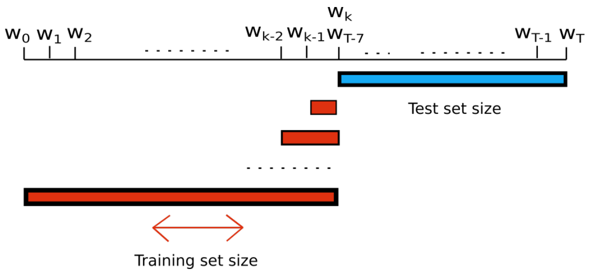


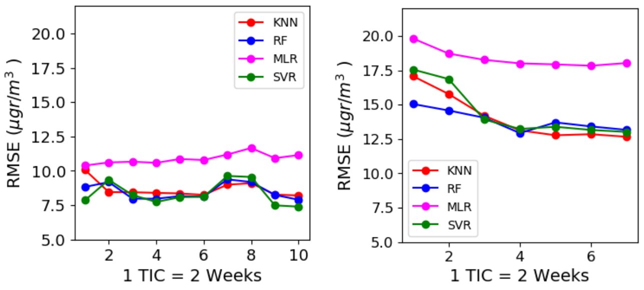
**Figure 3.3:** Setting for the impact of the calibration period size experiment. While the test set remains fixed the size of the training set is increased one week at a time.

and the training set is increased one week at a time from the start of the test set to the beginning of the data set. This way, the resulting models can be compared using the same data set, and the proximity of the train and test set do not influence the results, i.e., to keep the training set as representative as possible of the test set. For illustrative purposes, we show the results of the experiment for a metal-oxide sensor (C17016-s4) and an electrochemical sensor (R69-18) since the same pattern is observed for the other sensors.

Figure 3.4.a) shows the results for the metal-oxide sensor, where the difference in error between the MLR and the nonlinear methods is clear. The RMSEs obtained are quite steady, in fact, the error of the nonlinear methods with training sets of size of one and two weeks is quite unstable, but the error stabilizes with four to seven weeks of training. These variations are not observed with the MLR, as the difference in error between a training set of two weeks and a training set of three weeks is very small. The difference in the methods is clearer looking at the results for the Raptor node in Figure 3.4.b). The error of the nonlinear models seems to become stable with three or more weeks of calibration. The difference in test RMSE is smaller in the case of the MLR, where the difference in RMSE between two and three weeks of training is very small. Therefore, nonlinear models are shown to require more training data.

### 3.4.4. Long-term performance

So far we have seen how nonlinear models improve the calibration of LCSs in the short term, consequently improving their data quality, it is now time to look at how the sensors perform over longer deployment periods. The performance of long-term calibration models is one of the most important issues in air pollution monitoring with LCSs. The long-term performance of sensors can be affected by many factors [45, 62]. First of all,



(a) Captor node 17016 sensor 4. (b) Raptor node R69 2018.

**Figure 3.4:** Test RMSE for the impact of the calibration period size experiment, where the training set is increased one week at a time.

the fact that calibration data are scarce and may not represent the environmental conditions of the deployment site [18]. Secondly, LCSs are known to suffer from drift [45] and other problems such as aging that can affect the quality of the data captured. And lastly, since some pollutants such as ozone are seasonal, environmental conditions may change during the deployment period, examples include temperature changes, ozone concentrations being higher than those seen during training, etc. The ozone concentration values during the summer of 2017 in the places where the Captor nodes C17013, C17016, and C17017 were placed ranged from 0.00-200.00 $\mu$gr/m$^3$. Actually, concentrations during the sensor calibration period (training set) are lower than those seen over the months of July and August. The concentrations observed for the three locations can almost reach 200.00 $\mu$gr/m$^3$. In this way, LCSs are known to suffer from a problem called *concept drift*, where the calibration models become outdated as the environmental conditions to which the sensors are exposed change over time [12]. This is a common situation in sensor calibration, where the training data for the calibration are scarce and are data captured for a small time period before the deployment of the sensors.

**Definition 8** ***Concept drift*** *refers to the case where the data distribution changes dynamically over time. In ML and time series analysis, this problem is known as data set shift or non-stationarity, where the data distribution during the test set can differ from the distribution seen in the training set. This causes the learned sensor calibration models to become outdated.*

Figure 3.5, shows the ozone empirical distributions for the three different locations, as it is observed, the training distributions differ with respect to the testing ones.

In order to check the sensor calibration performance in the long term, we set the calibration period to the four first weeks of the data sets, and we evaluate the accuracy of the models' predictions day by day. This setting allows for checking how the error metrics evolve over time. This precise evaluation allows observing how their performance change as the estimation moves away from the calibration period. This may mean that the environmental conditions with respect to the calibration have changed, observe dif-

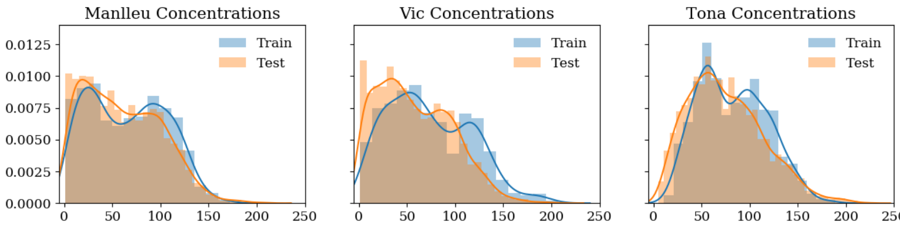


**Figure 3.5:** Ozone concentration empirical distributions for the three different locations. Training is set to the four first weeks and the testing is set to the remaining weeks (~16 weeks).

ferent concentrations, or even if the sensor is evaluated over a long period of time see drift or aging. More formally, we define the design matrix **X** for the training as:

$$\mathbf{X}_{w_i} = \left[\mathbf{x}_{O3_{w_i}}, \mathbf{x}_{RH_{w_i}}, \mathbf{x}_{T_{w_i}}\right]; \qquad \mathbf{X} = \begin{bmatrix}\mathbf{X}_{w_1}\\ \mathbf{X}_{w_2}\\ \mathbf{X}_{w_3}\\ \mathbf{X}_{w_4}\end{bmatrix} \in \mathbb{R}^{N_{tr}\times 3} \tag{3.16}$$

Where the sub-index $w_i$ indexes the data belonging to the $i$-th week of the data sets. Figure 3.6 shows the day-to-day RMSE for the different calibration models for sensor C17016-s4. As it can be seen, there is a lot of variability from day to day, it does not seem that any of the four methods is able to perform much better than the others. This is partly due to the fact that ozone concentrations peak in the hottest months, mainly in July and August, and the calibration was performed under much lower ozone conditions in May and June. During the days of September, the error seems to be similar to the error after the calibration period since the concentrations in September and June can be quite similar.

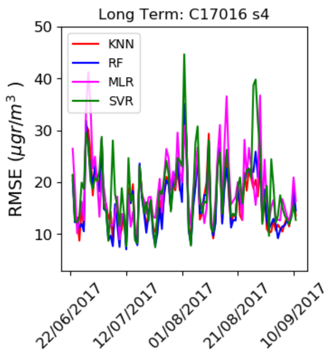


**Figure 3.6:** Daily test RMSEs for the long-term prediction for node C17016-s4.

To check how the performance of the different models evolves with time the best monitoring tool is the target diagram. The target diagram allows us to evaluate the two types of errors; CRMSE and mean bias. Several studies claim the need to recalibrate the

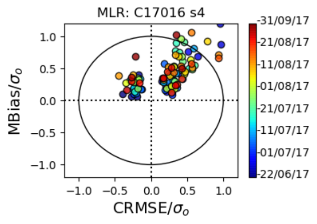
LCSs deployed in an area in order to maintain data quality [7]. Therefore, in this experiment, we show how the mean bias and CRMSE evolve over time for three different methodologies. First, we use the methodology used so far, where a pre-deployment calibration is performed and then the sensor is evaluated as time progresses. Second, we use a technique we call *"augmented training"* that simulates doing a pre-deployment calibration and then taking the sensor every three weeks for one week at a reference station to add one week of data to the training set. In this way, data are added containing the weather conditions prior to deployment and also current conditions with the weeks we augment the training set. In the real case, this technique increases maintenance costs as the sensor has to be placed periodically at a reference station, and also a week of monitoring data is lost, which can be critical for applications such as extreme event detection and early warning. And finally, we simulate a constant *"recalibration"* where the four previous weeks of the test week are used as training and each week the ML model is retrained with the four previous weeks. This technique is not feasible since the sensor would always be recalibrated, but it allows us to eliminate the impact of changing environmental conditions with respect to the training conditions.

Figures 3.7.a), d), g), and j) show the evolution of the CRMSE and Mean bias for the pre-deployment calibration for the four different ML models and the sensor C17016-s4. Blue dots denote the earlier stage of the deployment (late June) and the red dots denote the latest stage of the deployment (late September). The first thing observed is that the four models behave similarly, as the CRMSE and mean bias increase over time, but the nonlinear models exhibit less variance than the MLR. The variance of the target concentrations is sometimes overestimated and underestimated (positive and negative x-axis). The bluest and the reddest points correspond to similar environmental conditions, which is why the mean bias and CRMSE are smaller. This is because the conditions during the calibration period (May and June) are similar to the conditions during late September. Those points corresponding to the days of July and August tend to have increasing CRMSE and increasing mean bias. It is to say, as time evolves, and the environmental conditions change during the summer, the models start having more mean bias and more CRMSE. Indeed, at a given time, the points start falling out of the unit circle, this means that the trained models behave worse than the null model[5], so models start suffering from bias and excessive variance and become wrong.

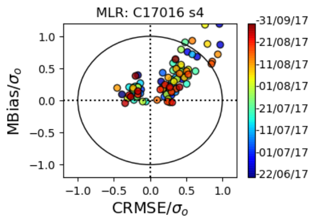
Figures 3.7.b), e), h), and k) show the same results for the augmented training technique. In this case, the models also suffer from a large mean bias and CRMSE as time progresses but it can be seen how some points are more centered around the MBias≈0.00. This shows, that augmenting the training set with a week every three weeks can mitigate a little bit the effects in the long-term, but in general, the training set may not be representative enough to cope with the environmental changes (e.g. episodes of ozone pollution, high temperatures, etc.) produced during the summer. Again, the four models behave similarly.

Finally, Figures 3.7.c), f), i), and l) show the results for the recalibration strategy. Here, the bias and CRMSE problem is improved as most of the points lay around MBias≈0.00. So that, the bias problem is reduced and there are only a few or no points that fall outside the circle, so the models do not degrade as much as in the initial case. Moreover,

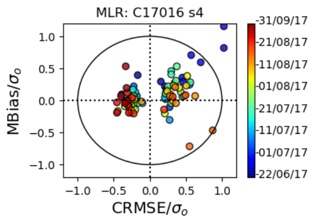
[5]The null model corresponds to constant prediction using the average concentrations of the training $\hat{y}_i = \bar{\mathbf{y}}_{tr}$.



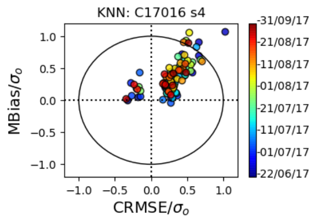
(a) MLR (three weeks of training size).



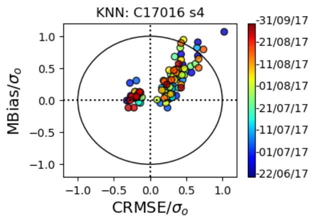
(b) MLR (training size augmented).



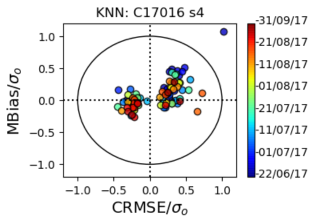
(c) MLR (recalibrated).



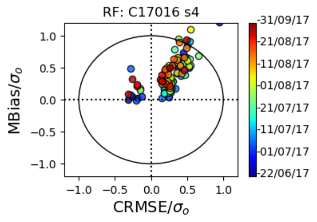
(d) KNN (three weeks of training size).



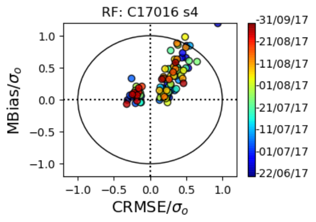
(e) KNN (training size augmented).



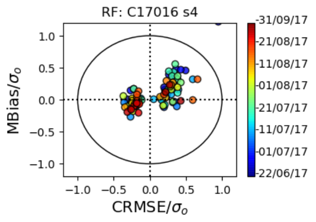
(f) KNN (recalibrated).



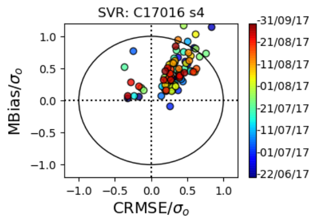
(g) RF (three weeks of training size).



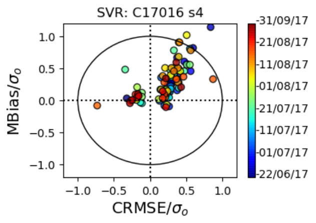
(h) RF (training size augmented).



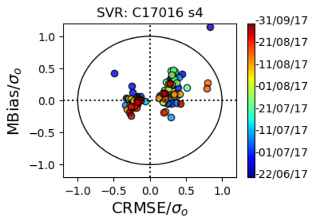
(i) RF (recalibrated).



(j) SVR (three weeks of training size).

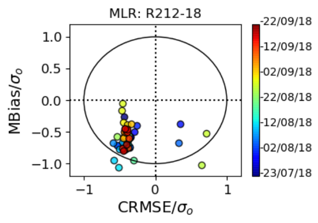


(k) SVR (training size augmented).

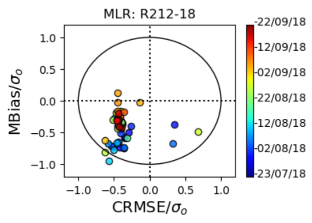


(l) SVR (recalibrated).

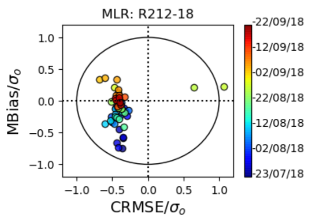
**Figure 3.7:** Long-term prediction: target diagram for Captor node C17016 (sensor s4). The left column corresponds to the pre-deployment calibration, the center column corresponds to the augmented training setting, and the right column corresponds to the continuous recalibration case.



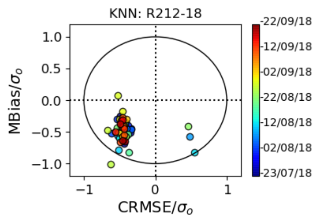
(a) MLR (three weeks of training size).



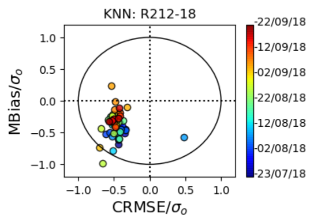
(b) MLR (training size augmented).



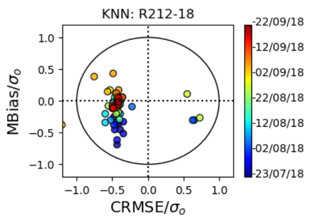
(c) MLR (recalibrated).



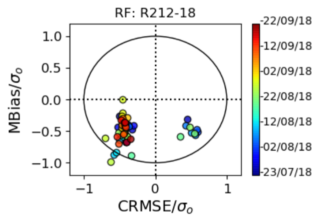
(d) KNN (three weeks of training size).



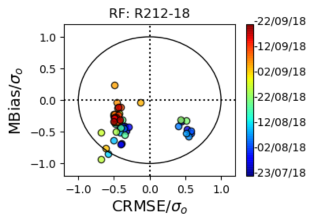
(e) KNN (training size augmented).



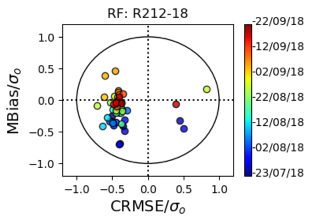
(f) KNN (recalibrated).



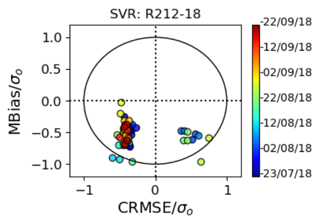
(g) RF (three weeks of training size).



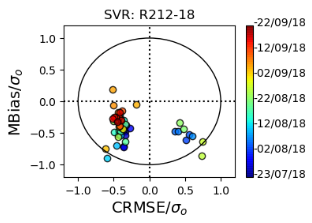
(h) RF (training size augmented).



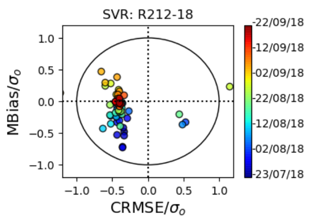
(i) RF (recalibrated).



(j) SVR (three weeks of training size).



(k) SVR (training size augmented).



(l) SVR (recalibrated).

**Figure 3.8:** Long-term prediction: target diagram for Raptor R212-18. The left column corresponds to the pre-deployment calibration, the center column corresponds to the augmented training setting, and the right column corresponds to the continuous recalibration case.

the nonlinear models also exhibit less variance since the points are more grouped together around a certain CRMSE and the MBias≈0.0. This basically shows how having a representative training set reduces the problems derived from the long-term prediction. However, this methodology is not feasible in practice given that the calibration periods are overlapped in order to have four weeks of representative data and predict the following week.

The long-term results for the Raptor R212 2018, Figure 3.8, are similar to the results of the Captor with small variations. First, Figures 3.8.a), d), g), and j) show that in this case the variance of the concentrations is usually underestimated (CRMSE<0.0) and that the models also suffer from bias but in this case negative bias MBias<0.0. Again, the nonlinear methods exhibit less variance than the linear method. The performance of the augmented training set approach, Figures 3.8.b), e), h), and k), is almost the same as the original case, only some points exhibit less bias but in general the improvement is very small. Finally, the recalibration strategy shown in Figures 3.8.c), f), i), and l) reduces the bias problem since most of the errors have a bias close to zero MBias≈0.0.

After having seen the long-term performance of metal-oxide and electrochemical sensors, we have observed that; *i)* calibration models start suffering from bias in the long-term predictions. This is mainly due to the change in the environmental conditions, *ii)* having scarce training data can lead to bias problems. The results of the short-term show that with a large and representative training set the bias has almost no contribution to the RMSE, and *iii)* the recalibration strategy reduces the long-term prediction problems. However, this technique may be infeasible as the maintenance costs and loss of data are too large. Smart recalibration strategies are required to deal with long-term performance.

## 3.5. Concluding remarks & Future work

This chapter has evaluated the use of ML models for the LCS calibration setting. This ML-based calibration has proved to be one of the most valid options to improve the quality of the data measured by LCSs. In this way, we have tackled the second research question raised in this thesis:

**(R.Q.1.2):** *How much do non-linear methods improve the calibration of ozone sensors compared to linear methods? How much data do they need? How do they work in the long term?*

A lot of effort has been put during the last years into the application of ML for sensor calibration [15, 16]. Some studies show the superiority of nonlinear models [27]. In this chapter, we have compared four ML models (MLR, KNN, RF, and SVR) using real data from H2020 Captor tropospheric ozone measurement campaigns of two different sensor technologies; metal-oxide and electrochemical. We have seen the ability of these models to predict short-term and long-term concentrations, seeing the limitations of this ML-based *in-situ* calibration in the long term. We have also studied the training size (or calibration time) required for each of the models to be able to predict in the short term. In short, we have answered the three corresponding questions, giving enough information for future projects and applications of these models in air pollution measurement campaigns.

Summarizing, we highlight the most important aspects observed in each of the sections:

**Short-term performance**

- Nonlinear models outperform the linear model by 2.00-4.00 $\mu gr/m^3$. Among the nonlinear models, there is no clear better model, but the SVR seems to perform slightly better by a very small margin.
- All models show no biases in the short-term performance, so the variance (CRMSE) is the term that contributes most to the error.
- There is a large variability in performance from LCS to LCS, probably due to the manufacturing process.

**Training set size**

- Nonlinear models require at least four calibration weeks, while the linear model requires at least three weeks.

**Long-term performance**

- All four models start degenerating after a few weeks and the bias starts dominating the error. The models become less accurate as the environmental conditions differ from the conditions in the training.
- None of the four models seems to work better than the other.
- The recalibration strategy seems to overcome this bias problem, yet its operational costs are high. However, it shows that when training the models with similar environmental conditions to the test, the long-term problems are reduced.

Future trends will focus on adaptive techniques and recalibration techniques to address the problems of long-term performance. Thus, the study of ML techniques will continue and will be used in conjunction with specific methodologies for sensors in order to reduce bias problems or sensor drifts. In fact, some research is already making use of techniques based on opportunistic recalibration and period recalibration [58]. As future research, it is interesting to study more complex calibration models with the ability to take into account further patterns present in air pollution phenomena. In the next chapter, we focus on how to take advantage of the intrinsic sensor variability to improve the ML-based calibration.

**Practical Tip !**

In the case of deploying a node with the highest possible data quality, one can place the sensor for approximately four weeks at a reference station to train *in-situ* a supervised nonlinear machine learning model and perform recalibration stages in the long term.

□ *For part III of the thesis, the LCSs are assumed to be calibrated in-situ using a machine learning approach.*

# 4

# Multisensor Low-Cost Sensor Calibration

*The best way to predict the future is to invent it.*

Alan Kay

As we have seen so far, low-cost sensors (LCSs) calibrated by machine learning (ML) techniques provide good estimates for monitoring air pollution gas species. Although all sensors of the same family measure a certain gas, it has been found that there is variability from sensor to sensor, possibly due to their low-cost nature and manufacturing process. Therefore, using an array of sensors of the same type can provide a good alternative to calibrating a single sensor. In fact, sensors of different gases have already been used in calibration in order to take advantage of the cross-sensitivities and correlations present between pollutants and sensors [14–16]. In addition, a first attempt to use several tropospheric ozone sensors in the calibration process by linear regression has already been studied [17].

Throughout this chapter, we study the fusion of sensors, which measure the same phenomenon, for the estimation of air pollution concentrations. Thus, a LCS calibration scheme is developed using a multisensor data fusion approach, where different LCSs are used to estimate the pollutant, showing how the performance changes as sensors are added. Furthermore, sensors from two different technologies widely used in the field of air monitoring, metal-oxide (MOX) sensors and electrochemical (EC) sensors, are fused.

The outline of this chapter is organized as follows; section 4.1 explains the state of the art regarding sensor fusion for air pollution estimation. Then, section 4.2 describes the methodology used for the multisensor data fusion approach. Section 4.3 describes the data sets used and the experiments' results. Finally, section 4.4 concludes the chapter. This chapter presents the findings made in the article "*Multisensor data fusion calibration in IoT air pollution platforms*", IEEE IoT-J, [63].

## 4.1. Sensor fusion using low-cost sensors

Sensor fusion has been widely used in the context of wireless sensor networks (WSN) [38]. In the WSN paradigm, sensor fusion is based on the aggregation of data in order to reduce the number of messages sent through the network, as well as its energy consumption. In the field of location and activity tracking using sensors, sensor fusion is defined as the integration of information from different sensors to combine the data provided by different sensors. Therefore, the fusion of measurements from inertial units of measurement (IMU) sensors using complementary filters and Kalman filters has been proposed [64]. In a similar way, Wu *et al.* [65] use a sensor fusion approach to combine different sensor time series in order to recognize daily activities. Data fusion techniques have also been used to improve the data reliability, and lower the detection error probability, by combining data from different distributed sources [66, 67].

In the air pollution monitoring field using LCSs, Barcelo-Ordinas *et al.* [10] define the term multisensor data fusion as: "*combination of information from two or more data sources (sensors) into a single one that provides a more accurate description than that of any of the individual data sources*". In this line, the use of several LCSs (sensor array) for sensor calibration or air pollution estimation has proven to be effective, taking advantage of the cross-sensitivities and cross-correlations present in the sensors measuring different phenomena [11, 15].

There are several examples of the use of sensor arrays for the calibration of LCSs. Bigi *et al.* [16] calibrated low-cost NO and $NO_2$ sensors using an array of one NO and one $NO_2$ sensor, feeding this array into linear and nonlinear ML models. Spinelle *et al.* [55] used an array of $NO_2$ and $O_3$ to calibrate an $O_3$ sensor using an artificial neural network. Similarly, Zimmerman *et al.* [15] used an array of CO, $NO_2$, $O_3$ and $CO_2$ sensors for the calibration of these sensors, using linear and nonlinear techniques. Mailings *et al.* [68] used a wide variety of ML techniques for calibration using a sensor array, for example, they calibrated a CO sensor using an array of CO, $SO_2$, $NO_2$, $O_3$, $CO_2$, T, and RH sensors. Lately, Zaidan *et al.* [69] used an array of $PM_{2.5}$, T, and RH to create $CO_2$ virtual sensors. Hence, the use of sensor arrays has not been limited to the calibration of LCSs, but also for the creation of virtual sensors and air pollution proxies. Although the use of sensor arrays has been studied previously, sensors for different pollutants have always been included. In our case, we study the use of a sensor array composed of sensors measuring the same pollutant, trying to take advantage of their inaccuracies and nonlinearities caused by the manufacturing process to improve the calibration with respect to using a single sensor. It is to say, combine the information provided by different sensors measuring the same phenomenon but with different noise and errors.

Multisensor data fusion techniques are widely used for many applications [70, 71]. For instance, fusion by weighted averages has been used for ultrasonic and infrared sensors with uncorrelated errors [72], and also in tracking applications with correlated errors [73]. Avery *et al.* [74] defined the optimal weight for a weighted averages fusion with correlated errors. As mentioned in the previous paragraph, data fusion using ML introducing the sensor array as features has been used for air pollution. Nevertheless, only one case has studied the use of a sensor array with sensors of the same pollutant to improve the estimation using multiple linear regression [17]. Thus, a set of up to four metal-oxide ozone sensors have been fused using multiple linear regression, showing improvements in estimation accuracy. The rest of the chapter is devoted to the use of sensor arrays of different sizes (multisensor data fusion calibration) to improve air pollution estimation, taking into account the possible problems that arise when dealing with correlated sensors. In the following section, we explain the different multisensor techniques evaluated as well as the multisensor data fusion framework proposed for *in-situ* calibration, showing the problems that arise when using highly correlated sensors in the calibration.

## 4.2. MULTISENSOR DATA FUSION CALIBRATION

THE purpose of using a multisensor calibration scheme is to use different sensors for the estimation of an air pollutant, therefore taking benefit of correlated sensors that bring additional information to the calibration model. As mentioned in the previous section 4.1, there exist several works that introduce complementary sensors into the calibration to boost it up [14, 15, 69]. However, we use the fact that LCSs have large variability within the same family of sensors, due to their low-cost nature [18]. Hence, it is interesting to implement a multisensor calibration scheme where several sensors measuring the same phenomenon are introduced to complement each one's inaccuracies and to obtain a more precise calibration model.

A multisensor *in-situ* calibration setting is defined by having a sequence of tuples $\{(y_i, \mathbf{x}_i)\}_{i=1}^{N}$, where $y_i \in \mathbb{R}$ is a sample of the values recorded by the reference instrumentation, $\mathbf{x}_i \in \mathbb{R}^P$ are a sample of the $P$ sensors' values introduced in the calibration. The proposed multisensor approach corresponds to the specific case where $P_s$ of the sensors introduced in the calibration measure the same air pollutant, $1 \leq P_s \leq P$. It should be noted that using sensors that are highly correlated with each other can lead to multicollinearity problems, so these problems must be addressed before the multisensor data fusion calibration can be performed.

### 4.2.1. MULTICOLLINEARITY

Multicollinearity occurs when two or more predictors of a ML model can be linearly described from other predictors of the model with a certain accuracy. Although multicollinearity does not have direct implications on the predictive quality of the ML model, it can affect its interpretability, as well as it can hinder the training process of some non-linear models, where the ideal case would be the use of independent predictors.

**Definition 9** ***Multicollinearity***: *given a set of predictors $\mathcal{X} = \{x_p\}_{p=1}^{P}$, multicollinearity occurs when two or more predictors $\{x_i, x_j\}$ are strongly correlated and can be described linearly fairly well using the rest of the predictors. The extreme case occurs when a predictor can be exactly described using a linear combination of the rest of the features $x_i = \beta^T \mathcal{X}_{-i}$, in this case, predictor $x_i$ does not introduce any new information to the model.*

In order to avoid possible problems, it is important to diagnose whether the data suffer from multicollinearity so that actions can be taken. There are many techniques to measure the degree of multicollinearity present in the data (e.g., non-significant regression coefficients or the moment matrix condition number) but in this work, we use the variance inflation factor (VIF):

$$\mathrm{VIF}(x_i) = \frac{1}{1 - R_i^2} \tag{4.1}$$

Where $R_i^2$ corresponds to the coefficient of determination where the predictor $\mathcal{X}_i$ is regressed with respect to the rest of the predictors. This VIF can be interpreted as the effect of the multicollinearity on each one of the predictors. As a rule of thumb, it is checked whether $\text{VIF}(x_i) > 10$ to detect predictors that may be affected by multicollinearity.

To perform the experiments to test the multisensor data fusion calibration, five different data sets have been used, which are explained in more detail in section 4.3.1. These data sets consist of different sensors that coincided during a calibration period in a reference station, making possible the introduction of different sensors in the multisensor calibration. Table 4.1, shows the correlation for some of the sensors that compose the first and the second data sets. It can be seen how the different tropospheric ozone sensors exhibit a high correlation between them, since they measure the same atmospheric phenomenon, but far from being a perfect correlation. Even though the Pearson correlation coefficient $\rho$ can reach 0.99 or 0.98, in most cases the correlation between two sensors can be around 0.8 and even much lower in some cases (e.g., $\rho$ =0.60). This correlation opens the door to the possible benefits of multisensor calibration, although it also shows the possible existence of multicollinearity problems in the calibration models.

**Table 4.1:** Pearson correlation coefficient ($\rho$) map for some sensors of data set 1 (left) and data set 2 (right).

| | ref | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 | Temp | RH |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ref | | | | | | | | | | | |
| s1 | 0.92 | | | | | | | | | | |
| s2 | 0.89 | 0.98 | | | | | | | | | |
| s3 | 0.92 | 0.99 | 0.97 | | | | | | | | |
| s4 | 0.92 | 0.98 | 0.95 | 0.98 | | | | | | | |
| s5 | 0.68 | 0.87 | 0.84 | 0.85 | 0.87 | | | | | | |
| s6 | 0.48 | 0.63 | 0.6 | 0.63 | 0.58 | 0.59 | | | | | |
| s7 | 0.87 | 0.96 | 0.96 | 0.97 | 0.93 | 0.82 | 0.73 | | | | |
| s8 | 0.88 | 0.98 | 0.98 | 0.97 | 0.95 | 0.87 | 0.64 | 0.97 | | | |
| Temp | 0.77 | 0.59 | 0.52 | 0.6 | 0.64 | 0.37 | 0.17 | 0.52 | 0.5 | | |
| RH | -0.66 | -0.54 | -0.49 | -0.57 | -0.57 | -0.39 | -0.32 | -0.52 | -0.49 | -0.82 | |

| | ref | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 | Temp | RH |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ref | | | | | | | | | | | |
| s1 | 0.87 | | | | | | | | | | |
| s2 | 0.74 | 0.95 | | | | | | | | | |
| s3 | 0.42 | 0.60 | 0.66 | | | | | | | | |
| s4 | 0.90 | 0.94 | 0.89 | 0.67 | | | | | | | |
| s5 | 0.63 | 0.91 | 0.90 | 0.65 | 0.80 | | | | | | |
| s6 | 0.68 | 0.92 | 0.97 | 0.68 | 0.83 | 0.91 | | | | | |
| s7 | 0.81 | 0.95 | 0.90 | 0.47 | 0.81 | 0.86 | 0.91 | | | | |
| s8 | 0.79 | 0.95 | 0.89 | 0.42 | 0.81 | 0.88 | 0.90 | 0.98 | | | |
| Temp | 0.66 | 0.33 | 0.11 | 0.09 | 0.44 | 0.06 | 0.07 | 0.27 | 0.28 | | |
| RH | -0.82 | -0.68 | -0.49 | -0.18 | -0.69 | -0.48 | -0.47 | -0.63 | -0.64 | -0.83 | |

In order to show if multicollinearity problems exist and to see how many sensors can be affected by this problem, the following experiment has been performed; *i)* random subsets $\S \subseteq \mathscr{X}$ of an increasing size of sensors have been selected (i.e., as if they were predictors), *ii)* the VIF has been calculated for each of the chosen sensors, *iii)* the percentage of sensors with a $\{\text{VIF}(x_i)>10 : x_i \in \S\}$ has been calculated, and *iv)* the average percentage and its confidence interval have been calculated for ten repetitions of each predictor set size. Figure 4.1 shows the experiments' results for data sets one and two. It can be seen how for subsets of four sensors about 40-70% of the sensors in the model have a VIF greater than 10, indicating multicollinearity problems. Thus, approximately two out of four sensors present multicollinearity problems. In addition, the results indicate a clear upward trend, where the more sensors are added to the calibration the more sensors suffer from multicollinearity, with as many as 80% to 90% of them.

These results indicate the potential problems with the correlation between sensors, and the possibility that it may affect the training process of some ML models. One of the most common solutions is the extraction of features that are independent of each other so decorrelating the predictors used. An example is the use of principal component analysis (PCA) to obtain orthogonal variables that do not present multicollinearity problems. In this approach, we use an alternative to PCA, the *partial least squares* (PLS), to obtain a set of orthogonal features and avoid learning problems. In the next section

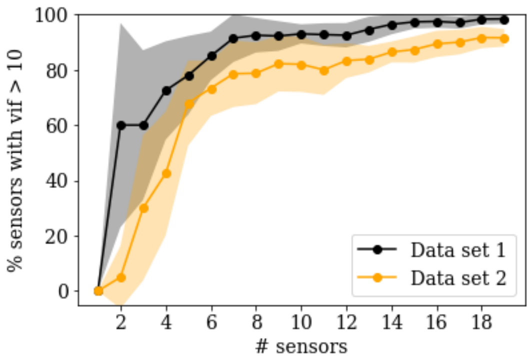


**Figure 4.1:** Average percentage of sensors with a VIF > 10, for subsets of sensors of increasing size. Bands indicate 95% confidence intervals for the average, calculated as a t-student.

4.2.2, the different methods for multisensor data fusion are explained, as well as the proposed method and the details of the use of the PLS to overcome multicollinearity. The experiments conducted to evaluate the different methods are explained in detail in section 4.3.

### 4.2.2. PROPOSED MODEL & OTHER MODELS

In this section, we explain the different models used for *in-situ* calibration and multisensor calibration. We focus on sensor data fusion using ML techniques, i.e., the objective is to learn the function $f_{fus}: \mathbb{R}^P \to \mathbb{R}$ whose input features are an array of sensors $\mathbf{x}_i \in \mathbb{R}^P$:

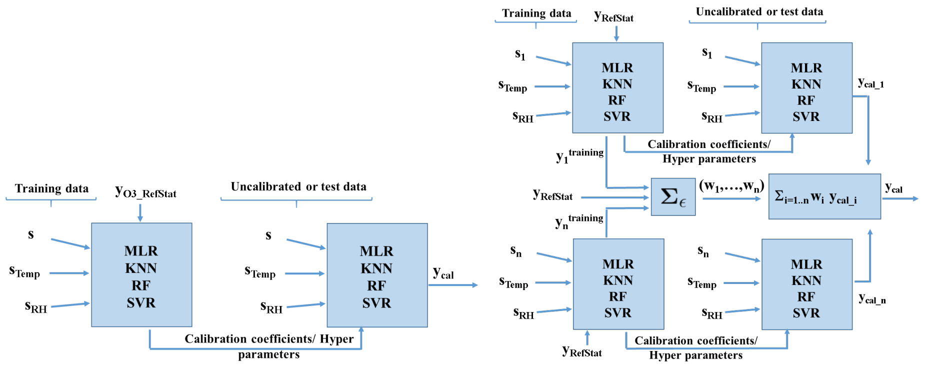
$$y_i \approx f_{fus}(\mathbf{x}_i) \tag{4.2}$$

Where $\mathbf{y} \in \mathbb{R}^N$ is the set of ground-truth values provided by a reference instrument, and the goal is to learn the function $f_{fus}(\cdot)$ using a ML model. Specifically, in this chapter we test three different models:

- **Sensor array calibration**: calibration using a simple sensor array, which is composed of only one target pollutant sensor ($P_s = 1$).

- **Multisensor data fusion with weighted averages**: the multisensor calibration ($P_s > 1$) is performed as a linear combination of the sensors' air pollution estimations, where the weights are computed based on the variances of the sensors' errors obtained during the training phase.

- **Multisensor data fusion with machine learning**: this is the proposed methodology, where a sensor array (with $P_s > 1$) is used to obtain derived PLS components that are orthogonal to avoid multicollinearity issues. Then, the derived components are introduced as features into a ML model, obtaining a multisensor estimation.



(a) Sensor array calibration scheme.

(b) Multisensor data fusion with weighted averages scheme.

**Figure 4.2:** Calibration schemes for the sensor array calibration and the data fusion approach using weighted averages.

## Sensor array calibration

In this case, the one studied in the previous chapter 3, the features introduced in the ML model are an array of sensors that includes a single sensor of the target pollutant, $P_s = 1$. An example of an array can be the one studied in the previous chapter, where the nodes collect measurements and form tuples such as $\{s_{O_3}, s_T, s_{RH}\}$ where we have a reading for the ozone sensor $s_{O_3}$, a reading for the relative humidity sensor $s_{RH}$, and a reading for the temperature sensor $s_T$. These tuples are then fed directly into a ML model, which can be either linear (MLR) or nonlinear (e.g., RF or SVR). This technique has been studied in the previous chapter 3, where ozone sensors have been calibrated using these tuples. To test the calibration, the data sets are divided into a training set (i.e., data collected during the calibration period) and a hold-out test set. Three steps are performed for calibration and prediction; *i)* a 10-fold CV process is performed over the training set to obtain the best set of hyperparameters for the nonlinear models, *ii)* models are trained using the whole training set and the best hyperparameters, and *iii)* trained models are applied over the test set. Figure 4.2.a) summarizes the process described for sensor array calibration where four ML models (MLR, KNN, RF, and SVR) are tested for calibration. Algorithm 1 in chapter 3 describes in detail all the ML-based calibration process, which goes from the training data $\mathbf{X}_{tr}$ to the final air pollution estimates $\hat{y}_{new}$.

## Multisensor Data Fusion With Weighted Averages

Sensor fusion has already been studied in many areas, including infrared sensors for target tracking applications [72–74]. One of the classical and most interpretable methods is the sensor fusion using weighted averages. Weights $w_p \in \mathbb{R}$ are obtained depending on whether the measurement errors, $\boldsymbol{\epsilon}_p = \hat{\mathbf{y}}_{cal_p} - \mathbf{y}$, are correlated or uncorrelated. To perform this type of sensor fusion, we assume that the $P_s$ sensors to be fused have been previously calibrated, having for each time step $i$ the tuples $\{y_i, (\hat{y}_{i,cal_1}, \ldots, \hat{y}_{i,cal_{P_s}})\}_{i=1}^{N}$, where $y_i$ are the ground-truth measurements of the reference instrument. Then, the estimation is obtained as follows:

$$\hat{y}_i = \sum_{p=1}^{P_s} w_p \hat{y}_{i,cal_p}, \quad i = 1,\ldots,N \tag{4.3}$$

Where the linear combination coefficients fulfill $\sum_p w_p = 1$. The optimal weights $\hat{\mathbf{w}}$ for the case of uncorrelated errors are [72, 74]:

$$\hat{w}_i = \frac{(\sigma_i^2)^{-1}}{\sum_{j=1}^{P_s} (\sigma_j^2)^{-1}} \tag{4.4}$$

Where $\sigma_i^2$ is the sample variance of the error produced by the $i$-th sensor, $\boldsymbol{\epsilon}_i$. The obtained linear estimator is the best in the sense that it is unbiased and efficient [74]. Nevertheless, there is a problem, as we have seen in the previous section, the sensors to be fused have a very high correlation with each other. Therefore, we must assume that the errors are correlated. To include these correlations present in the errors, we obtain the sample covariance matrix $\boldsymbol{\Sigma}_\epsilon$ of the errors, where the diagonal $diag(\boldsymbol{\Sigma}_\epsilon) = \{\sigma_1^2,\ldots,\sigma_{P_s}^2\}$ are the error variances of the different sensors. The optimal weights are obtained as [74]:

$$w_i = \frac{\sum_{j=1}^{P_s} (\boldsymbol{\Sigma}_{\epsilon_{ij}})^{-1}}{\sum_{j=1}^{P_s} \sum_{k=1}^{P_s} (\boldsymbol{\Sigma}_{\epsilon_{jk}})^{-1}} \tag{4.5}$$

Figure 4.2.b) shows the sensor fusion process for *in-situ* calibration using weighted averages. First, the $P_s$ sensors are calibrated *in-situ* using a ML model $f(\cdot)$, obtaining the estimates $\hat{\mathbf{y}}_{cal}$. Then, given the estimates of these models, and the reference values $\mathbf{y}$, the error covariance matrix $\boldsymbol{\Sigma}_\epsilon$ is calculated, from which the optimal fusion weights $\hat{\mathbf{w}}$ are derived. Finally, during the testing phase, the sensor values are estimated $\hat{\mathbf{y}}_{cal}$ and then the combination of the different estimates is performed $\hat{\mathbf{y}}_i = \hat{\mathbf{w}}^\top \hat{\mathbf{y}}_{i,cal}$, where $\hat{\mathbf{y}}_{i,cal} = [\hat{\mathbf{y}}_{i,cal_1},\ldots,\hat{\mathbf{y}}_{i,cal_{P_s}}]^\top$. It should be noted that although the estimator is unbiased, if one of the models starts to become biased, the combination will also be biased.

### MULTISENSOR DATA FUSION USING MACHINE LEARNING

In this chapter, we propose the multisensor data fusion using ML, which means that we introduce a multisensor array $\{s_{O_31},\ldots,s_{O_3P_S}, s_T, s_{RH}\}$ into a ML model, which can be either linear or nonlinear. This approach is entirely feasible given the LCSs price, which in general for both EC and MOX sensors is under 100 euros. For instance, the Captor node (the one used to obtain the data sets for this chapter) contains four tropospheric ozone metal-oxide sensors, raising its cost to around 300 euros. This means that, with a reduced cost overrun, more than one sensor can be mounted on an IoT node, so that redundancy and resilience can be achieved and the monitoring can continue even if a sensor fails (using simple sensor array calibration) or even improve the accuracy of the estimates (making multisensor data fusion with all the sensors). The idea behind this is to directly feed the raw values of the sensors into a ML model and let the model find the corresponding importance of each of the sensors automatically so that the estimation is as accurate as possible. In this way, given the multisensor array $\{x_{iP_1},\ldots,x_{iP_S}, s_{iT}, s_{iRH}\}_{i=1}^N$

and the ground-truth values of the reference station where the sensors have been placed $\mathbf{y} \in \mathbb{R}^N$, a calibration can be performed to estimate the concentrations using ML models. In this particular case, we test the models seen in the previous chapter 3 (MLR, KNN, RF, and SVR). Similarly, ML techniques have been used to calibrate sensors using sensor arrays containing sensors of other pollutants that are correlated with the target pollutant [14–16]. Previously, Barcelo-Ordinas *et al.* [17] proceeded with the fusion of a small subset of ozone sensors using MLR. Here, we evaluate in more detail the multisensor calibration approach; *i)* how linear and nonlinear models affect such calibration, *ii)* what is the optimal number of sensors to introduce in the fusion (we have about twenty sensors to calibrate), and *iii)* what happens when fusing ozone sensors from two different technologies (EC and MOX).

The first step in multisensor calibration using ML is to solve the multicollinearity problem, see previous section 4.2. Although multicollinearity does not necessarily affect the quality of the estimates, in the linear case it only hinders the interpretation of the coefficients and increases their variability, and in the case of nonlinear models it can hinder the model training process. For instance, in the case of k-nearest neighbors, having multicollinearity can give too much priority to some variables in the calculation of distances and make the interpretation of the results difficult. One of the most widely used solutions in the literature is the use of features derived from the original predictors that are independent of each other. An example is the use of Principal Components Analysis regression where a subset of derived components is used as features and the regression is calculated [75]. In this case, we opt for the Partial Least Squares (PLS) procedure since its objective is twofold; *i)* obtaining orthogonal components and *ii)* obtaining components correlated with the response variable. In fact, the PLS procedure reduces to obtaining orthogonal components $\hat{\boldsymbol{\phi}}_i$ (i.e., $<\hat{\boldsymbol{\phi}}_i, \hat{\boldsymbol{\phi}}_j >= 0$), where the $m$-th component can be found as:

$$\begin{aligned} \max_{\hat{\boldsymbol{\phi}}_m \in \mathbb{R}^P} \quad & \mathrm{Corr}^2(\mathbf{y}, \mathbf{X}\hat{\boldsymbol{\phi}}_m)\mathrm{Var}(\mathbf{X}\hat{\boldsymbol{\phi}}_m) \\ \text{s.t.} \quad & \|\hat{\boldsymbol{\phi}}_m\| = 1 \\ & \hat{\boldsymbol{\phi}}_m^T \hat{\boldsymbol{\Sigma}} \boldsymbol{\phi}_l = 0, \quad l = 1,..,m-1 \end{aligned} \tag{4.6}$$

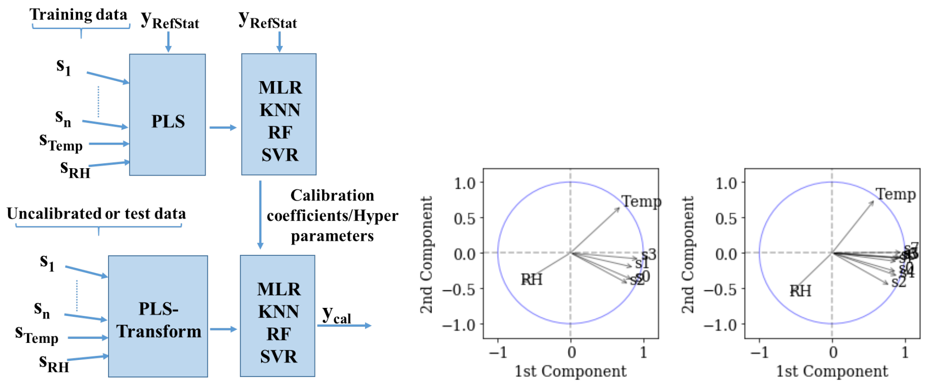
Where $\hat{\boldsymbol{\Sigma}} \in \mathbb{R}^{P\times P}$ is the sample covariance matrix. Then, we can compute the projection of the variables onto those derived components as:

$$\mathbf{P} = \mathbf{X}\boldsymbol{\Phi} \tag{4.7}$$

Where $\boldsymbol{\Phi} = [\hat{\boldsymbol{\phi}}_1, \dots, \hat{\boldsymbol{\phi}}_\kappa] \in \mathbb{R}^{P\times\kappa}$, $\kappa$ are the number of PLS components kept, $\mathbf{X} \in \mathbb{R}^{N\times P}$ is the matrix of sensor measurements, and $\mathbf{P} \in \mathbb{R}^{N\times\kappa}$ is the matrix of the projected variables. Specifically, in this case we use the $P$ components for the simple fact of introducing orthogonal features to the nonlinear training algorithms. Thus, it is decided to keep all the components ($\kappa = P$) in order to keep all the information, however small, introduced by the sensors measuring the same phenomenon. Therefore, the resulting design matrix is $\mathbf{P} = \mathbf{X}\boldsymbol{\Phi}_{\boldsymbol{P}}$, where $\boldsymbol{\Phi}_{\boldsymbol{P}} = [\hat{\boldsymbol{\phi}}_1, \dots, \hat{\boldsymbol{\phi}}_P]$. Then, the function $f : \mathbb{R}^P \to \mathbb{R}$ is estimated so that:

$$y_i \approx f(\mathbf{p}_i), \ \ i = 1, \dots, N \tag{4.8}$$

Figures 4.3.b) and c) show a small example of the subspaces obtained with the PLS pro-



(a) Multisensor data fusion calibration using a machine learning scheme.

(b) Variables' projection to the first two PLS components using data set 1 including only four sensors.

(c) Variables' projection to the first two PLS components using data set 1 including only eight sensors.

**Figure 4.3:** On the left, the scheme for the calibration using multisensor data fusion with ML. On the right, an example of the projection of the variables in the first two PLS components.

cess using data set 1 with four and eight sensors. It can be observed the projection of the different variables into the first two PLS components, all the variables have a high contribution to these components. It is also observed by looking at the different angles between the projections and the components that the first component is closely related to the different sensors, although not all of them are in the same way, which is logical since they do not have a perfect correlation. It can also be seen how the temperature is related to the second component and the relative humidity has an inverse relationship with the temperature. In short, the first component can be associated with the ozone measured by different sensors.

Figure 4.3.a) shows the multisensor fusion process using supervised ML. First, the different PLS components $\mathbf{\Phi}_P$ are obtained, then the samples are projected onto the subspace generated by these components, and the model $f(\cdot)$ is trained using these projections $\mathbf{P}_P \in \mathbb{R}^{N\times P}$. During the deployment period, first, the samples $\mathbf{x}$ are projected ($\mathbf{p}$) and then the estimation $\hat{y}$ is obtained using the trained model. Algorithm 2 explains the whole process in more detail, showing all required steps needed to go from the raw sensor data $\mathbf{x}$ to the air pollution estimates $\hat{\mathbf{y}}$.

The following section explains the different experiments carried out to evaluate the accuracy of the sensor fusion approach for sensor calibration.

## 4.3. EXPERIMENTAL EVALUATION

In this section, we show the experiments performed to evaluate the multisensor data fusion calibration. The goal is to evaluate whether sensor fusion can obtain an improve-

**Algorithm 2** Multisensor data fusion calibration using ML.

```
Input: ${\mathbf{X}_{tr}, \mathbf{y}_{tr}, f(\cdot)}$
1: $\bar{\mathbf{X}}_{tr} \leftarrow$ Standardization($\mathbf{X}_{tr}$)
2: $\bar{\mathbf{y}}_{tr} \leftarrow$ Standardization($\mathbf{y}_{tr}$)
3: $\mathbf{\Phi}_{\boldsymbol{P}} \leftarrow$ PLS($\bar{\mathbf{y}}_{tr}, \bar{\mathbf{X}}_{tr}$)
4: $\mathbf{P}_P \leftarrow \bar{\mathbf{X}}_{tr}\mathbf{\Phi}_{\boldsymbol{P}}$
5: hyp $\leftarrow$ Cross_Validation($f(\cdot), \mathbf{P}_P, \bar{\mathbf{y}}_{tr}$)        ◁ Model Selection
6: $\hat{\bar{f}} \leftarrow$ Train_Model($f(\cdot), \bar{\mathbf{X}}_{tr}, \bar{\mathbf{y}}_{tr}$, hyp)        ◁ Model Training
7: while $\mathbf{x}_{new}$ do                                  ◁ Prediction Phase
8:     $\bar{\mathbf{x}}_{new} \leftarrow$ Standardization($\mathbf{x}_{new}$)
9:     $\mathbf{p} \leftarrow \bar{\mathbf{x}}_{new}^T \mathbf{\Phi}_{\boldsymbol{P}}$
10:    $\hat{\bar{y}}_{new} \leftarrow \hat{\bar{f}}(\mathbf{p})$
11:    $\hat{y}_{new} \leftarrow$ Unstandardization($\hat{\bar{y}}_{new}$)
12:    RETURN $\hat{y}_{new}$
13: end while
```

ment with respect to the results of a sensor array *in-situ* calibration [14, 15, 30, 69][1]. More specifically, we describe the different data sets used to evaluate the methods and we explain the results obtained. Given the available data sets, multiple MOX ozone sensors, and in some cases even an EC ozone sensor, two experiments have been performed:

(A) **Fusion of sensors of the same technology**: the objective is to evaluate the accuracy of the fusion of sensors of the same technology. The first two data sets contain about twenty MOX sensors that were placed at the same site for a few weeks.

(B) **Fusion of sensors of different technologies**: the objective is to evaluate the accuracy of the fusion of sensors from different technologies to see whether they provide complementary information to each other. The other three data sets used contain four MOX sensors and one EC sensor, allowing the inclusion of an EC sensor in the sensor fusion.

The sensor array calibration scheme has been used when only one ozone sensor was present in the calibration and the two multisensor techniques (weighted averages and ML) have been used for scenarios with more than one ozone sensor introduced in the calibration. To evaluate the different models, the different data sets have been randomly divided into 75% of the data for model selection and training, and 25% as hold-out test set to evaluate the performance of the different models used. 75% of the data are used for the selection of models' hyperparameters by 10-fold cross-validation and the final model is trained with these same data. Then, the performance of the different fusion models and ML algorithms is compared in terms of root-mean-squared error (RMSE) on the test set.

The aim is to evaluate how the RMSE evolves as more sensors are added to the multisensor data fusion calibration. Thus, it is possible to see how much the addition of

[1]A python implementation of the proposed data fusion method for in-situ calibration is available at https://bitbucket.org/sans-rg/iot-sensor-fusion/.

more sensors improves and to evaluate whether it is feasible to use more than one sensor. To do this, first, we find which is the best sensor of the data sets using the sensor array calibration, we name the best sensor as $s_{best}$. Then, to evaluate the performance of the multisensor calibration with two sensors, we create random subsets of sensors of size two that always include the best sensor $s_{best}$. Then, for subsets of three sensors, the best sensor $s_{best}$ is fixed and the other two are randomly chosen, and so on. Hence, we define the random subset of selected sensors of size $k$ as $\mathscr{S}_k = \{s \in \mathscr{S}_k : \mathscr{S}_k \subseteq \mathscr{S} \wedge |\mathscr{S}_k| = k \wedge s_{best} \in \mathscr{S}_k\}$. The average test RMSE is used as the performance metric, as well as the 95% confidence interval of these averages computed as a t-student.

## 4.3.1. DATA SETS

Two different nodes have been used, obtaining different data sets. First, *Captor* nodes, as mentioned in the previous chapter, developed by UPC, have been used as IoT nodes to collect air pollution measurements. These nodes have an Ardunio Nano as microcontroller unit, a 3G modem to send data to the cloud, four ozone MICS 2614 Sensortech sensors, a temperature sensor, and a relative humidity sensor. Different nodes were deployed during 2017, 2018, and 2019 (their labels start with the year of deployment, e.g., C19000 of 2019), where they remained at reference stations for calibration tasks and to validate the accuracy of the different sensors. In fact, three separate testbeds were created in Spain, Austria and Italy, within the framework of the European H2020 program. More than twenty-five sensor nodes were deployed to make a measurement campaign and raise social awareness of air pollution [8].

The Captor node has collected measurements every 1 minute, where it has read the measurements from the different sensors, and these measurements have been aggregated in one-hour periods in order to be synchronized with the reference station measurements and to be able to perform the calibration. In this way, tuples {timestamp, $s_1, s_2, s_3, s_4, s_T, s_{RH}$} have been obtained, where the ozone sensor measurements are in $k\Omega$, the temperature measurements are in degrees Celsius ($^\circ C$) and the relative humidity measurements are in percentages (%). These tuples results are sent to be stored in a central database. As an exception, the C19000 node mounted an EC ozone sensor Alphasense OX-B431 and an Alphasense NO2-B43F.

The other nodes used in the study are called *Raptors*, and were developed by the University of Clermont also within the framework of the H2020 Captor program. In this case, the node architecture was similar but with a Raspberry Pi as processing unit, and with EC ozone sensors. In addition, the node was separated in a server to be inside the houses and the client to collect the pollution measurements outdoors. Samples were also obtained with the same temporal resolution, i.e., 1 hour. Table 4.2 shows the main characteristics of the data sets, the stations where the sensors were matched for calibration, the year, and other characteristics. In the case of fusion by MOX sensors, data sets one and two have twenty-eight and twenty-four sensors respectively. In the case of EC fusion, there is one EC sensor and four metal-oxide sensors. The Palau Reial station is an urban station in Barcelona, Spain. The others are from a more inland area of Catalonia. As for the sizes of the data sets, the shortest corresponds to one month of data and the longest corresponds to three months of data.

All data used have been made available contributing to open data initiatives. The

**Table 4.2:** Data sets used to evaluate the multisensor data fusion calibration.

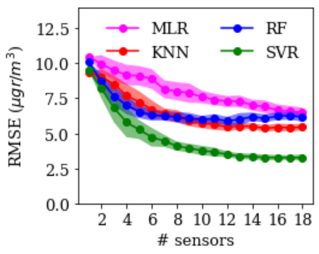
| Data Set Label | Reference St. Name | Year | Node Labels | Technologies | # Sensors | # of Samples |
|---|---|---|---|---|---|---|
| 1 | Manlleu | 2017 | C17001, C17002, C17003, C17005, C17010, C17011, C17013 | MOX | 28x MICS 2614 | 918 |
| 2 | Tona | 2017 | C17006, C17007, C17012, C17014, C17017, C17027 | MOX | 24x MICS 2614 | 1395 |
| 3 | Tona | 2017 | R69-17 | EC | 1x OX-B431 + 1x NO2-B43F | 2366 |
| | | | C17017 | MOX | 4x MICS 2614 | |
| 4 | Tona | 2018 | R69-18 | EC | 1x OX-B431 + 1x NO2-B43F | 933 |
| | | | C18017 | MOX | 4x MICS 2614 | |
| 5 | Palau Real | 2019 | C19000 | EC | 1x OX-B431 + 1x NO2-B43F | 1179 |
| | | | C19027 | MOX | 4x MICS 2614 | |

data from the reference stations of Catalonia (Spain) are available on the government's open data website [2]. Data from the 2017 H2020 CAPTOR air quality monitoring campaign are available [9]. The rest of the data are available for download on the following page[3].

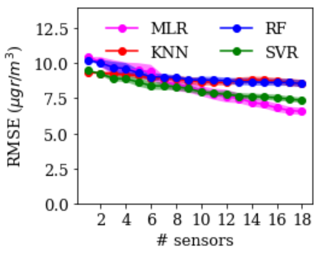
### 4.3.2. Sensor fusion calibration: Same Sensing Technology

In this experiment, data sets 1 and 2 corresponding to the sensors located at the reference stations of Manlleu and Tona (Spain) have been used. Given that each of these data sets contains twenty-four and twenty-eight sensors respectively, it is possible to test the multisensor data fusion with many sensors. First, Figure 4.4.a) shows the results for the multisensor calibration with different ML models applied to data set 1 (*Manlleu* data set). It can be seen how for the single sensor calibration, scenario corresponding to the sensor array calibration with the best sensor $s_{best}$ of the sensor set $\mathscr{S}$, the MLR is able to obtain 10.50 $\mu$gr/m$^3$ and the nonlinear models are able to reduce this error down to 9.50 $\mu$gr/m$^3$, without seeing a clear outperforming nonlinear model. This result is in agreement with the one obtained in the previous chapter, where the nonlinear models provide an improvement over the MLR. Secondly, we observe that the general trend is the same for all four ML models, the RMSE decreases as we add sensors to the calibration. The main differences are observed as soon as sensors are added to the calibration, where the nonlinear models actually benefit more from adding more sensors. Specifically, we see how SVR is the model that observes a greater improvement as MOX sensors are added to the calibration, going from obtaining 9.50 $\mu$gr/m$^3$ with one sensor to obtaining around 5.30 $\mu$gr/m$^3$, reducing the error 4.00 $\mu$gr/m$^3$ with a total of four sensors. A similar trend is also observed for the other models, but they do not improve their RMSE as much as sensors are added. In fact, at the other extreme, the MLR is only able to reduce by about 1.00 $\mu$gr/m$^3$ by adding four sensors to the calibration, indicating that when performing an MLR calibration the overhead of adding sensors may not be feasible for the small improvement that occurs. The most important thing to note is that with around four or six sensors introduced in the calibration, the RMSE stops improving significantly, this means that up to four sensors can produce a worthwhile improvement, e.g., 4.00 $\mu$gr/m$^3$ in the case of calibration with SVR.

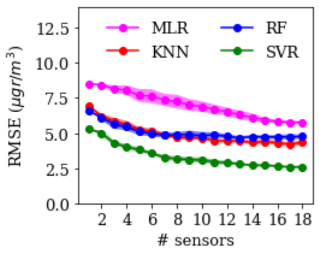
[2]Catalonia air pollution open data web.
[3]Raw sensor data is available at the research group page http://sans.ac.upc.edu/?q=node/231.



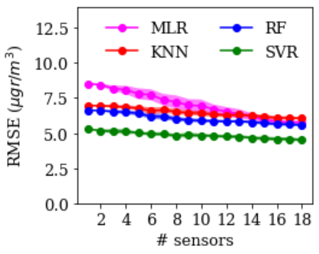
(a) Average test RMSE for data fusion by ML using data set 1.



(b) Average test RMSE for data fusion by weighted averages using data set 1.



(c) Average test RMSE for data fusion by ML using data set 2.



(d) Average test RMSE for data fusion by weighted averages using data set 2.

**Figure 4.4:** Average test RMSE for 10 repetitions with random subsets of sensors (always including the best sensor $s_{best}$) and bands indicating 95% confidence intervals. Results for *in-situ* calibration via sensor fusion using data sets 1 and 2.

On the other hand, Figure 4.4.b) shows the results of the fusion using weighted averages with each of the ML models used for the individual calibration of each of the sensors. It can be seen how in the case of having a single sensor, $s_{best}$, the models have the same error as in the previous case since it corresponds to the sensor array calibration of the best sensor. Regarding the trends, we can see how the error improvement trends are very different in the case of the nonlinear models with respect to the previous case. Besides, it is also observed that for the linear model the trend is very similar, this means that fusing the $k$ sensors by MLR has a performance similar to having the sensors calibrated individually and then combining these measurements by weighted averages (see section). This makes sense since the combination of sensors using weighted averages is still a linear model, so the linear combination of something predicted with MLR is similar to putting the raw values of the sensors in the fusion with multiple linear regression. As for the weighted average fusion of calibrated data with nonlinear methods, we see that the fusion loses the efficiency seen previously, the RMSE improvement is much reduced compared to the best seen in the multisensor data fusion with ML. This is because weighted averages of calibrated values with nonlinear models have much less flexibility than inputting all raw values to a ML nonlinear model and letting the model do the fusion as efficiently as possible. That is, the fusion (Figure 4.4.a)) with nonlinear models is able to overcome the multicollinearity problem and take advantage of important and complementary information from each of the sensors used in the calibration.

Figures 4.4.c) and d) show the same results for data set 2. Figure 4.4.c) shows the results of the ML fusion, where the main difference is in the performance of the sensor array calibration. That is, in the case $k = 1$, the different models have a very different error, on the one hand, the MLR obtains about 8.50 $\mu$gr/m$^3$ while the SVR obtains about 5.10 $\mu$gr/m$^3$. On the other hand, it is also observed that the improvement when introducing sensors is lower than in the case of the first data set, although there is the same tendency where the nonlinear models have a more significant improvement when adding sensors while the linear model improves much less. As an example, we see how the SVR obtains a test RMSE of 5.10 $\mu$gr/m$^3$ with one sensor and is able to improve up to 3.50 $\mu$gr/m$^3$ with four sensors and can approach 2.50 $\mu$gr/m$^3$ test RMSE by adding more sensors. As for the optimal number of sensors, we see how with four or six sensors the improvement is significant. The other nonlinear models (KNN and RF) have a trend very similar to SVR, but as they obtain more error with a single sensor, they are not able to reach around 2.50 $\mu$gr/m$^3$ test RMSE like SVR. As for the fusion with the weighted averages, Figure 4.4.d), it is observed the same trend as with the data set 1 where the fusion with calibrated data with the MLR obtains a trend in the improvement very similar to the multisensor data fusion with MLR. On the other hand, the fusion of calibrated data with nonlinear models does not show much improvement, unlike those seen in data set 1, the initial error is the same, but the improvement when adding sensors is not significant.

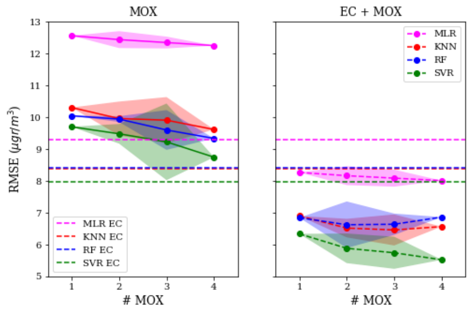
In short, the results indicate that including more than one MOX sensor in the calibration improves the performance of the calibration resulting in reductions of 20-40% of the RMSE, and the fusion using nonlinear ML models greatly increases the improvement over a weighted average fusion.

### 4.3.3. SENSOR FUSION CALIBRATION: DIFFERENT SENSING TECHNOLOGY

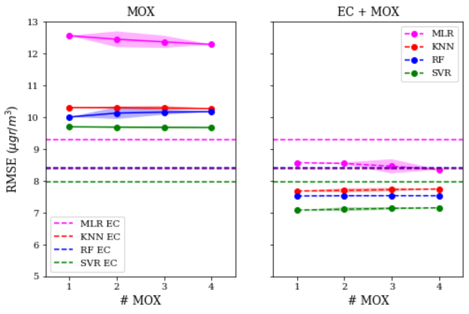
In this second experiment, the calibration by multisensor data fusion is evaluated when ozone sensors of different technology (EC and MOX) are introduced. For this purpose, data sets 3, 4 and 5 are used, since they contain one EC and four MOX sensors. This configuration is perfect because as we have seen in the previous section 4.3.2, introducing four sensors in the calibration is already enough to improve the calibration performance a lot. Therefore, first the fusion of the four MOX is evaluated without taking into account the EC sensor (as in the previous experiment), and then the EC sensor is used as baseline and the MOX sensors are added one by one. The sensor array calibration performance using only the EC ozone sensor is denoted in dotted lines.

First, Figure 4.5.a) shows the results for the multisensor calibration of data set 3 using ML. On the left side, the fusion using only MOX sensors, we see how the SVR is the method that performs the best (as in the previous cases), with one sensor it obtains a test RMSE of 9.70 $\mu$gr/m$^3$ while with four sensors it obtains an error of 8.70 $\mu$gr/m$^3$. Even so, the calibration using four MOX sensors does not improve the simple sensor array calibration using only the EC sensor, which obtains an RMSE of 8.00 $\mu$gr/m$^3$, there is a difference between the accuracy of the EC sensor and the MOX sensor. Then, on the right (EC + MOX fusion) the fusion between the EC sensor and at least one MOX sensor greatly improves the test RMSE. In the case of the SVR the error is able to go down from

8.00 $\mu$gr/m$^3$ to 6.40 $\mu$gr/m$^3$ with only two sensors (one EC and one MOX), then adding more MOX sensors improves the error a little more to an error of 5.80 $\mu$gr/m$^3$. The results for the weighted averages fusion, Figure 4.5.b), show the same trend as in the previous experiment, where the fusion using data calibrated with MLR works well, but in the case of weighted averages fusion using data calibrated with nonlinear models the improvement is not significant at all. In addition, in the case of fusing an EC sensor with a MOX, there is also an improvement, but not the same as with the ML fusion, the error with SVR is reduced from 8.00 $\mu$gr/m$^3$ to 7.20 $\mu$gr/m$^3$.



(a) Average test RMSE for MOX fusion and EC + MOX fusion for data set 3, using ML fusion.

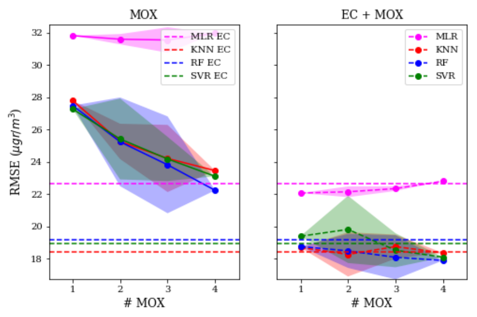


(b) Average test RMSE for MOX fusion and EC + MOX fusion for data set 3 using weighted averages fusion.

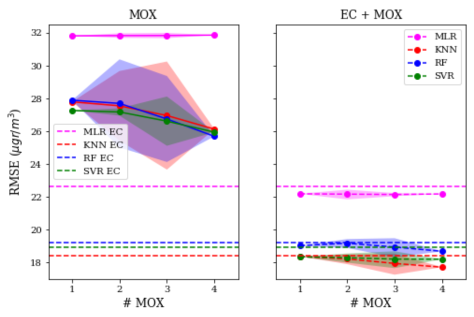
**Figure 4.5:** Average test RMSE for data set 3 using just one EC sensor and data fusion average test RMSE using one EC sensor and several MOX sensors. Bands indicate 95% confidence intervals for the mean.

Figure 4.6.a) shows the same results for data set 4. First, we see how MOX fusion only obtains a significant improvement from 27.50 $\mu$gr/m$^3$, in the case of SVR and one sensor, to 23.75 $\mu$gr/m$^3$ in the case of four MOX sensors and the same model. However, it can be seen that the estimates in this data set have a much larger error than in the previous case. More specifically, an RMSE of 19.00 $\mu$gr/m$^3$ is obtained with the SVR and the sensor array calibration with the EC sensor. If we look at the results of merging the two technologies we see very different results, where combining them practically does not improve the estimation, and neither does adding more sensors. In the case of weighted average fusion, Figure 4.6.b), we observe the same trend, where fusing EC sensors with MOX sensors does not improve the prediction. This basically tells that when one of the fused technologies performs much worse than the other, the fusion is not better since it will be dominated by the sensor that performs well.

Finally, the results for data set 5, Figure 4.7.a) and b), show the opposite case to the previous one, MOX sensors performing better than the EC sensor. First, Figure 4.7.a), shows the fusion by ML where we see the same trend of decreasing error as we add more sensors in the multisensor data fusion calibration. For example, in the RF case, the error with one MOX sensor is 17.70 $\mu$gr/m$^3$ and the error with four MOX is 16.60 $\mu$gr/m$^3$, reducing the RMSE by 1.10 $\mu$gr/m$^3$. In the subplot on the right (EC + MOX fusion). Figure 4.7, we see how the EC sensor performs worse than the MOX sensors, obtaining an RMSE of 19.60 $\mu$gr/m$^3$ for the RF and the sensor array calibration sensor. Then, we see

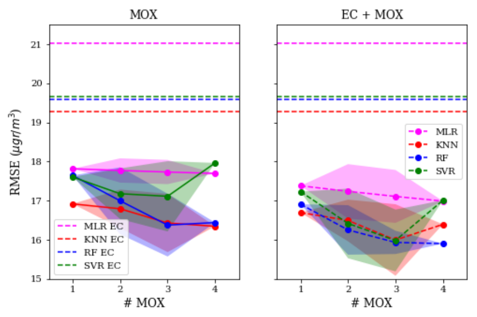


(a) Average test RMSE for MOX fusion and EC + MOX fusion for data set 4 using ML fusion.

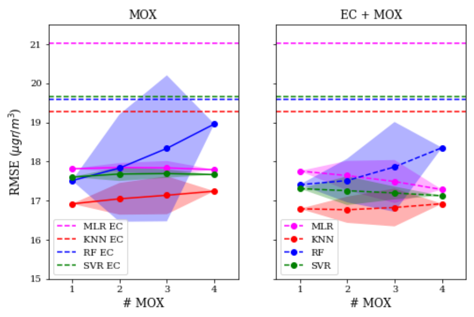


(b) Average test RMSE for MOX fusion and EC + MOX fusion for data set 4 using weighted averages fusion.

**Figure 4.6:** Average test RMSE for data set 4 using just one EC sensor and data fusion average test RMSE using one EC sensor and several MOX sensors. Bands indicate 95% confidence intervals for the mean.



(a) Average test RMSE for MOX fusion and EC + MOX fusion for data set 5 using multisensor data fusion approach.



(b) Average test RMSE for MOX fusion and EC + MOX fusion for data set 5 using weighted averages fusion approach.

**Figure 4.7:** Average test RMSE for data set 5 using just one EC sensor and data fusion average test RMSE using one electrochemical sensor and several MOX sensors. Bands indicate 95% confidence intervals for the mean.

how adding the EC sensor to the MOX fusion does not improve significantly with respect to the case of having only MOX sensors. Specifically, fusing an EC sensor with a MOX sensor only improves over having a single MOX sensor by 0.60 $\mu$gr/m$^3$. As for the weighted averages fusion, Figure 4.7.b), we see the same trends as in the previous case, where the fact of fusing the two technologies does not incur a significant improvement since the EC sensor has much worse performance than the MOX sensors. It is observed that the fusion by weighted averages of RF-estimated data is not performing well, indicating a possible sensor obtaining a bad estimation, thus, increasing the confidence bands. Moreover, as we have seen in the other experiments, the fact of merging by means of weighted averages of the calibrated sensors with nonlinear models does not offer any improvement, even in the case of the RF it is observed that the error worsens.

In short, the results indicate that the fusion of two technologies (in this case EC and MOX sensors) works better the similar their performances are. In fact, in the case of having a similar error, the fusion of an EC sensor and a MOX sensor has achieved an improvement of 2.00 $\mu$gr/m$^3$ in the RMSE over having a single sensor.

## 4.4. CONCLUDING REMARKS & FUTURE WORK

In this chapter, we have investigated the *in-situ* calibration of LCSs in the case where the sensor array includes more than one sensor measuring the same air pollutant. Thus, the research question posed at the beginning of the thesis has been solved:

**(R.Q.1.3):** *Does having replicated sensors on a node improve the calibration accuracy? Is it possible to perform a calibration based on data fusion and machine learning?*

The purpose has been to take advantage of the inaccuracies and variability that LCSs have, due to their low-cost nature, to check whether sensors can provide complementary information to each other. Using experimental data from both MOX and EC ozone sensors, it has been observed that the correlations between sensors are not perfect and that the inclusion of different sensors can provide benefits. For this purpose, the multisensor data fusion calibration with ML has been proposed, where firstly new independent features have been derived from the sensors by means of partial least squares to overcome the multicollinearity problems. Then, these derived variables have been introduced to a ML model to produce the final calibration. The results have shown how adding more than one sensor of the same technology (MOX) in the fusion significantly improves the calibration performance, and can improve up to 3.00 $\mu$gr/m$^3$ by adding four sensors. In the case of merging two different technologies (EC and MOX), it has been observed that the improvement occurs when the two technologies have a good performance, so with one sensor of each technology a significant improvement is obtained. On the other hand, when one of the two technologies has a bad performance, the fusion practically does not benefit the calibration model.

The multisensor data fusion approach has proven to be useful since LCSs have a very low price (e.g., about 40$ in the case of the Sensortech MICS 2614), so adding more than one sensor in a monitoring node not only adds robustness to the system to continue measuring in case of sensor failure, but all sensors can be used to enhance the node's calibration. Thus, it is feasible to add up to four sensors in a node to improve the accuracy of the calibration in exchange for a slight increase in the node's cost.

**Multisensor data fusion calibration**

- The multisensor data fusion approach using ML has proven to be superior to the weighted averages fusion approach.
- Among the ML models used to perform data fusion, the nonlinear models have outperformed the linear model, especially the SVR has performed better than the others in most cases.
- This approach has proven to be feasible, as using a few sensors (e.g., four sensors) has improved the air pollution estimation significantly. Given the low price of these

sensors, introducing up to four sensors adds a small additional cost in exchange for improved data quality and robustness.

**Fusion of sensors of the same technology**

- The fusion of sensors of the same technology has resulted in an increase in calibration quality. In particular, the inclusion of up to four MOX sensors has reduced the RMSE by 3.00 $\mu$gr/m$^3$.

**Fusion of sensors of different technology**

- The fusion of sensors of different technologies (in this case MOX and EC) is beneficial in the case that both technologies have a similar performance. The results show that one sensor of each technology is sufficient to reduce the RMSE by more than 1.50 $\mu$gr/m$^3$.
- In the case that one of the two technologies performs significantly worse than the other, the improvement is limited by the latter. Thus, the fusion does not provide a significant improvement over the better-performing sensor.

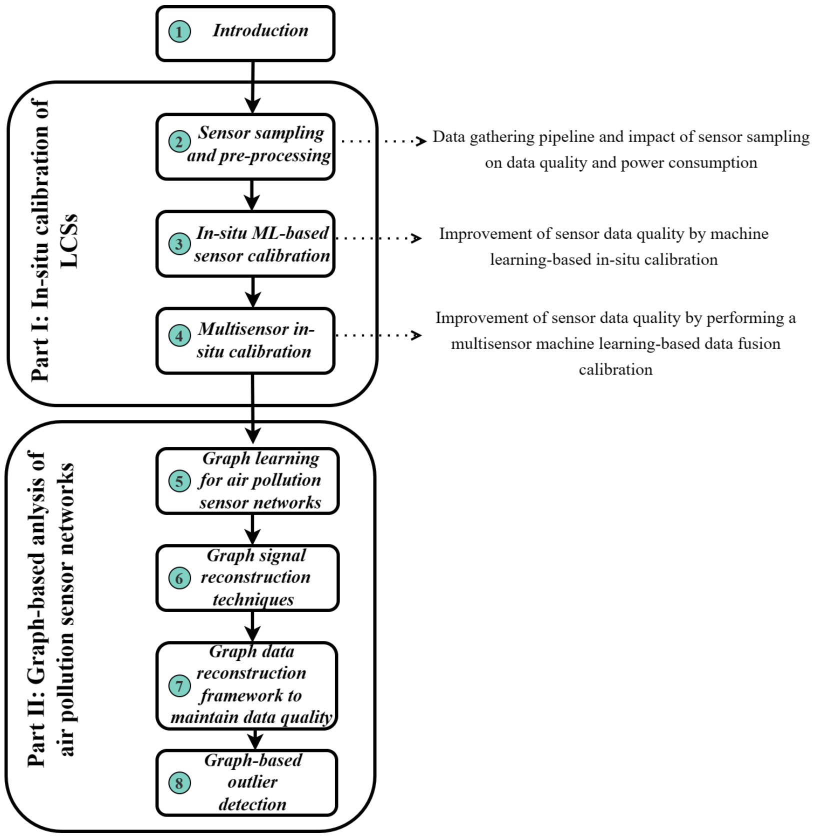
As a future trend, it would be interesting to see how using a multisensor calibration approach can help in the long-term prediction. In fact, having more than one sensor in a node could help detect if any of the sensors have drifted or could help compensate for the errors of these sensors. In fact, the regulated introduction of multiple sensors not only to improve the accuracy but to increase the overall reliability of pollution estimates is another aspect to study of the multisensor approach.

**Practical Tip !**

Given the inaccuracies of LCSs, in the case of wanting to improve the accuracy of pollution estimates and even the resilience of the device, one can mount at least two sensors on the node, whether they are of the same technology or not, and perform a multisensor calibration based on machine learning.

# II

# Graph-Based Analysis of Air Pollution Sensor Networks


1 Introduction
Part I: In-situ calibration of LCSs
2 Sensor sampling and pre-processing
Data gathering pipeline and impact of sensor sampling on data quality and power consumption
3 In-situ ML-based sensor calibration
Improvement of sensor data quality by machine learning-based in-situ calibration
4 Multisensor in-situ calibration
Improvement of sensor data quality by performing a multisensor machine learning-based data fusion calibration
Part II: Graph-based anlysis of air pollution sensor networks
5 Graph learning for air pollution sensor networks
6 Graph signal reconstruction techniques
7 Graph data reconstruction framework to maintain data quality
8 Graph-based outlier detection

# 5
# Representing Air Pollution Sensor Networks With Graphs

*Scientists have become the bearers of the torch of discovery in our quest for knowledge.*

Stephen Hawking

Low-cost sensors (LCSs) have been the focus of study for the past few years [4, 8, 76, 77]. In fact, a major research topic has been the improvement of data quality by *in-situ* calibration of sensors with machine learning (ML) techniques [12, 14, 15, 78]. The objective of having nodes with good data quality, when deployed together with high-precision government nodes, is to increase the spatial resolution of the monitoring networks to carry out different applications and try to use these data in a regulated way for different actions. Thus, the cornerstone of the practical use of sensor networks formed by LCSs is the quality of their data. So far, the main paradigm has been to improve the quality of each sensor individually, using *in-situ* calibration, and then deploying sensors that have good data quality [11, 12]. Nevertheless, this type of network can present problems in the long-term as sensors tend to suffer from inaccuracies, data loss due to sensor malfunction, and anomalous measurements, among others [12]. Therefore, it is important to be able to monitor and maintain the data quality of a sensor network that has been deployed. In recent years, different works have analyzed data from air quality sensor networks, using geostatistical techniques such as Kriging to create air pollution maps of an area or to create virtual sensors [53, 79, 80]. More recently, with the rise of graph signal processing (GSP), graphs have been used to cluster measurements from ozone sensor networks [21]. Yet, little work focuses on evaluating and maintaining the quality of network data through data analysis techniques. Therefore, in this thesis, we focus on the maintenance of the data quality of a sensor network, in particular, we employ graph-based models, combining the graphs representing the sensor network with GSP techniques and ML techniques.

In this chapter, we show the main techniques for inferring a graph that represents the relationships between the different pollution sensors that form a sensor network. First of all, section 5.1 introduces the problem and the presented approach. Then, section 5.2 shows the main fundamentals of graph signal processing in order to develop the rest of the chapter. Section 5.3 shows the different graph inference techniques used, the experimental setup, and the results. Finally, section 5.5 concludes the chapter. This chapter presents the findings made in “*Graph learning techniques using structured data for IoT air pollution monitoring platforms*“, IEEE IoT-J, [81], and some results of “*Graph Signal Reconstruction Techniques for IoT Air Pollution Monitoring Platforms*“, IEEE IoT-J, [82].

## 5.1. Data quality, sensor networks, and applications

The quality of the data provided by LCSs has been the main concern in recent years [8, 76, 83]. This has been the cornerstone since their main use is focused on increasing the spatial resolution provided by reference stations. In this way, different applications can be carried out using measurements from deployed sensor networks. In addition, these data can be used for future studies to increase public awareness and to be able to introduce measures in a regulated way in the creation of prevention and containment measures. Williams *et al.* [4] discusses the ability of LCSs to produce accurate values. Similarly, Castell *et al.* [53] highlight the LCSs’ lack of accuracy for being used in a regulated way, with necessary in-field sensor evaluation. Thus, data quality is one of the most important challenges in order to be able to include measurements from these deployed sensors in the development of new environmental policies. In fact, Niu *et al.* [84] show the importance of the data quality in a Chinese testbed but remark that data can be used in a complementary way to the data from government instrumentation. Besides, Rai *et al.* [7] mention that LCSs allow increasing the air pollution monitoring spatial resolution, however, they emphasize that the data quality is a major concern. Indeed, chapter 3 already shows the need for sensor recalibration for long-term air pollution monitoring campaigns. All in all, the calibration of LCSs has been the main research task in order to improve the quality of the data provided by these nodes.

### 5.1.1. Improving low-cost sensor data quality: Sensor Calibration

As mentioned in the previous chapters, sensor calibration using ML techniques has gained importance in recent years [8, 12, 14, 15, 85]. In fact, all kinds of techniques have been applied, from linear models such as multiple linear regression [8, 14], to nonlinear models such as random forest [15, 16], support vector regression [86] or artificial neural networks [14]. In addition to increasing data quality by calibrating the sensors, other techniques such as feature selection have also been studied. In this way, it is intended to introduce measurements from different sensors, which may contain cross-sensitivities and cross-correlation, that can benefit and improve the calibration of a specific sensor [87]. Once the LCSs have been individually calibrated at a reference station they are deployed over the area of interest. The deployed nodes can be either static sensors or mobile sensors [88]. The calibration performed, i.e., the models obtained in the calibration can be used throughout the whole air pollution monitoring campaign, with possible data quality problems, and recalibration schemes can be performed to keep the calibration models updated [58].

### 5.1.2. LOW-COST SENSOR NETWORK DEPLOYMENT: USE CASES

Once the nodes with LCSs have been calibrated, they are deployed in the area of interest with different purposes, some common uses are (see chapter 1); *i)* to increase the spatial resolution of the reference stations and provide pollution measurements at specific points of interest where there are no measurements, *ii)* to use the measurements from the LCSs together with the measurements from the reference stations in the area to create pollution maps and to extrapolate the observed values to the whole area of interest, and *iii)* to create virtual sensors in locations where a LCS could not be deployed. In all cases, the quality of the data collected by the LCSs is crucial, in the first case, to obtain accurate measurements of the location where we are interested in measuring concentrations, in the second case, because these data are used in conjunction with those of the stations to create pollution maps, and in the third case, to obtain accurate virtual sensors. Schneider *et al.* [19] used a geostatistical method, universal Kriging, to create a pollution map by mixing information from a dispersion model and $NO_2$ measurements from LCSs in the city of Oslo, Norway. The results showed good predictive ability compared to reference stations. Likewise, Van Zoest *et al.* [89] used a spatio-temporal Kriging to create an air pollution $NO_2$ 25-meter resolution map in the city of Eindhoven, Netherlands. Ahangar *et al.* [90] compared the use of a dispersion model for modeling $PM_{2.5}$ with the use of geostatistical tools applied directly to the observation made by the network, showing better results for the dispersion model. Continuing with the importance of data quality, Crawford et al. [91] showed how LCSs can help to detect and evaluate the evolution of volcanic emissions, thus showing the use of LCSs in the measurement of extreme events or natural disasters. Apart from the use of Kriging techniques, there are also works where the inverse distance weighting (IDW) model is used to model pollution concentrations [92]. In the same way, we can see how pollution maps can be created using spatial extrapolation methods using the measurements of an area of interest [93, 94].

Land use regression (LUR) has also been a tool used in the extrapolation of LCS measurements and the creation of virtual sensors. Nori *et al.* [94] predicted urban exposure $NO_2$ concentrations using LUR. Similarly, Weissert *et al.* [95] combined LUR with LCSs to create a $NO_2$ air pollution map with 50-meter resolution with good results, showing the possibility to detect pollution hotspots with this technique. Coker *et al.* [96] showed the performance of land use regression combined with ML techniques for a particulate matter LCS network deployed in Uganda. Indeed, the authors mentioned the importance to use data quality control measures in the deployment of LCSs in order to improve the data reliability and air pollution exposure estimates in places where governmental reference stations are scarce or nonexistent. Hofman *et al.* [97] used different spatio-temporal techniques for the creation of air pollution maps. More precisely, they used the Air Variational Graph Autoencoder (AVGAE) to perform matrix completion and air pollution extrapolation. They mentioned the need for accurate sensors in order to create maps with good data quality. All these cases where concentrations at a particular location are predicted using different techniques can alternatively be interpreted as the creation of virtual sensors at the predicted locations.

### 5.1.3. Sensor network data quality: Graph-Based Approach

As we have seen, the key to the different applications of LCSs is the reliability and quality of the data [98, 99]. Data quality is important to obtain good measurements at the area of interest where the sensors have been placed and to accurately assess exposure, possible hotspots, or extreme pollution episodes. In fact, most research highlight the dependence between the successful application and the quality of the sensors deployed. However, the state of the art is mostly focused on improving the data quality by *in-situ* calibration techniques, i.e., improving the quality of each sensor individually and deploying them in the expectation of having a good quality during the air pollution measurement campaign.

Therefore, in this chapter, we propose to model deployed heterogeneous LCS networks, with LCSs and reference stations, using *graphs*. Basically, the sensors are deployed together to measure a phenomenon over an area of interest, so the different network nodes can be related to each other, and this relationship can be exploited by using the information from similar nodes. For instance, in the case of having a LCS near or in similar conditions to a reference station, there is a relationship between these two and the information from the station can benefit the information from the LCS. Thus, the signal at a node can be reconstructed, either because the sensor has missings or has produced erroneous measurements, by means of the measurements of similar sensors.

Recently, the field of GSP has emerged, opening the door to the use of signal processing techniques on signals defined on graphs [20, 21]. Thus, these tools are ideal to be applied on pollution sensor networks and to be able to apply signal processing techniques and improve data quality. In fact, Jablonski *et al.* [100] modeled a network of ozone reference stations in Poland using a graph, then they applied spectral clustering to group the stations according to their relations. But in order to apply different techniques, such as clustering or signal reconstruction, one must first infer the different relationships in a network, which is the objective of the study in this chapter. Discrete models, such as Gaussian Markov random fields (GMRF), which can be interpreted as undirected graphs, have also been used in the prediction of PM concentrations [101]. Regarding geospatial models, Song *et al.* [102], compared the use of Gaussian geospatial models and Gaussian Markov random fields for modeling $PM_{2.5}$ concentrations.

In short, GSP and ML tools offer a good combination to model and control the quality of data from a sensor network. The first step in using these techniques is to find the graph that models the measurements and relationships between the different sensors in the network. During the last few years, the creation of graphs has gained importance for the representation and analysis of sensor networks, such as heterogeneous networks for air quality monitoring. The most classical method for the creation of such relationships is based on the assumption that closer sensors will be more strongly related due to a spatial correlation of the measured phenomenon [21, 103]. Therefore, the relationship between sensors can be defined as a decreasing exponential function dependent on the geographical distance between the nodes. More recently, graph inference techniques based on measured network data have been developed [104–106]. The most commonly used criterion has been the smoothness of the graph with respect to the measured signals. Similarly, GMRF inference has also been used to infer graphs, e.g., Friedman *et al.* [107] proposed the Graphical Lasso to infer the precision matrix of the data assuming

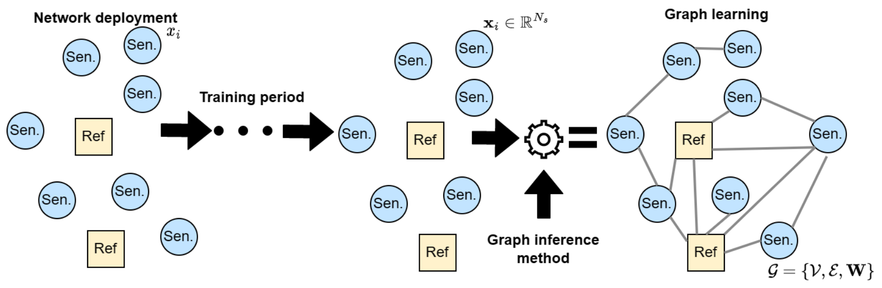

Network deployment
$x_i$
Sen.
Ref
Training period
$\mathbf{x}_i \in \mathbb{R}^{N_s}$
Graph inference method
Graph learning
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$

**Figure 5.1:** Heterogeneous sensor network deployment and posterior graph inference phase to obtain the graph topology $\mathcal{G}$.

that these form a GMRF. Actually, Dong *et al.* [24] classified graph learning models into; *i)* statistical models based on Markov random fields (MRF), *ii)* models based on GSP, and *iii)* models derived from physical processes (e.g., diffusion).

In this chapter, we compare the ability of graphs to model heterogeneous sensor networks, comparing different graph learning techniques on heterogeneous network data. In order to compare the different graphs' performance, we use the main application, signal reconstruction, where the signal from different nodes of the network is reconstructed using their neighboring nodes, defined by the connections of the graph. This task is really important in maintaining the network data quality since it allows for imputing missing values, improving sensor estimates, detecting drifts, creating virtual sensors, etc. The following section deals with the fundamental aspects of GSP, which necessary to understand the remaining sections. Figure 5.1 depicts the graph-based approach to describe heterogeneous air pollution sensor networks.

## 5.2. FUNDAMENTALS OF GRAPH SIGNAL PROCESSING

In this section, we introduce the GSP framework, restricting ourselves to the knowledge necessary to be able to develop and understand the techniques of this thesis. From the definition of a graph to the definition of graph signal smoothness. The complementary knowledge for the other chapters will be explained as the concepts appear. For more information on GSP processing in general refer to [20, 21, 23, 103].

First, we define a graph $\mathcal{G}$ as the triplet $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of nodes and its cardinality $|\mathcal{V}| = N$ is the number of nodes in the network. The weight matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ defines the edges' weights. The edge set $\mathcal{E}$ is defined as $\mathcal{E} = \{e_{ij} : i, j \in \mathcal{V} \wedge \mathbf{W}_{ij} \neq 0\}$, so an edge $e_{ij} \in \mathcal{E}$ denotes a connection between node $x_i$ and node $x_j$. Depending on the characteristics of the network and the graph, the $\mathbf{W}$ matrix may have different characteristics. Roughly speaking, we can classify graphs into directed and undirected. In the undirected case, $\mathbf{W}_{ij} = \mathbf{W}_{ji}$, so that the matrix $\mathbf{W}$ is symmetric, $\mathbf{W}^{\mathsf{T}} = \mathbf{W}$. Nonetheless, in the case where the graph is directed, i.e., $\exists (i, j) : \mathbf{W}_{ij} \neq \mathbf{W}_{ji}$, the $\mathbf{W}$ matrix is not symmetric. There are more general cases, such as the case of multigraphs, where more than one edge is allowed between two nodes, being $\mathcal{E}$ a multiset.

The weight matrix $\mathbf{W}$ is not the only matrix that can be used to describe a graph.

**Table 5.1:** Some useful properties of Laplacian matrices for undirected graphs. PSD denotes a positive semi-definite matrix.

| **Laplacian L** | **Normalized Laplacian $\mathbf{L}_N$** |
|---|---|
| Symmetric PSD | Symmetric PSD |
| $\lambda_i \geq 0$ | $\lambda_i \geq 0$ |
| $\mathbf{L1} = \mathbf{0}$ | $\mathbf{L1} = \mathbf{0}$ |
| $\#(\lambda_i = 0) \equiv$#components | $\#(\lambda_i = 0) \equiv$#components |
| | $0 = \lambda_1 \leq \cdots \leq \lambda_N \leq 2$ |
| $tr(\mathbf{L}_N) = \sum_i \lambda_i = \sum_i \mathbf{D}_{ii}$ | $tr(\mathbf{L}_N) = \sum_i \lambda_i = N$ |

The adjacency matrix $\mathbf{A} \in \mathbb{B}^{N\times N}$, which simply indicates the existence of a relationship between two nodes, can also be used to describe a graph $\mathscr{G}$. A weight matrix can be converted to an adjacency matrix by setting:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } \mathbf{W}_{ij} \neq 0 \\ 0 & \text{if } \mathbf{W}_{ij} = 0 \end{cases} \tag{5.1}$$

In addition, there is also the notion of *Laplacian* matrix $\mathbf{L} \in \mathbb{R}^{N\times N}$ that can be interpreted as the negative discrete Laplace operator in matrix form and has been intensively studied in graph theory and spectral graph theory [108]. Now, the Laplacian matrix $\mathbf{L}$ and its normalized version $\mathbf{L}_N$ can be computed from the weight matrix $\mathbf{W}$ as:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{5.2}$$

$$\mathbf{L}_N = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} \tag{5.3}$$

Where $\mathbf{D}$ is the degree matrix, which is a diagonal matrix whose diagonal is defined as $D_{ii} = \sum_j W_{ij}$. As mentioned, both the Laplacian $\mathbf{L}$ and the normalized Laplacian $\mathbf{L}_N$ are symmetric and positive semidefinite matrices, meaning that all their eigenvalues are nonnegative ($\lambda_i \geq 0$). Besides, the normalized Laplacian has a trace equal to the number of nodes of the graph, $tr(\mathbf{L}_N) = N$. Another interesting property is that the multiplicity of the Laplacian $\mathbf{L}$ eigenvalue equal to zero ($\lambda_i = 0$) is equal to the number of connected components of the graph and the eigenvalue spectrum for the normalized Laplacian matrix is $0 = \lambda_1 \leq \cdots \leq \lambda_N \leq 2$. Summarizing, Table 5.1 denotes some basic properties of these Laplacian matrices. There exist other Laplacian matrices (e.g., random walk Laplacian) that we do not cover in this thesis.

Before continuing it is important to define the notion of a signal defined over a graph, it is defined as the map $x : \mathscr{V} \to \mathbb{R}$, that maps a vertex index to a real value [21, 103]. Thereby, $x_i \in \mathbb{R}$ is regarded as the value of the graph signal at the $i$-th node. Another basic concept is that of *graph shift*, this operator consists of the translation of a signal shift in the classic signal processing [21]. In this case, the graph shift is defined as the multiplication of the graph shift operator (GSO) $\mathbf{S} \in \mathbb{R}^{N\times N}$ by a signal defined on the graph $\mathbf{x} \in \mathbb{R}^N$:

$$\mathbf{x}^{[1]} = \mathbf{S}\mathbf{x} \tag{5.4}$$

The resulting shifted signal $\mathbf{x}^{[1]} \in \mathbb{R}^N$ can be seen as the result of a diffusion process where the value at the i-th vertex is defined by a linear combination of the value at that vertex and the signal at the neighboring sensors. The *neighborhood* of a node can be defined as the set of nodes that are connected to that node via an edge, $\mathcal{N}(x_i) = \{j : j \in \mathcal{V} \wedge W_{ij} \neq 0\}$. The result of the signal shift will depend on the definition of the GSO, in fact, in the field of GSP the adjacency **A**, the weighting matrices **W**, and Laplacian matrices **L** have been used as GSO [21, 22]. For instance, in the case of having a linear graph representing a discrete-time signal, where the adjacency matrix is defined as $\mathbf{A}_{t,t+1} = 1$, the graph shift is equivalent to the shifting of the discrete signal in the classic signal processing framework.

Once we have the notions of signal **x** defined over a graph and the notion of graph shift, we can move on to the definition of *signal smoothness*. The Laplacian quadratic form, also known as *total variation* (TV) or Dirichlet energy, is a measure of graph signal smoothness with respect to the underlying graph topology defined by its Laplacian **L**:

$$\mathrm{TV}(\mathbf{x}, \mathbf{L}) = \mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum_{i,j} W_{ij}(x_i - x_j)^2 \tag{5.5}$$

This quantity is a measure of signal smoothness, where the lower the more smooth the signal is. This notion of smoothness is further used when defining the *graph discrete Fourier transform* (GDFT), as well as an objective function for graph learning tasks, signal reconstruction, and other applications. Basically, it is the weighted summation of squared differences between the values of connected nodes, the more similar the values $x_i$ and $x_j$ the lower the Laplacian quadratic form. Signal smoothness can be used as an indicator of how well a signal fits a graph and vice versa, how well the graph fits a signal.

Now, the concept of GSO and signal smoothness allows for defining the GDFT. In fact, the basis of the GDFT are obtained from the eigendecomposition of the graph shift operator. From now on, we assume **S** = **L**, so the eigendecomposition of **L**, since it is a PSD symmetric matrix, is defined as:

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} \tag{5.6}$$

where $\mathbf{U} \in \mathbb{R}^{N\times N}$ is the matrix of eigenvectors, whose columns are the corresponding eigenvectors $\mathbf{u}_i \in \mathbb{R}^N$, and $\mathbf{\Lambda} \in \mathbb{R}^{N\times N}$ is a diagonal matrix containing the eigenvalues $\lambda_i$ associated to the eigenvectors $\mathbf{u}_i$. Now, the eigenvectors $\mathbf{u}_i$ are defined as the GDFT basis and their corresponding eigenvalues $\lambda_i$ correspond to the frequencies. Indeed, the frequencies $\lambda_i$ are the smoothness of the corresponding eigenvectors:

$$\lambda_i = \mathbf{u}_i^\top \mathbf{L}\mathbf{u}_i, \quad i = 1, \ldots, N \tag{5.7}$$

Then, we can sort the eigenvalues in increasing order $0 \leq \lambda_1 \leq \cdots \leq \lambda_N$, where larger eigenvalues indicate larger frequencies. In fact, we can see how the eigenvector $\mathbf{u}_1$ associated to the constant frequency ($\lambda_1 = 0$) is a constant signal since its $\mathrm{TV}(\mathbf{u}_1, \mathbf{L}) = 0$. Moreover, high frequencies (large eigenvalues) will have a higher quadratic form of the eigenvector, indicating that high frequencies correspond to high-frequency GDFT basis where the values between neighboring nodes are more disparate. The graph discrete Fourier transform of a graph signal **x** and its *inverse graph discrete Fourier transform* (IGDFT) are defined as [21, 103, 109]:

$$\mathbf{X} = \mathrm{GDFT}\{\mathbf{x}\} = \mathbf{U}^{-1}\mathbf{x} \tag{5.8}$$

$$\mathbf{x} = \mathrm{IGDFT}\{\mathbf{X}\} = \mathbf{U}\mathbf{X} \tag{5.9}$$

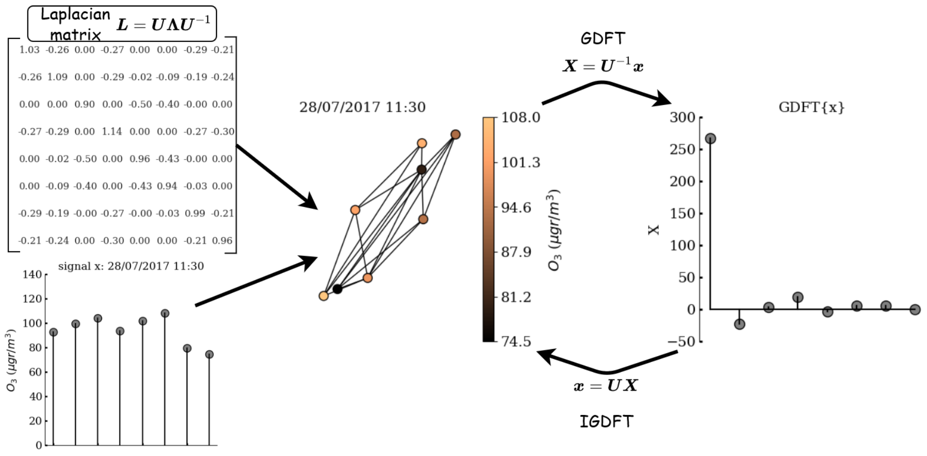


**Figure 5.2:** Example of the GDFT and IGDFT applied to a graph signal defined over a sensor network via a Laplacian matrix **L**.

Given that the eigenvector matrix is an orthonormal matrix, then $\mathbf{U}^{-1} = \mathbf{U}^\top$. Thus, $\mathbf{X} \in \mathbb{R}^N$ corresponds to the GDFT of the signal $\mathbf{x}$ defined on the graph, where the first components correspond to the Fourier coefficients of the low frequencies and the last components of the transformed signal $\mathbf{X}$ correspond to the Fourier coefficients of the high frequencies. As an example, Figure 5.2 shows the GDFT and IGDFT process for a signal defined on a graph representing a sensor network. The example shows the GDFT coefficients of a signal whose low-frequency coefficients have a higher magnitude than the high-frequency coefficients, which means that the signal has no abrupt changes between the values of neighboring nodes. Finally, the *signal filtering* operation can also be translated into graph setting [21, 109]. In a very naïve way, one could define a low-pass filter by calculating the GDFT of a certain graph signal $\mathbf{x}$ and setting the coefficients of the high frequencies to zero from a certain cut-off frequency $\lambda_c$:

$$\mathbf{Y}_k = \begin{cases} \mathbf{X}_k & \lambda_k \le \lambda_c \\ 0 & \lambda_l > \lambda_c \end{cases} \tag{5.10}$$

In a more formal way, the filtering of a signal on a graph in the spectral domain can be defined as $\mathbf{Y} = \mathbf{G}(\boldsymbol{\Lambda})\mathbf{X}$, where $\mathbf{G}(\cdot)$ is the graph transfer function defined as a function of the eigenvalues $\boldsymbol{\Lambda}$ of the graph. This graph transfer function can be expressed as a linear combination of the powers of $\boldsymbol{\Lambda}$. Conversely, applying the IGDFT, a filter can be defined in the vertex domain as the linear combination of the powers of the shift matrix $\mathbf{S}$ (in this case the Laplacian $\mathbf{L}$):

$$\mathbf{Y} = \mathbf{G}(\boldsymbol{\Lambda})\mathbf{X} = \sum_{i=0}^{N} h_i \boldsymbol{\Lambda}^i \mathbf{X} \qquad \text{[FrequencyDomain]} \tag{5.11}$$

$$\mathbf{y} = \mathbf{f}(\mathbf{L})\mathbf{x} = \sum_{i=0}^{N} h_i \mathbf{L}^i \mathbf{x} \qquad \text{[VertexDomain]} \tag{5.12}$$

Where $h_i \in \mathbb{R}$ are the taps of the filter. Depending on the application, one transfer function or another can be designed. There are more techniques such as the z-transform or signal convolution, but since they are not necessary for the development of this chapter we skip them. For more information on the mentioned techniques and other techniques, refer to literature surveys [21, 103, 109].

## 5.3. SENSOR NETWORK GRAPH INFERENCE

In this section, we define the main problem addressed in this chapter, the sensor network graph inference. This problem has been of special interest during the last decade, with a multitude of methods with different assumptions to obtain the topology of the graph in which the signals are defined [23, 24].

**Definition 10** ***Graph learning task**: the graph learning or network topology inference task can be defined as the inference of the graph $\mathcal{G}$ representing the network. This can be achieved by obtaining a GSO, either the adjacency matrix $\mathbf{A}$, the weight matrix $\mathbf{W}$ or the Laplacian matrix $\mathbf{L}$, which defines the edges $\mathcal{E}$ of the graph and their weights.*

There are many techniques, some based on prior information, such as the geographical information (e.g., the distance between the i-th and j-th nodes $d_{ij}$) of the different nodes of the network. Other techniques are based on obtaining a graph that adapts to the signals already collected by the network. In short, there are many techniques, and here we compare three of the best known to see their effectiveness for air pollution sensor networks. First, we review statistical techniques such as graphical Lasso, where the data collected by the network are assumed to come from a GMRF whose precision matrix describes the relationships between nodes. And secondly, we study GSP techniques where the Laplacian (used as GSO) is obtained from the geographical information of the nodes or from a training set following a smoothness criterion.

### 5.3.1. MOTIVATION

Sensor networks for measuring air pollution are complex. In this thesis, we approach the data quality of a sensor network from a graphical model and GSP perspective. This allows us to model a network in a discrete way, the opposite case of the creation of maps that try to infer a continuous field representing the pollution in space, where we focus on each of the network nodes and their data quality. In fact, the main challenges of this type of sensor network are:

(a) The existence of heterogeneous networks where two types of sensors coexist, LCSs and high-precision government instrumentation. Both nodes, with different types of data quality,

(b) These type of network measure pollutants that may have spatial variability or may contain more complex relationships defined by anthropogenic emissions and location conditions,

(c) Similarly, spatial correlation may not hold in the case of heterogeneous networks where LCSs may be more similar to other sensors depending on their deployment conditions regardless of the geographical distance.

Thus, the use of graphs $\mathcal{G}$ to describe sensor networks have the following advantages:

(I) Explicitly describes the relationship between different sensors in the network. That is to say, a kind of feature selection for each sensor to define the neighboring sensors of each node. Graphs are known to ease the interpretability of ML models [110],

(II) Takes advantage of the information contained in the neighborhood (network data) to correct data in the different sensors by means of GSP and ML techniques,

(III) The application of GSP and ML tools on this type of network, e.g., spectral clustering, graph signal filtering, outlier detection, etc.

### 5.3.2. APPROACHES

In this section, we describe the two main approaches compared for the inference of the graph $\mathcal{G}$ representing the sensor network. First, we evaluate statistical techniques such as *graphical Lasso* [107]. Second, we study the use of more recent GSP techniques, where we obtain the topology of the network $\mathcal{G}$ either by using prior information such as the *distance* between nodes, or by obtaining a *smooth* graph with respect to the data collected by the network. Since the relationship between the measurements of two sensors of a pollution network is mutual, i.e., if sensor $x_i$ is related to sensor $x_j$ then it makes sense that sensor $x_j$ has the same relationship with sensor $x_i$, we opted for the use of *undirected* graphs.

#### STATISTICS

In the case of statistical graphical models such as GMRF, the objective is to find the joint distribution of the network variables as a zero-mean Gaussian defined by the precision matrix $\mathbf{\Theta} \in \mathbb{R}^{N\times N}$ (inverse of the covariance matrix $\mathbf{\Sigma}$). This precision matrix is found to guarantee conditional independence between the variables. Meinshausen *et al.* [111] proposed a neighborhood selection algorithm to find $\mathbf{\Theta}$. This precision matrix encodes the graph topology $\mathcal{G}$ since the conditional independencies work as a neighborhood (or feature) selector finding similar nodes in the network that need to be connected. Friedman *et al.* [107] developed the graphical Lasso, which is the most remarkable model since it forces the sparsity of the precision matrix to reduce the number of edges of the graph and is the model used as statistical-based.

The graphical Lasso assumes that the underlying network variables $x_i$ follow a GMRF, so every node $x_i$ can be seen as a random variable over the graph $\mathcal{G}$, $\{x_i : i \in \mathcal{V}\}$ that satisfies the pairwise Markov property and the joint-distribution of vertex-indexed variables follows a zero-mean Gaussian distribution with precision matrix $\mathbf{\Theta}$.

**Definition 11** ***The pairwise Markov property*** *states that two vertices that are not connected in the graph ($e_{ij} \notin \mathcal{E}$ or $W_{ij} = 0$) are conditionally independent given all other vertices in the graph, $x_i \perp\!\!\!\perp x_j | \mathcal{V} \setminus \{i, j\}$. Similarly, the precision matrix $\mathbf{\Theta}$ encodes the conditional independencies, where $\Theta_{ij} = 0$ implies that variables $x_i$ and $x_j$ are conditionally independent given the rest of variables.*

The main focus of statistical topology inference is on the inference of the precision matrix since its entries have direct correspondence with the conditional independencies and partial correlations of each variable. In fact, the adjacency matrix of the graph **A** can be obtained using conditional independencies encoded by the precision matrix as $\{A_{ij} = 1 : \Theta_{ij} \neq 0\}$. The first attempt at estimating the entries of the precision matrix dates back to Dempster *et al.* [112] who developed the covariance selection. Since this solution can present problems with large graphs, Friedman *et al.* [107] developed the graphical Lasso which learns a sparse precision matrix. The graphical Lasso solves an $l_1$-regularized maximum likelihood problem, where the $l_1$-norm of the precision matrix is used as the convex relaxation of the $l_0$-norm, which minimizes the number of nonzero entries. Thus, forcing sparsity results in graphs with smaller and more selective neighborhoods.

$$\arg\max_{\Theta} \quad \underbrace{\log\det(\Theta) - \mathrm{tr}(\hat{\Sigma}\Theta)}_{\text{log-likelihood GMRF}} - \underbrace{\lambda \|\Theta\|_1}_{\text{sparsity}} \tag{5.13}$$

The optimization problem formulated above is the convex problem solved by the graphical Lasso algorithm. The first term corresponds to the log-likelihood of the GMRF, where $\hat{\boldsymbol{\Sigma}}$ is the empirical covariance matrix, while the second term is a sparsity-promoting term of the precision matrix weighted by the hyperparameter $\lambda \in \mathbb{R}$. The $l_1$-norm of the matrix $\|\cdot\|_1$ is the convex relaxation of the $l_0$-norm, which minimized the number of entries different than zero. $\lambda$ controls the penalizes the sparsity of the precision matrix, larger $\lambda$ values will incur in more sparse precision matrices. It is important to note that the matrix $\boldsymbol{\Theta}$, which contains the partial correlations $\rho$, cannot be used directly as the weight matrix **W** of the graph as it may contain negative values, leading to an invalid weight matrix. However, as mentioned above, the precision matrix $\boldsymbol{\Theta}$ can be used to define the adjacency matrix **A** of the network.

### GRAPH SIGNAL PROCESSING

The other alternative recently defined in the field of GSP [23, 24] consists of defining a valid Laplacian matrix either from its definition ($\mathbf{L} = \mathbf{D} - \mathbf{W}$) or by solving an optimization problem with the objective of finding a Laplacian that satisfies certain conditions. Thus, in this thesis we explore two cases; *i)* the use of prior information such as the graph distance between nodes and *ii)* the learning of the graph from the data by optimization.

#### ***Distance-based graph***

This is the most common type of network used in most applications [20, 103, 109]. In fact, it is based on the assumption that nodes that are close in space tend to be related (e.g., correlated). In a similar way, Kriging techniques and inverse distance weighting (IDW) use the distance between the different nodes of the network and the points at which concentrations are predicted. This type of graph has been used to represent temperature networks, pollution networks, and other phenomena due to their simplicity and lack of data requirements [100, 113]. Likewise, this type of graph has been very popular in fields such as semi-supervised learning and manifold learning [108].

To explicitly create the weight matrix **W** of the network, a similarity function $\delta : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$ between two nodes of the network is necessary to obtain the different weights.

There are different ways to create the nodes' relationships using the geographical information but the most important ones use as similarity function a decaying exponential function as a function of the distance between two nodes $d_{ij}$. Then, in order to explicitly manipulate the density of the network, a threshold $TH$ can be applied to eliminate all the edges of the network whose distance between nodes is greater than $TH$. This serves as thresholding to eliminate the effect of distant nodes with very small weights. Accordingly, we can define the weight matrix **W** as:

$$W_{ij} = \begin{cases} e^{-\frac{d_{ij}}{2\tau}} & if \ \ d_{ij} \leq TH \\ 0 & if \ \ d_{ij} > TH \end{cases} \tag{5.14}$$

Where the decaying exponential function, $\delta(i,j) = e^{-\frac{d_{ij}}{2\tau}}$, is the radial basis function (RBF) defined by the distances, $\tau \in \mathbb{R}$ is the scale of the RBF kernel, and $d_{ij}$ is the Haversine distance between two nodes. It is quite straightforward to see that $d_{ij} = d_{ji}$, then the graph obtained is symmetric. Next, obtaining the combinatorial symmetric Laplacian is straightforward from its definition. In a similar way, in other distance-based options, instead of using a threshold to reduce the graph density, a nearest-neighbor approach is used to define the connectivity of the graph [103].

***Smoothness-based graph***

Lately, there has been a growing interest in the inference of the weight matrix or Laplacian from a set of graph signals [23, 24, 105]. The framework of graph learning within GSP consists of inferring a valid shift matrix **S** (that meets the requirements to be a shift matrix) that conforms to the relationships present in the observed graph signals $\mathbf{X} \in \mathbb{R}^{N \times N_s}$, where $N$ is the number of nodes and $N_s$ is the number of graph signals, and has certain sparsity characteristics. Thus, we can define a general framework to learn the shift matrix **S** from graph signals **X** such as:

$$\min_{\mathbf{S} \in \mathscr{S}} \ \mathrm{F}(\mathbf{S},\mathbf{X}) + \lambda \mathrm{R}(\mathbf{S}) \tag{5.15}$$

Where $\mathrm{F}(\cdot,\cdot)$ is a criterion that evaluates a learned shift matrix **S** with respect to the observed graph signals **X**, $\mathrm{R}(\cdot)$ is a regularization function that penalizes the number of the edges of the resulting shift matrix **S**, $\lambda \in \mathbb{R}$ is a hyperparameter that controls the regularization term, and $\mathscr{S}$ is the set of valid shift matrices. A shift matrix **S** is said to be valid if it fulfills its corresponding properties (see Table 5.1). Although different $\mathrm{F}(\cdot,\cdot)$ criteria can be used, GSP is built under the assumption that signals over a graph tend to be smooth relative to the underlying graph. Thus, in the case of learning the Laplacian **L**, the total variation in matrix form $\mathrm{F}(\mathbf{L},\mathbf{X}) = \mathrm{TV}(\mathbf{X},\mathbf{L}) = tr(\mathbf{X}^\top \mathbf{L}\mathbf{X})$[1] is a good criterion to evaluate the fit of the graph on the observed signals. Thus, a smooth graph with respect to the signals will imply that two nodes $x_i$ and $x_j$ connected by a significant weight ($\mathbf{L}_{ij}$ or $\mathbf{W}_{ij}$) will tend to have similar values. As well, it is possible to obtain a graph that makes the signals low-pass over the graph since smooth signals imply that the energy of the signal is concentrated in the lowest GDFT components given that

[1] $tr(\mathbf{X}^\top \mathbf{L}\mathbf{X})$ corresponds to the matrix-form of the total variation.

$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \mathbf{x}^\top \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}\mathbf{x} = GDFT\{\mathbf{x}\}^\top \boldsymbol{\Lambda} GDFT\{\mathbf{x}\}$ and $\boldsymbol{\Lambda} = diag(\lambda_1, \ldots, \lambda_N)$ where the eigenvalues are sorted in increasing order $0 \le \lambda_1 \le \cdots \le \lambda_N$. Thus, minimizing the total variation penalizes signals with more energy in the highest-frequency components. Moreover, the regularization term R(·) is also important to control the topology of the learned graph and its complexity. Thus, this term is used to penalize more complex or dense graphs using criteria such as the Frobenius norm of the shift matrix $\|\mathbf{S}\|_\mathsf{F}$ or the $l_{1,1}$ matrix norm $\|\mathbf{S}\|_1$, where the former penalizes the magnitude of the entries of the shift matrix and the latter is used as a convex relaxation of the $\mathrm{l}_0$-norm to penalize the number of entries different from zero, so penalizing the sparsity of the matrix.

There are different works that focus on graph inference from a set of graph signals using smoothness as a criterion and penalizing the complexity of the graph [23, 104, 105]. Egilmez *et al.* [106] even proposed to learn a generalized precision matrix $\boldsymbol{\Theta}$ for a GMRF by restricting it to be a valid Laplacian. Among the most prominent works, we focus on the work of Dong *et al.* [104] where a valid Laplacian $\mathbf{L}$ is directly inferred using a latent factor analysis approach. The proposed optimization problem is a not jointly convex problem that can be solved iteratively:

$$\begin{aligned} \min_{\mathbf{L},\mathbf{Y}} \quad & \underbrace{\|\mathbf{X}-\mathbf{Y}\|_\mathsf{F}^2}_{data\ \ fidelity} + \underbrace{\underbrace{\alpha\, tr(\mathbf{Y}^\top \mathbf{L}\mathbf{Y})}_{smoothness} + \beta\|\mathbf{L}\|_\mathsf{F}^2}_{sparsity} \\ \text{s.t.} \quad & tr(\mathbf{L}) = N, \\ & L_{ij} = L_{ji} \le 0, \quad i \ne j, \\ & \mathbf{L}\cdot\mathbf{1} = \mathbf{0}. \end{aligned} \tag{5.16}$$

Where $\mathbf{X} \in \mathbb{R}^{N\times N_s}$ are the training graph signals and $\mathbf{Y} \in \mathbb{R}^{N\times N_s}$ are their filtered version. Thus, the first term of the objective function penalized the data fidelity, i.e., the filtered version of the signals $\mathbf{Y}$ are as similar as possible to the original graph signals $\mathbf{X}$. Then, the second and the third term jointly penalize the complexity of the obtained Laplacian $\mathbf{L}$, first penalizing the smoothness $tr(\mathbf{Y}^\top \mathbf{L}\mathbf{Y})$ of the Laplacian with respect to the signals and then penalizing the Frobenius norm $\|\mathbf{L}\|_\mathsf{F}^2$ of the Laplacian. $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$ are the hyperparameters controlling the sparsity of the Laplacian matrix. As for the constraints on the problem, the first constraint ($tr(\mathbf{L}) = N$) prevents obtaining the trivial solution and forces the sum of the diagonal to be $N$, $\sum_i L_{ii} = N$, as in a normalized Laplacian. The second constraint ($L_{ij} = L_{ji} \le 0$) forces the matrix to be symmetric and with non-positive values in the off-diagonal elements. And finally, the last constraint ($\mathbf{L}\cdot\mathbf{1} = \mathbf{0}$) forces the sum of the rows to be zeros so that they are normalized. All these restrictions make the Laplacian a positive semidefinite (PSD) matrix and a valid Laplacian.

Given the different methods explained in this section, the next section describes how to use these methods with air pollution sensor network data, as well as the methodology used to evaluate the different graph learning techniques.

### 5.3.3. GRAPH SENSING: GRAPH LEARNING & SIGNAL RECONSTRUCTION

Once we have seen some of the most common techniques for inferring the graphs of a sensor network, we explain how to exploit these graph topologies in air pollution sensor networks and the methodology undertaken to compare the different models. First of all, it should be taken into account that these graphs can be used to maintain the quality of the data from this type of sensor (see section 5.1), i.e., all the existing relationships in a network can be used to flow information from one sensor to another and thus benefit each sensor in the network. Thus, in order to maintain the quality of the data from the different sensors that make up the network, we can distinguish two different tasks:

(A) **Graph learning**: infer the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W} \text{ or } \mathbf{L}\}$ that represents the network in order to obtain the implicit relationships between the different sensors of the monitoring network $x_i$, $i \in \mathcal{V}$. For instance, we are interested in knowing which sensors are related to a low-cost tropospheric ozone sensor $x_i$ that has been deployed in an area, i.e., we are interested in finding out if there are other sensors that follow a similar trend to this one and even the existence of a reference instrument related to this one. It is to say, the graph describes the set of sensors related to every sensor via the neighborhood $\mathcal{N}(x_i), i \in \mathcal{V}$ and the weights of these relationships via $\mathbf{W}_{i\mathcal{N}(x_i)}$ or $\mathbf{L}_{i\mathcal{N}(x_i)}$.

In this way, once a sensor network is deployed for an air pollution monitoring campaign, it is possible to assess the graph that represents the network during the first weeks since it is assumed that the sensors tend to be non-problematic during the first weeks of deployment.

(B) **Signal reconstruction**: the topology inferred for the graph ($\mathbf{W}$ or $\mathbf{L}$) serves as a feature selector to select the neighborhoods $\mathcal{N}(x_i)$ of the different sensors, i.e., similar sensors. Therefore, these relationships can be used to reconstruct the signal $x_i \in \mathbb{R}$ in the $i$-th node using the signals of the nodes in its neighborhood $\mathbf{x}_{\mathcal{N}(x_i)} \in \mathbb{R}^{|\mathcal{N}(x_i)|}$ since they are related. A clear example is the case where a reference station or a LCS has not reported data for a certain hour, then this value can be imputed using the signals from the nodes similar to the conflicting sensor, which are potentially non-problematic. There are different ways to perform this signal reconstruction, but in this chapter, we explore two precise yet interpretable techniques to perform the reconstruction. In this way, we can define the objective of reconstructing the signal $x_i$ at a certain time step $t$ using the function $f : \mathbb{R}^{|\mathcal{N}(x_i)|} \to \mathbb{R}$ as:

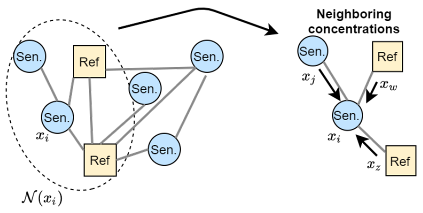


**Figure 5.3:** Neighborhood information can be used to reconstruct sensor signals.

$$\hat{x}_i = f(\mathbf{x}_{\mathcal{N}(x_i)}), \quad \forall i \in \mathcal{V} \tag{5.17}$$

Then, a simple but at the same time very flexible and effective estimator is to assume that the signal can be modeled as a weighted average of the neighborhood signals:

$$\hat{x}_i = f(\mathbf{x}_{\mathcal{N}(x_i)}) \approx \sum_{j \in \mathcal{N}(x_i)} w_{ij} x_j, \quad \forall i \in \mathcal{V} \tag{5.18}$$

In this chapter, we explore two possible approaches for the estimator $f(\mathbf{x}_{\mathcal{N}(x_i)})$; supervised learning and graph-based semi-supervised learning.

B.1) *Supervised machine learning approach*: Since the learned Laplacian matrix $\mathbf{L}$ encodes the resulting adjacency matrix $\mathbf{A}$, these adjacencies $A_{ij}$ can be used as feature selector, where neighbors $j \in \mathcal{N}(x_i)$ of a $i$-th node are used as co-variates to regress the value of the target node $x_i$. In this way, one can define a ML model that predicts the value of the node $x_i$ using the neighboring signals $\mathbf{x}_{\mathcal{N}(x_i)}$. This implies, that in the case where only the target node $x_i$ needs to be reconstructed, $N$ different models are needed (one per network node), but in the case where any node can fail, $2^{|\mathcal{N}(x_i)|} - 1$ different models per network node are needed. Thus, assuming the estimator to be a weighted average makes MLR the most feasible model for this scenario, where the function $f : \mathbb{R}^{|\mathcal{N}(x_i)|} \to \mathbb{R}$ to be estimated has the form:

$$\hat{x}_i = f(\mathbf{x}_{\mathcal{N}(x_i)}) \approx \beta_0 + \sum_{j \in \mathcal{N}(x_i)} \beta_{ij} x_j, \quad \forall i \in \mathcal{V} \tag{5.19}$$

Where the vector of coefficients $\boldsymbol{\beta}$ are the coefficients that weight the sum of the signals $x_j$.

B.2) *Semi-supervised learning approach*: an alternative previously studied in the field of graph semi-supervised learning is the Laplacian interpolation, defined by Belkin *et al.* [108], which is also a graph signal reconstruction model. This is a transductive method, which estimates the value at the unobserved nodes $\{x_j : j \in \mathcal{U}\}$, where $\mathcal{U}$ is the set of unobserved/to be reconstructed nodes (there can be more than one) from the observed nodes $\{x_j : j \in \mathcal{M}\}$, where $\mathcal{M}$ is the set of observed nodes. The objective function to be minimized is the total variation:

$$\begin{aligned} \min_{\hat{\mathbf{x}} \in \mathbb{R}^N} \quad & \hat{\mathbf{x}}^\top \mathbf{L} \hat{\mathbf{x}} \\ s.t. \quad & \hat{x}_i = x_i, \quad \forall i \in \mathcal{M} \end{aligned} \tag{5.20}$$

Where the cost function minimized is the total variation $\mathrm{TV}(\hat{\mathbf{x}}, \mathbf{L})$, and $\hat{\mathbf{x}}_{\mathcal{U}}$ are the values of the missing nodes. The problem is restricted to obtaining graph signals where the observed measurements have the same value (constraint). Solving this optimization problem we can observe that the estimator for $\mathbf{x}_{\mathcal{U}}$ is obtained as:

$$\hat{\mathbf{x}}_{\mathcal{U}} = -\mathbf{L}_{\mathcal{U}\mathcal{U}}^{-1} \mathbf{L}_{\mathcal{U}\mathcal{M}} \mathbf{x}_{\mathcal{M}} \tag{5.21}$$

This results in a linear estimator where the weights of the linear combination are: $\boldsymbol{\beta} = -\mathbf{L}_{\mathcal{U}\mathcal{U}}^{-1} \mathbf{L}_{\mathcal{U}\mathcal{M}}$. This formulation is very useful since it is convex, has an analytical solution, is flexible (it allows more than one missing for a timestep), and uses as the objective function the criterion with which the Laplacian has been learned (in the case of the smoothness-based Laplacian)[2].

**Definition 12** *A* ***transductive method*** *is a method that, given specific training data or observations, infers a function for specific test cases. An example is graph signal reconstruction models, where the unobserved measures are estimated from the observed ones.*

## Methodology

Here we explain the methodology used to implement the different graph learning (graphical Lasso and graph signal processing based) and signal reconstruction (supervised and semi-supervised) models for air quality monitoring network data. To do so, we highlight the particularities of air pollution sensor network data and how they are solved. Figure 5.4 shows the proposed methodology, where it can be seen the different options available for graph learning as well as the required input data and hyperparameters for every model, and the inputs required for the two signal reconstruction approaches.

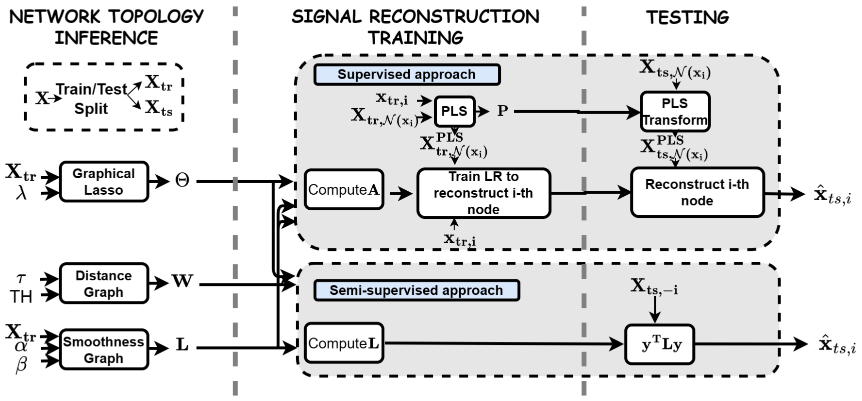


**Figure 5.4:** Graph learning and signal reconstruction methodology for the different models applied to air pollution sensor networks. Both the supervised and semi-supervised approaches are described.

### Graph Inference

In an air pollution monitoring campaign using heterogeneous networks of LCSs, an *in-situ* calibration of the LCSs is first performed at stations close to their deployment site [6, 8]. Then, in the network deployment phase, the nodes mounting the LCSs are placed at the respective points of interest and measurement of the network. At this point, just during the first weeks of deployment, the graph describing the implicit relationships between the different sensors can be learned, since it is assumed that their functioning is correct for at least this small period after the beginning of the deployment. Once this is done, the learned graph can be used for signal reconstruction purposes to maintain the data quality as at the initial deployment.

[2]A similar method minimizing the graph total variation has been proposed from the graph signal processing perspective, where instead of using the Laplacian matrix **L** any graph shift operator **S** can be used [113].

In this case, where all the data from the air pollution monitoring campaign is available, the first step is to split the data **X** into training and testing, $\mathbf{X}_{tr}$ and $\mathbf{X}_{ts}$. Since the average concentrations at each node can be different, we standardize all variables representing the network nodes' data. Then, the training data along with a graph learning model and its hyperparameters are used to learn the graph. Figure 5.4 summarizes the different graph estimation processes evaluated and Table 5.2 shows the hyperparameters of the different methods along with their role in the graph learning process and their range of values. First, we learn the network topology with one of the different algorithms, the training set $\mathbf{X}_{\text{tr}}$, and its corresponding hyperparameters. A 5-fold cross-validation (CV) procedure is applied to the training to find the optimal hyperparameters leading to a graph which together with the reconstruction method obtain a minimum average CV RMSE. The result is a precision matrix $\mathbf{\Theta}$ for the graphical Lasso method, a weight matrix **W** for the distance-based method, and a Laplacian **L** for the smoothness-based method. Since these three matrices are related, we can obtain for each method a matrix of adjacencies **A** and a Laplacian **L** that can be used by the signal reconstruction methods. The resulting three matrices can be interpreted as feature selectors, using an adjacency matrix **A** with edges for the sensors that are connected. Then, with both the adjacency matrix **A** obtained by the graphical Lasso and the weights matrix **W** obtained from the distances, the Laplacian **L** can be obtained using the combinatorial Laplacian formula.

**Table 5.2:** Different method's hyperparameters to learn the network topology along with the role they play in the learning process.

| Method | Parameters | Role |
|---|---|---|
| Graphical Lasso | $\lambda$ | Controls the precision matrix sparsity. |
| Distance-based graph | $\tau$ | Gaussian kernel parameter. |
| | Threshold (TH) | Radius (in meters) in which edges are taken into account, controls sparsity. |
| Smoothness-based graph | $\alpha$ | Smoothness penalization constant. Controls sparsity of the Laplacian and smoothness. |
| | $\beta$ | $L_1$-norm Laplacian penalization constant. Controls the sparsity of the Laplacian. |

### Signal Reconstruction Training

Regarding the signal reconstruction with MLR, we need to select the set of features for every regression model, one model per node ($N$). So, we use the weight **W** or adjacency matrix **A**, to find the neighborhood of the i-th node $\mathcal{N}(x_i)$, that is, the nodes that are connected to the $i$-th node are used as features. Now, the signal reconstructed in the $i$-th vertex, $\hat{x}_i$, is a linear combination of the signal measured in the neighborhood $\mathbf{x}_{\mathcal{N}(x_i)}$. Since the signals measured at the nodes are highly correlated given that all sensors measure the same phenomena, even though they are located at different geographical positions, the problem of multicollinearity may arise. To avoid multicollinearity, we use the partial least squares (PLS) method to obtain components that are orthogonal to each other and make a dimensional reduction to avoid ill-posed conditioned matrix problems.

In the first phase, the training data of the neighbors of the $i$-th vertex, $\mathbf{X}_{\text{tr},\mathcal{N}(x_i)}$, are used together with the data of the $i$-th node to obtain the PLS components and the loading matrix $\mathbf{P} \in \mathbb{R}^{N\times N}$. The loading matrix **P** is now used to project the training regressors, $\mathbf{X}_{\mathrm{tr},\mathcal{N}(x_i)}$, onto the PLS components $\mathbf{X}^{PLS}_{\mathrm{tr},\mathcal{N}(x_i)}$. The goal is to keep only a few components until the condition number of the moment matrix $\mathbf{X}^{PLS^\mathsf{T}}\mathbf{X}^{PLS}$ is small enough to reduce possible multicollinearity problems [114]. To obtain the coefficients $\mathrm{w}_k$ of the linear regression, we train the linear model with the projections of $\mathbf{X}_{\mathrm{tr},\mathcal{N}(x_i)}$ as regressors ($\mathbf{X}^{PLS}_{\mathrm{tr},\mathcal{N}(x_i)}$) and with $\mathbf{x}_{\mathrm{tr,i}}$ as the dependent variable. Finally, by taking new data from the test data set and projecting it to the PLS components, we can reconstruct the $\hat{\mathbf{x}}_{\mathrm{ts,i}}$ signal from neighboring node signals $\mathbf{X}_{\mathrm{ts},\mathcal{N}(x_i)}$. The disadvantage of this method is that to reconstruct the signal in the $i$-th vertex $x_i$, we need all the values of the signals in its neighborhood. If there is missing data from any neighboring node for a given instant, then it is not possible to reconstruct the signal at that instant. Hence, if we are interested in reconstructing $N$ nodes of the network, using the same constructed graph we will create $N$ linear regression models to be able to reconstruct the signal in those nodes. Another clear disadvantage is that for the calculation of the loading matrix $\mathbf{P}$ the values of the neighboring nodes $\mathbf{X}_{\mathrm{tr},\mathcal{N}(x_i)}$ and the training values of the target node $\mathbf{X}_{\mathrm{tr,i}}$ are needed, that's why as with the regression model we have to obtain a loadings matrix for each different node signal reconstruction model.

Now, regarding the signal reconstruction using the Laplacian interpolation, we assume that we know the structure of the graph through the Laplacian, we know the value of the signal in $m<N$ vertices, and we have gaps, missing or corrupted values in the other $N-m$ vertices. Let's call $\hat{\mathbf{x}}$ the signal to be estimated since we know $\mathbf{x}$ and the set of nodes $\mathcal{M}$ where the values are known. The aim of the interpolation process is to estimate $\hat{\mathbf{x}}$ such that $\hat{x}_i = x_i$ for those vertices where $x_i$ is known ($i \in \mathcal{M}$). For this, we solve the optimization problem shown in eq. (5.20). This method does not need a training phase and can be applied directly to reconstruct the signal when values are missing in time step K, given the set of observed nodes $\mathcal{M}$, or when we want to include a virtual sensor, a case that represents a sensor where graph node has no data at all. Thus, for this case, the Laplacian matrix $\mathbf{L}$ allows the calculation of all weights for all possible signal reconstructions, even allowing the reconstruction of different nodes simultaneously (e.g., different nodes have missing samples at the same instant).

## 5.4. Experimental evaluation

In this section, we describe the data sets used for the experiments and we show the results of the comparison of the different graph inference models applied to the four data sets described in section 5.4.1 using the methodology explained in the previous section 5.3.3. We split the experiments into three blocks; reference station networks, the heterogeneous H2020 Captor network, and the analysis of the methodology scalability. In this way, we explore:

(A) **Reference station networks**, where we compare the different graph inference models and signal reconstruction techniques applied to high-precision instrumentation networks. We use 100% of the data sets to perform a 5-fold CV and show the CV results to analyze the behavior of the graphs in terms of signal reconstruction performance,

(B) **Heterogeneous low-cost sensor network H2020 testbed**, where we simulate the real application case, where 66% of the data is used as training to perform CV and obtain the hyperparameters and the graph describing the network. Then, we apply the graph and the signal reconstruction on the different nodes of the network in the test set (33% of the data) to simulate possible signal reconstructions during the air monitoring campaign,

(C) **Scalability**, where the scalability of the proposed graph-based approach is analyzed and a clustering-based approach is proposed to overcome possible issues.

### 5.4.1. DATA SETS

In order to compare the different graph inference methods and their application, we use two different types of real data sets; networks of governmental reference stations and a heterogeneous network of LCSs. The monitoring networks contain high-precision instrumentation to accurately measure an area of interest. Therefore, we use data from the Spanish government monitoring network, for the Barcelona metropolitan area, used for pollution measurement and compliance with European regulations. These data are open and available to any user via the web. To test the effectiveness of the methods we have used data for three different pollutants; tropospheric ozone ($O_3$), nitrogen dioxide ($NO_2$), and particulate matter 10 $\mu m$ ($PM_{10}$). Thus, these three data sets are listed in Table 5.3, with their characteristics, where the data cover a period of five months between 01/01/20019 to 30/05/2019 representing what would be an air monitoring campaign with measurements at a temporal resolution of one hour and a number of nodes varying from thirteen to twenty.

**Table 5.3:** Data sets 1, 2, and 3 use reference stations in Barcelona, Spain, and data set 4 uses three reference stations and five LCSs deployed in the H2020 CAPTOR project in Spain.

| Data set ID | Pollutant | # Nodes | Temporal resolution | # Samples | Period |
|---|---|---|---|---|---|
| 1 | $O_3$ | 15 | 1 h | 2775 | 01/01/2019 - 30/05/2019 |
| 2 | $NO_2$ | 20 | 1 h | 2526 | 01/01/2019 - 30/05/2019 |
| 3 | $PM_{10}$ | 13 | 1 h | 2595 | 01/01/2019 - 30/05/2019 |
| 4 | $O_3$ | 8 | 30 min | 2831 | 18/06/2017 - 16/09/2017 |

As for the data set of the heterogeneous network of LCSs, we used data from a heterogeneous network deployed by the European Captor H2020 project in the Vic area (Catalonia) during the summer of 2017 to carry out a tropospheric ozone monitoring campaign. The data set consists of three reference stations and four SGX Sensortech MICS 2614 $O_3$ sensors distributed over the deployment area. The same nodes explained in the previous chapters were used, using Arduino Yun as microcontroller, a 3G modem to send the data to a central database, and four ozone and temperature and relative humidity sensors. For the deployment, the sensors of the Captor nodes were calibrated *in-situ* with MLR during the first weeks where they were placed in reference stations (Vic, Tona, and Manlleu) near their deployment area. Once calibrated, they were placed at the sites of interest. Table 5.3 shows the characteristics of this fourth data set, constituted by eight nodes, with samples collected during three months and an hourly resolution of 30 minutes.

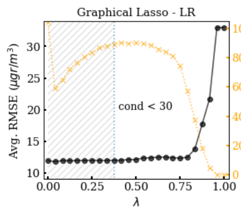
Ultimately, these data sets comprise a wide variety of data; different pollutants, networks with reference stations, and heterogeneous networks with *in-situ* calibrated LCSs and reference stations.

### 5.4.2. Reference station networks

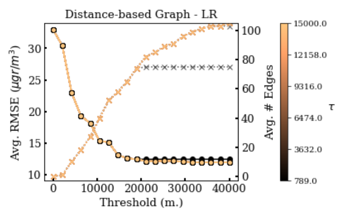
We apply the different methodologies explained in section 5.3.3 to the data sets of the reference stations in the Barcelona area (data sets 1, 2, and 3), which correspond to the air pollutants; $O_3$, $NO_2$, and $PM_{10}$. For illustrative purposes, we select the results from the $O_3$ (data set 1). Figure 5.5 shows the CV performance of the three methods in obtaining the network topology and the performance of the two signal reconstruction methods. Figures 5.5.a) and d) show the optimal parameters for choosing the best topology using graphical Lasso and reconstructing the signal using linear regression and Laplacian interpolation respectively. Figures 5.5.b) and e) and 5.5.c) and f) show the optimal parameters for the choice of the best topology using the distance-based method and the smoothness-based method with both linear regression and Laplacian interpolation respectively. First, we analyze the relationship between the hyperparameters of each model with the final number of edges in the resulting graph (graph density) and with the signal reconstruction performance denoted by the root mean square error (RMSE) value obtained by each method. Then, we compare the performance of the models, discussing their advantages and disadvantages. The RMSE used to select the best hyperparameters is the average of the cross-validation RMSE of all the nodes participating in the network, that is, we reconstruct each of the network nodes one by one during each CV test fold. Thus, the choice of the graph inference method depends on how well the signal reconstruction model performs on top of the inferred graph.

#### Graphical Lasso

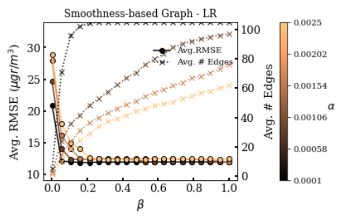
Figures 5.5.a) and d), dotted orange curve, show the graph's number of edges as a function of the $\lambda$ hyperparameter for the graphical Lasso method. Indeed, recalling the graphical Lasso formulation, the larger the $\lambda$ value the more penalization is added to the $l_1$-norm of the $\mathbf{\Theta}$. It can be seen that as $\lambda$ approaches 1.0, the number of edges decreases, becoming the precision matrix more and more sparse, and then in the $10^{-4}$ value (there is almost no regularization of the matrix) producing a totally connected graph. However, we can also observe a hump effect for $\lambda$ values between $10^{-4}$ and 0.35 producing an increasing number of edges trend as the $\lambda$ value increases. This hump effect observed for $\lambda$ values between $10^{-4}$ and 0.35 is produced by poor conditioning of the empirical covariance matrix $\hat{\mathbf{\Sigma}}$ when solving the graphical Lasso method and has already been reported in other studies [115]. This problem appears because the graphical Lasso algorithm uses the inverse of the empirical covariance matrix as an initial guess, so that, the inverse of an ill-conditioned matrix may produce the unstable results observed. An ill-conditioned moment matrix $\mathrm{COV}[\mathbf{X}] = \mathrm{E}[\mathbf{X}\mathbf{X}^\top]$ (for the case of standardized variables) may be caused by the presence of multicollinearity, indeed, Heinävaara *et al.* [115] explain how this problem can lead to bad conditioning of the covariance matrix $\hat{\mathbf{\Sigma}}$ and thus to the instability of the results. A condition number above thirty $\mathrm{cond}(\mathbf{\Sigma}^{-1}) > 30$ indicates that the regression may suffer from severe multicollinearity [116]. For all this, only values of $\lambda$ for which the initial guess of the covariance matrix $\mathbf{\Sigma}$ has a condition number less than thirty are taken into account, thus obtaining a stable solution. This, together with the



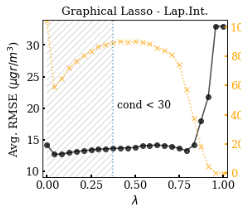
(a) Validation metrics for $O_3$ reference stations with graphical Lasso and linear regression procedure.



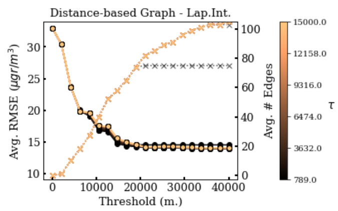
(b) Validation metrics for $O_3$ reference stations with the distance-based graph and linear regression procedure.



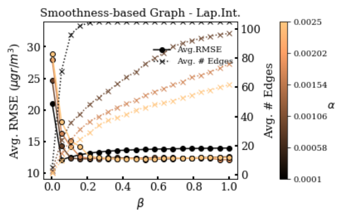
(c) Validation metrics for $O_3$ reference stations with the smoothness-based method and linear regression procedure.



(d) Validation metrics for $O_3$ reference stations with graphical Lasso and Laplacian interpolation procedure.



(e) Validation metrics for $O_3$ reference stations with the distance-based graph and Laplacian interpolation procedure.



(f) Validation metrics for $O_3$ reference stations with the smoothness-based method and Laplacian interpolation procedure.

**Figure 5.5:** Average cross-validation RMSE and average number of edges of the different techniques applied to data set 1. The shaded area corresponds to hyperparameter values that produce ill-posed problems (condition number of the initial guess greater than 30) for the graphical Lasso.

PLS procedure, reduces the effects of multicollinearity.

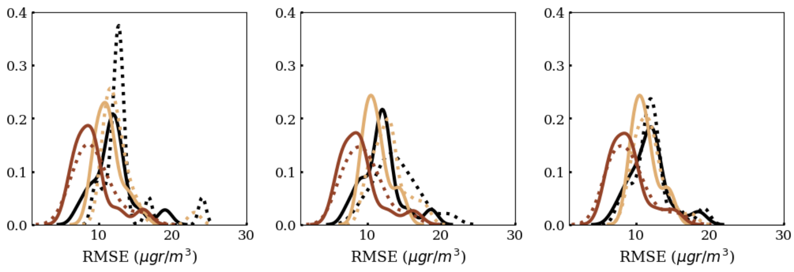
From the results of the CV grid search shown in Figure 5.5, Table 5.4 shows the best configuration for each model corresponding with the hyperparameters producing the lowest average CV RMSE. The minimum average RMSE value with graphical Lasso and the linear regression method is 11.96 $\mu$gr/m$^3$, and is produced with a $\lambda$ value of 0.375 and an average number of edges in the graph of 89.4 edges, while with Laplacian interpolation, the minimum RMSE is produced with $\lambda$ of 0.792 and RMSE of 13.26 $\mu$gr/m$^3$ and 57.2 edges on average, Table 5.4. Overall, the MLR signal reconstruction obtains the lowest CV RMSE at the cost of a denser graph (larger number of edges) than the best configuration for the Laplacian interpolation reconstruction method. The same behavior is observed for the other two pollutants ($NO_2$ and $PM_{10}$), Table 5.4. Nevertheless, the performance gap between the MLR and Laplacian interpolation is less significant than in the $O_3$ case. Regarding the signal reconstruction capability of the different air pollutants, in terms of $R^2$ the $O_3$ gives better values than $NO_2$, which in turn is better than the $PM_{10}$ values. Figure 5.6.a) shows the empirical distribution of the CV RMSEs of the reference stations for the best graphical Lasso configurations. As it can be seen from the three contaminants and the two reconstruction methods, the distributions are positively skewed with some reference stations with large RMSEs compared to the distribution mode. This means that there may exist sensors in a network that are not related to other sensors, so they cannot benefit from reconstruction using network data, producing worse estimates. It is observed that the distribution mode of the Laplacian interpolation reconstruction is larger than the MLR case, with more extremely bad-performing sensors (long right tails).

**Definition 13** *The **density** of a graph $\mathscr{G}$ can be defined as the ratio between the graph edge number and the maximum number of possible edges. We define the density of a graph as:*

$$D(\mathscr{G}) = \frac{|\mathscr{E}|}{|\mathscr{V}|(|\mathscr{V}|-1)/2} \tag{5.22}$$

**Table 5.4:** Best CV metrics for data sets 1, 2, and 3 respectively applying the different graph learning and signal reconstruction models.

| | | | $O_3$ | | | $NO_2$ | | | $PM_{10}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **RMSE** | **# Edges** | **$R^2$** | **RMSE** | **# Edges** | **$R^2$** | **RMSE** | **# Edges** | **$R^2$** |
| **Graphical** | **LR** | 11.96 | 89.4 | 0.70 | 11.39 | 109.0 | 0.66 | 9.11 | 61.2 | 0.46 |
| **Lasso** | **Lap.Int** | 13.26 | 57.2 | 0.54 | 12.75 | 53.2 | 0.57 | 9.50 | 60.2 | 0.45 |
| **Distance** | **LR** | 11.90 | 101.0 | 0.71 | 11.23 | 190.0 | 0.67 | 9.02 | 76.0 | 0.48 |
| **Graph** | **Lap.Int** | 13.94 | 103.0 | 0.62 | 12.89 | 140.0 | 0.56 | 9.50 | 77.0 | 0.45 |
| **Smoothness** | **LR** | 11.84 | 102.2 | 0.71 | 11.23 | 190.0 | 0.67 | 9.00 | 40.4 | 0.49 |
| **Graph** | **Lap.Int** | 12.08 | 71.2 | 0.71 | 11.81 | 56.0 | 0.63 | 9.04 | 70.2 | 0.50 |



(a) RMSE distribution obtained with the graphical Lasso.

(b) RMSE distribution obtained with the distance-based graph.

(c) RMSE distribution obtained with the smoothness-based graph.

**Figure 5.6:** Empirical RMSE distributions for the best configurations for the different graph learning and signal reconstruction methods. The continuous lines draw the linear regression while the dotted lines draw the Laplacian interpolation. The black, orange, and red lines represent $O_3$, $NO_2$, and $PM_{10}$ respectively.

## Distance-based method

Recalling the formulation of a distance-based graph, this method uses two hyperparameters; $\tau$ is a Gaussian kernel normalization parameter, and the threshold of the distance $TH$ that controls the sparsity of the resulting weight matrix $\mathbf{W}$. The units of $\tau$ and $TH$ are in meters. Figure 5.5.b) and e), dotted orange and gray curves with crosses, show that larger threshold values $TH$ allow for denser graphs. Line colors correspond to different $\tau$ values, the orange curve is obtained for a high $\tau$ while the gray curve is obtained for a smaller $\tau$. By increasing the distance threshold $TH$ we are increasing the number of neighbors a node has, and therefore the number of edges of the graph increases. Secondly, for those neighboring nodes which are within the radius $TH$ meters, a large $\tau$ value will assign a small weight, while a small $\tau$ value will assign a large weight. This makes large values of $\tau$ produce more connected networks with more edges than smaller values of $\tau$. The lowest CV RMSE value obtained with linear regression is 11.90 $\mu$gr/m$^3$ with 101 edges, with a $TH$ of 33710 meters and a $\tau$ of 1580 meters. The lowest RMSE value with Laplacian interpolation is 13.94 $\mu$gr/m$^3$ with 103 edges, with a $TH$ of 35817 meters and a $\tau$ of 2360 meters. We can see that the signal reconstruction method with linear regression produces less RMSE than the Laplacian interpolation method with similar values in the number of edges for $O_3$ and $PM_{10}$. For $NO_2$, the RMSE value is also lower for linear regression, but in this case with a higher number of edges. This means that with similar graphs, the MLR is capable of weighting the different sensors accordingly for signal reconstruction, while the distance between the nodes may not correctly weight the different signals for reconstruction. Again, the prediction capabilities of the different pollutants in terms of $R^2$ give better values for $O_3$ than $NO_2$, which in turn is better than the $PM_{10}$ values. The empirical distributions of the nodes CV RMSE in Figure 5.6.b) show the same trend as with the graphical Lasso where some reference stations cannot be well reconstructed using network data, showing low efficiency in capturing the relationships between nodes. This means that defining edges based on distances, or using a distance threshold as a feature selector, may include non-important relationships between sensors or exclude important relationships between sensors.

### SMOOTHNESS-BASED METHOD

The smoothness-based optimization problem uses two hyperparameters; the $\alpha$ controls the signal smoothness and the sparsity of the Laplacian matrix, and the $\beta$ penalizes the Frobenius norm of the Laplacian matrix $\mathbf{L}$. The joint combination $\{\alpha, \beta\}$ controls the sparsity and smoothness of the graph. Low values of $\beta$ and large values of $\alpha$, orange curves with crosses in 5.5.c) and f), promote sparser graphs and smooth representations. This is because the Frobenius norm of $\mathbf{L}$ tends to be small when $\beta$ increases, and decreasing $\beta$ has the opposite effect. Besides, when $\alpha$ increases the total variation of the training signals with respect to the graph $\mathrm{tr}(\mathbf{Y}^\top \mathbf{L}\mathbf{Y})$ is more penalized, promoting both sparse and smooth solutions. Regarding the average CV metrics, the lowest RMSE value achieved with linear regression is 11.84 $\mu$gr/m$^3$ with 102 edges, an $\alpha$ of $10^{-4}$, and a $\beta$ of 0.15. The lowest RMSE value achieved with Laplacian interpolation is 12.08 $\mu$gr/m$^3$ with 71 edges, an $\alpha$ of $10^{-4}$, and a $\beta$ of 0.05, being the best performing Laplacian interpolation of the three graph inferring techniques. Again, we can see that the signal reconstruction method with linear regression produces less RMSE than the Laplacian interpolation method with sparser graphs for $O_3$ and $NO_2$. For $PM_{10}$, the RMSE value is also lower for linear regression, but in this case with a higher number of edges. In general, $R^2$ gives better values for $O_3$ than $NO_2$, which in turn is better than the $PM_{10}$ values.

Finally, Figure 5.6.c) shows the distributions of the RMSEs obtained with the smoothness -based graph. The same trend is observed for all methods and contaminants, a positively skewed distribution, showing some stations that cannot be reconstructed with less error. Although, in this case, we can observe how the Laplacian interpolation is able to obtain RMSEs similar to the linear regression since the modes of the distribution coincide, except for the stations that cannot be well predicted.

#### DISCUSSION OF THE SIGNAL RECONSTRUCTION METHODS COUPLED WITH GRAPH TOPOLOGY INFERENCE

Regarding the performance of the different signal reconstruction methods, it is worth mentioning that the learned topology acts as a feature selector for the reconstruction, which has a large impact on the performance. Thus, the covariates for each signal reconstruction task are defined as $\hat{x}_i \sim \mathbf{x}_{\mathcal{N}(x_i)}$, whereby the connectivity of the graph and its density play an important role. In this experiment, we have selected the best graph, with its hyperparameters, based on the minimum average CV RMSE obtained, to obtain a graph whose superimposed signal reconstruction has a good performance. As an example, Figure 5.7.a), shows the 57-edge target graph learned with graphical Lasso for the $O_3$ data set can be observed.

The effect of the hyperparameters of the different models on the connectivity of the network is very diverse. In the case of the graphical Lasso, as mentioned above, multicollinearity produces erratic effects for low values of $\lambda$, requiring the diagnosis of the condition number to check the validity of the results. In the valid range of $\lambda$, we see how the range [0.75, 0.85] is very sensitive producing abrupt changes in the number of edges of the graph. In the case of the distance-based method, the number of edges $|\mathcal{E}|$ is directly governed by the defined threshold $TH$, which eliminates edges at more distant nodes. Thus, by increasing the value of the $TH$ we are inducing connections to more distant sensors, which may or may not be related to the node in question. This effect can also be seen as increasing the neighborhoods of the nodes $\mathcal{N}(x_i)$ by increasing the spherical-shaped neighborhood with radius $TH$. As we have observed from the pair $\{\tau, TH\}$ the threshold is the one that has the greatest impact on the reconstruction of the nodes. Finally, in the case of the smoothness-based graph, we see that it is the model that best adapts to the graph signals presented by the network, and consequently, it is the graph that works best with the Laplacian interpolation. In this case, the tuple models jointly the fit of the graph to the data and the graph density. In Figures 5.5.c and f) it can be seen how the graph density is very sensitive to the value of the alpha, although the sensitivity of the graph is given by the ratio (alpha beta), where we see that for higher alphas increasing the beta only implies a small increase in the number of edges.

To see the ability of the different models to find the correct neighborhood $\mathcal{N}(x_i)$ of each of the sensors, depending on the density of the graph, we conduct an experiment allowing an increase of 0.5 $\mu gr/m^3$ in the RMSE to obtain the best model allowing sparser solutions. For simplicity, we show the results for the $O_3$ data set, where the graphical Lasso model with linear regression goes from 11.96 with 89 edges to 12.45 with 57 edges, and with Laplacian interpolation it is not able to lower the number of edges. The distance-based graph with linear regression goes from 11.90 $\mu$gr/m$^3$ with 101 edges to 12.42 $\mu$gr/m$^3$ with 74 edges, and with Laplacian interpolation goes from 13.94 $\mu$gr/m$^3$ with 103 edges to 14.36 $\mu$gr/m$^3$ with 65 edges. Finally, the smoothness-based model with linear regression goes from 11.84 $\mu$gr/m$^3$ with 102 edges to 12.44 $\mu$gr/m$^3$ with 31 edges, and with Laplacian interpolation goes from 12.08 $\mu$gr/m$^3$ with 71 edges to 12.43 $\mu$gr/m$^3$

with 33 edges. As the results show, the smoothness-based graph with Laplacian interpolation allows a large reduction in the number of edges (from 71 to 33) in exchange for a small increase in error. Thus, there is a trade-off between reconstruction performance and graph density. This is of particular interest since in some situations optimality can be relaxed in exchange for a sparser graph. For example, in the case of a distributed data reconstruction, where each node has to communicate its value to its neighbors, it is interesting to have as sparse a network as possible to minimize the communication cost.

We can see that in general, although by very little difference, the MLR is able to obtain a better RMSE than the graph signal reconstruction model. This makes sense given a fixed graph structure, with different neighborhoods, the MLR is the best linear minimum mean squared error (MMSE) estimator for a fixed topology (fixed set of features). On the other hand, we see how the Laplacian interpolation method has a superior error in general with respect to the MLR, although in the case of the smoothness-based graph, we see how it is able to obtain an error similar to the MLR with a sparser graph, without being suboptimal. Thus, although the Laplacian interpolation is suboptimal, in the sense of mean squared error, it can result in a better performance for sparse graphs being able to obtain a more efficient trade-off between error and graph density than in the case of the graphical Lasso with MLR. Thus, Laplacian interpolation represents an efficient alternative to MLR with better results in the case of sparse graphs.

#### ADVANTAGES SEMI-SUPERVISED/GRAPH SIGNAL RECONSTRUCTION SETTING

The use of supervised models for data reconstruction represents a clear disadvantage from an engineering point of view. Given the heterogeneity of these contamination sensor networks, there can be a large number of sensor missings or gaps due to malfunctioning or under maintenance. Therefore, it would be necessary to have another preprocessing model to be able to impute the possible missings values in the neighborhood of a sensor or to train all the possible models taking into account that any node can fail, resulting in $2^{|\mathcal{N}(x_i)|}-1$ combinations per node, which is not feasible. In the case of using a missing value imputation method, its performance would directly impact the performance of the signal reconstruction. On the other hand, the Laplacian interpolation is a graph-based semi-supervised learning method (or graph signal reconstruction method) that allows adapting naturally to this setting where any set of nodes can have missing and be estimated using the available neighbors. Thus, this semi-supervised alternative provides robustness and resilience in the reconstruction and maintenance of data from a sensor network.

To see how important it is for the reconstruction methods to be flexible with respect to the presence of missings, Table 5.5 shows the different missings percentages for the three reference station data sets. We can see how each station has on average between 1.66-2.60% of missings. This means that the loss of data or the presence of missings is not only a problem that can affect low-cost sensors but also affects the reference instrumentation. Given the high heterogeneity of this type of network and the possible sensing and communication failures that may occur, the presence of missings is very common. The problem is that the percentage of incomplete samples (i.e., with missings) will depend on the size of the neighborhood of each sensor. Assuming a complete dense graph (100% of density), we have that the neighborhoods of each of the sensors include all the other sensors, $\mathcal{N}(x_i) = \mathcal{V} \setminus i$, then if there is a node that presents a missing for a time $t$, the sample is incomplete. The last column of Table 5.5 shows how with a complete network about 21.88-28.87% of the samples are incomplete, which is a very high percentage. This means that in the case of using a supervised model for signal reconstruction there would be three options; *i)* remove the sample, which would reduce the network monitoring capability, *ii)* impute the missing values using an imputation method, affecting the subsequent reconstruction, and *iii)* have another model trained that does not include the missing sensor, computationally infeasible. It is also possible to mitigate the impact of missings by controlling the network density and thus obtaining a better data cleansing technique. For a sparse graph, having a missing value in a node $x_i$ only affects the neighborhood of that node $\mathcal{N}(x_i)$, the other nodes can be reconstructed as usual since they have a neighborhood without missings. Thus, considering a graph with a density of 50%, the number of samples that are affected by missings is reduced to 10.73-15.87% of the samples (Table 5.5) since even if a node has missings, the nodes that are not in its neighborhood can be reconstructed in a natural way. In view of this, the smoothness method promotes sparsity and the Laplacian interpolation method naturally fits the situation of simultaneously estimating the missing nodes and the target node.

**Table 5.5:** Reference stations with incomplete observations

| | Avg. % missings per station | % incomplete samples graph with ~50% edges | % incomplete samples |
|---|---|---|---|
| **Data set 1** | 1.66 | 10.73 | 21.88 |
| **Data set 2** | 1.90 | 15.87 | 28.87 |
| **Data set 3** | 2.60 | 12.00 | 26.94 |

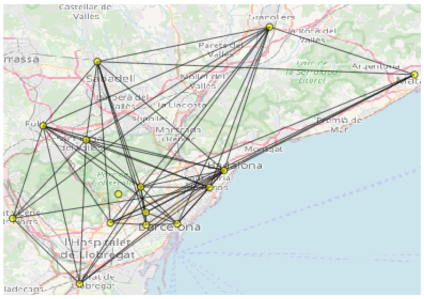
Just as an example of how Laplacian interpolation can handle missings, we performed an experiment with the smoothness-based graph and data set 1 ($O_3$) where a 5% random loss occurs in all nodes in the test phase. The result is shown in Table 5.6, and as it can be seen, the average RMSE has increased slightly from 12.08 (see Table 5.4) to 12.15 $\mu gr/m^3$, but the reconstruction has been possible in all nodes with a small increase of the RMSE, showing the resilience provided by using a smoothness-based graph coupled with a semi-supervised signal reconstruction method.

**Table 5.6:** CV RMSE with 5% of missing data at each node (during each test CV fold) using the smoothness-based graph inference and the Laplacian interpolation.

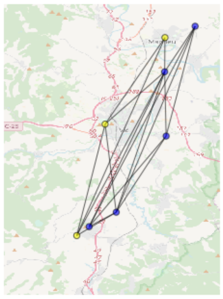
| Mean RMSE | Std RMSE | Max RMSE | Min RMSE |
|---|---|---|---|
| 12.15 | 2.70 | 19.28 | 8.07 |

### 5.4.3. Heterogeneous low-cost sensor network: H2020 Captor

The previously studied data sets correspond to reference station networks where the different nodes are high-precision instruments. In this section, we use a heterogeneous network with three reference stations, called Vic, Tona, and Manlleu, and five metal-oxide LCSs monitoring $O_3$ that have been calibrated *in-situ* before the deployment. This type of heterogeneous network with LCSs allows us to evaluate:



(a) Network topology learned with the graphical Lasso with 57 edges and $\lambda = 0.5$, area of Barcelona.



(b) Network topology learned with the smoothness method with 18 edges, $\alpha = 0.0017$ and $\beta = 0.6319$, low-cost CAPTOR network.

**Figure 5.7:** On the left is the metropolitan area of Barcelona with several ozone reference stations. On the right is the area of Manlleu, Vic, and Tona, where the CAPTOR network was deployed. Reference stations are depicted in yellow and LCSs in blue.

1. How a heterogeneous network with a mix of reference stations with accurate values and low-cost nodes with less accurate measurements behaves when using a graph with a topology learned from structured data, and

2. The quality of the estimates using the network data to see if their accuracy is comparable or better than that of a LCS calibrated *in-situ* and deployed at that location. This would allow applications such as recalibration or multi-hop calibration [10].

### SIGNAL RECONSTRUCTION AND GRAPH INFERENCE PERFORMANCE IN A HETEROGENEOUS NETWORK

Table 5.7 shows the test set results for the heterogeneous sensor network. The table shows specifically the error obtained in the reconstruction of the reference station nodes (to compare with the three LCSs deployed at the same location) as well as error statistics for the whole network reconstruction. The first thing we notice is that the models behave in a similar way when we have a heterogeneous network of nodes than when we had only reference stations giving accurate values. Signal reconstruction using linear regression gives slightly better results than Laplacian interpolation, and in general, the smoothness-based graph learning method with reconstruction based on Laplacian interpolation gives good results with few edges. It is important to note the disadvantage of the distance method. It can be observed first that the number of edges is the highest among the three methods, and that with Laplacian interpolation the RMSE is very high, in the order of 20 $\mu$gr/m$^3$. This is because when using distances the neighborhoods may not be well defined, using nearby LCSs. In this sense, both the graphical Lasso and the smoothness-based methods are effective in inferring the relationships of the data captured by the nodes regardless of the proximity of the nodes. Thus, distance-based graphs may suffer from erroneous edges and weights given the high level of heterogeneity of this type of network. In contrast, data-driven graph learning models work for both homogeneous and heterogeneous networks because they are based entirely on the measurements. Figure 5.7.b) shows the learned graph with an 18-edge target with the smoothness-based model.

**Table 5.7:** Test set results for Captor data set 4, including the corresponding results for the reference stations nodes of Manlleu, Vic, and Tona.

| | | Manlleu RMSE | Vic RMSE | Tona RMSE | Mean RMSE | Std RMSE | Max RMSE | Min RMSE | # Edges |
|---|---|---|---|---|---|---|---|---|---|
| **Graphical** | **LR** | 11.72 | 10.08 | 11.41 | 11.79 | 1.63 | 15.10 | 9.92 | 16 |
| **Lasso** | **Lap.Int.** | 12.02 | 10.21 | 12.09 | 11.80 | 0.85 | 12.94 | 10.21 | 14 |
| **Distance** | **LR** | 12.79 | 9.77 | 12.22 | 11.43 | 1.15 | 12.79 | 9.77 | 25 |
| **Graph** | **Lap.Int.** | 20.60 | 20.10 | 18.55 | 15.94 | 3.50 | 20.60 | 12.04 | 28 |
| **Smoothness** | **LR** | 12.06 | 9.77 | 11.71 | 11.46 | 1.06 | 12.72 | 9.77 | 28 |
| **Graph** | **Lap.Int.** | 11.82 | 10.11 | 11.59 | 11.65 | 0.85 | 12.72 | 10.11 | 18 |
| ***In-situ* LCS** | | 10.85 | 11.30 | 12.21 | | | | | |

### Network prediction versus *in-situ* low-cost sensor

The idea is to test the network's ability to predict $O_3$ concentrations at points where it has not been possible to deploy a sensor or where a recalibration can be performed due to the drift or aging of the sensors. To do this, we place a LCS at a point, e.g., at the reference stations, and compare the value given by the sensor with the predicted value using the different signal reconstruction models at that point. This sensor is identified in Table 5.7 with the label *in-situ* LCS. It is observed in Table 5.7 that except for the distance-based model with Laplacian interpolation, the rest of the models are capable of making an estimation with similar accuracy as having a LCS. Therefore, in the situation of having a reference station at a site during the graph learning period, and then becoming unavailable during the testing (may have moved to another site or it may not be available) the signal reconstruction estimations would produce similar results to having a pre-calibrated LCS on the same location. This may be due to the effect of the presence of reference stations in the neighborhood of the target node, which shows the potential of this methodology to recalibrate sensors without relocating the node in a reference station or increase the spatial resolution by including virtual sensors in the network. Taking a look at the learned graph for the Captor network, Figure 5.7.b), it is seen how the reference stations are mostly connected to other reference stations, given that the smoothness-based methods rely on the measured data and LCSs may not perform as similar to the reference stations. Thus, the real concentrations at one reference station can be reconstructed mainly using the real concentrations measured by other reference stations and to a lesser extent thanks to LCS measures.

## 5.4.4. Scalability

In this section, we analyze the scalability of the methodology shown in section 5.3.3, for the case of the smoothness-based graph learning methods and Laplacian interpolation, where the main components are graph learning and graph signal reconstruction tasks. Regarding the graph learning model, we use a model that iteratively solves a quadratic program that scales quadratically with the number of nodes of the network $N$ and performs matrix inversion operations. In addition, this model has two hyperparameters,

$\{\alpha, \beta\}$, which need to be found by 5-fold cross-validation. In the case of the graph signal reconstruction model, the model used (Laplacian interpolation) has as the most expensive operation a matrix inversion or a matrix multiplication depending on the number of observed nodes, $|\mathscr{U}| << |\mathscr{M}|$. The Laplacian interpolation does not have any hyperparameters (contrary to other cases that we will see in the following chapter 6), so the resulting hyperparameters are denoted as $\mathbf{\Omega} = \{\alpha, \beta\}$. Thus, the most computationally expensive process is graph learning since it is a quadratic program that scales with the number of nodes in the network. The more hyperparameters the more dimensions the grid search to be performed in the cross-validation will have and the more expensive it will be. During the prediction phase, the only task that involves a higher computational cost is the graph signal reconstruction.

Once the hyperparameters are selected and thus the graph learned, it is not necessary to use a training set to train the graph signal reconstruction method, since these are transductive methods, where only the values of the set of observed nodes $x_{\mathscr{M}}$ are needed to interpolate the unobserved ones $x_{\mathscr{U}}$ at time instant $t$. In the test set, the cost of the predictions will depend on the computational complexity of the graph signal reconstruction. To speed up the reconstruction, if the set of nodes does not vary, the signal reconstruction coefficients can be calculated once and applied at different time instants $t$. However, if this set changes, which is a common situation where some of the nodes may have missing values and need to be reconstructed, the signal reconstruction coefficients must be recalculated. If cross-validation becomes extremely computationally expensive, a greedy approach can be carried out, where a graph with a desired level of sparsity is first obtained, and then cross-validation is performed only for the hyperparameters of the reconstruction method.

The most demanding task is graph learning since it involves iteratively solving a convex problem, and the complexity of this problem scales quadratically with the number of nodes N. However, as we are dealing with air pollution data, there exist spatial patterns and correlations between sensors, so the graph learning problem can be split to learn a number of disjoint subgraphs without having a large impact on the performance. Thus, we propose to find the approximation of a large graph as a set of disjoint subgraphs with a small impact on the signal reconstruction error. In a similar problem, Stein *et al.* [117] propose a cluster-based methodology to solve the Kriging scalability issues, learning a single Kriging model per cluster. Therefore, given the correlations present in our data, a clustering-based approach to partitioning the graph learning problem is a good strategy to cope with the scalability problem.

Given the spatial correlation of certain pollutants (e.g. $O_3$ and $NO_2$) and the assumption that sensors with different variability patterns should be weakly connected, we divide the reference stations into $C$ clusters and learn $C$ Laplacians independently, resulting in $C$ disjoint subgraphs. Figure 5.8 shows the graph learning process for large graphs, where using the normalized time series of each of the nodes $\mathbf{x}_i \in \mathbb{R}^P$, where $P$ is the number of training samples, a clustering algorithm can be applied to the N nodes to find $C$ clusters. The number of clusters can be decided in several ways, e.g., unsupervised metrics such as the Calinski-Harabasz or the Silouhette index [118]. Depending on the specific data set the size of the clusters may vary, but the computational complexity improvement will always depend on the size of the largest cluster, the smaller the

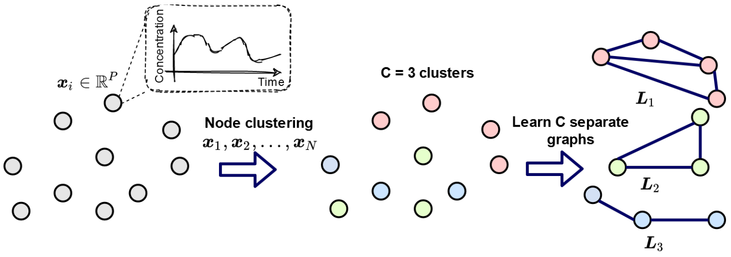


**Figure 5.8:** Clustering methodology for the graph learning and signal reconstruction scheme. In this precise example, the number of clusters is set to three, $C = 3$.

greater the improvement, as the subgraph learning can be solved in parallel. The performance of the solution will always depend on the localization of the pollutants and the dependencies present in the data. In addition, the resulting Laplacians $\mathbf{L}_i$ can be treated independently or joined together in a Laplacian in the form of a block-diagonal matrix:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & \mathbf{0} \\ & \mathbf{L}_2 & \\ \mathbf{0} & & \ddots & \\ & & & \mathbf{L}_C \end{bmatrix} \tag{5.23}$$

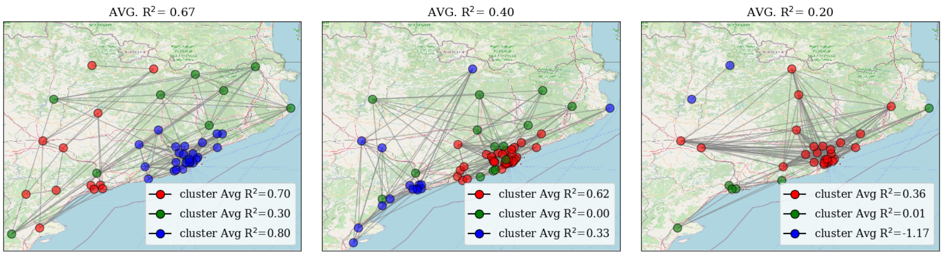
Algorithm 3 describes this cluster-wise procedure, where a different Laplacian $\mathbf{L}_i$ is learned for each cluster of nodes $\mathscr{C}_i$, with their corresponding $\alpha$ and $\beta$ hyperparameters. Once the Laplacians are learned, the graph signal reconstruction $\mathbf{f}(\cdot)$ can be applied in a cluster-wise manner, since nodes from different clusters are not connected. Algorithm 3 shows how this cluster-wise graph learning and signal reconstruction methodology can be applied, where $\{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$ are the set of graph learning hyperparameters for each one of the clusters, $\mathbf{X}$ is the training set of graph signals, $\mathbf{f}(\cdot)$ is the signal reconstruction model, **hyp** are the possible graph signal reconstruction model hyperparameters, and $C$ is the number of clusters. The different hyperparameters **hyp** and the number of clusters $C$ can be set by cross-validation and inspecting cluster-quality indexes, respectively.

In order to show an experiment applying algorithm 3, we used three data sets of air pollution sensors comprising reference stations in the area of Catalonia, Spain. These data sets have a total of 46, 60, and 33 nodes for $O_3$, $NO_2$, and $PM_{10}$ respectively. More information about these data sets is available in the next chapter 6 since these are used as baseline for the experiments. The objective of the experiment shown below is to compare the average $R^2$ obtained in the node-to-node reconstruction when using a single graph and when using three disjoint graphs.

Figure 5.9.a) shows the results for reconstructing each one of the nodes learning three Laplacians (one per cluster) with the $O_3$ data set using the hierarchical clustering algorithm (colors denote the clusters obtained), which is a widely studied algorithm for clustering air pollution time series [118], using the euclidean distance, and the ward linkage criterion. The results show a cluster of well-defined nodes that have a very good average CV $R^2$, 0.80, another that has a good average $R^2$ with 0.7, and another set of stations

**Algorithm 3** Cluster-wise graph learning and signal reconstruction.

```
Input: $\{\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{X}, \mathbf{f}(\cdot), \mathbf{hyp}, C\}$
 1: $\bar{\mathbf{X}} \leftarrow$ Standardization($\mathbf{X}$)
 2: $\mathscr{C}_1, \ldots, \mathscr{C}_C \leftarrow$ Clustering($\bar{\mathbf{X}}, C$)          ◁ ↑ Scalability
 3: for $i = 1$ to $C$ do
 4:     $\mathbf{L}_i \leftarrow$ Graph_Learning($\alpha_i, \beta_i, \bar{\mathbf{X}}_{\mathscr{C}_i}$)
 5: end for
 6: while $\mathbf{x}_{new}$ do                              ◁ New sample collected
 7:     $\mathscr{U} \leftarrow$ Get_Unobserved_Nodes($\mathbf{x}_{new}$)
 8:     $\mathscr{M} \leftarrow \mathscr{V} \setminus \mathscr{U}$
 9:     $\bar{\mathbf{x}}_{new} \leftarrow$ Standardization($\mathbf{x}_{new}$)
10:     for $i = 1$ to $C$ do
11:         $\bar{\mathbf{x}}_{new_{\mathscr{C}_{i_{\mathscr{U}}}}} \leftarrow \mathbf{f}(\bar{\mathbf{x}}_{new_{\mathscr{C}_{i_{\mathscr{M}}}}}, \mathbf{L}_i, \mathbf{hyp}_i)$          ◁ Reconstruction
12:     end for
13:     $\mathbf{x}_{new} \leftarrow$ Unstandardization($\bar{\mathbf{x}}_{new}$)
14:     return $\mathbf{x}_{new}$
15: end while
```



(a) $O_3$ data set. (b) $NO_2$ data set. (c) $PM_{10}$ data set.

**Figure 5.9:** Results for a clustering example, with $C$=3, applied to three data sets, comprising the reference stations of the Catalonia area, using the Laplacian interpolation.

that cannot be reconstructed well ($R^2$ of 0.30). Looking further into the results, we see how the cluster that is best predicted (blue cluster) is the one corresponding to an area with the highest density of reference stations while the second one (red cluster) has a few stations nearby. Furthermore, it can be observed how the clustering algorithm is able to group the nodes that do not benefit from the neighborhood information, it is to say, related nodes are grouped together and the remaining cluster contains those sensors that do not belong to any other cluster. However, we note that unrelated nodes can be included in the same cluster but still the graph learning algorithm will take care of assigning the weights corresponding to those relationships. Regarding the $NO_2$ and $PM_{10}$, Figures 5.9.b) and c) respectively, we see how the $R^2$ has worsened a little from 0.42 to 0.40 and from 0.26 to 0.20 with respect to learning the whole graph. In the case of the $NO_2$, there is only one cluster that has a good average CV $R^2$ (red cluster), which corresponds and some other reference stations located nearby. The other clusters have sensors whose reconstruction is not so good, it can be observed that they are much more distributed stations, a less spatially dense network. In the case of $PM_{10}$, it is observed that in one of the clusters $R^2$=-1.17, with a negative value, which means that the model does not follow the trend of the data, and therefore fits worse than if there was a null model. In general, none of the clusters obtains a good $R^2$, given the difficulty of predicting $PM_{10}$, thus we can observe that the clustering results in one very large cluster (red) and two other small clusters, since in general the relationships between $PM_{10}$ nodes are not well defined. These clusters correspond to nodes that cannot be reconstructed given their location and lack of influential neighboring information.

**Table 5.8:** Comparison of the average $R^2$ using one single graph and a graph per cluster.

| Data set | Original $R^2$ | Cluster $R^2$ | Problem size reduction |
|---|---|---|---|
| $O_3$ | 0.66 | 0.67 | ↓ 47.83% |
| $NO_2$ | 0.42 | 0.40 | ↓ 48.33% |
| $PM_{10}$ | 0.26 | 0.20 | ↓ 24.24% |

Table 5.8 shows the average CV $R^2$ using a signal graph and using clustering methodology to improve the scalability. The coefficients of determination are mostly maintained or are slightly lower, so the scalability is improved at almost no signal reconstruction cost. The "problem size reduction" label indicates the reduction in the size of the resulting graph learning with respect to the original case, in the cluster-based learning case, the size is defined by the size of the largest cluster, which will be the computational bottleneck. Therefore, the problem size is reduced to around 24-48%, meaning that the largest cluster is of size 75-52% of the original graph, in almost all cases. As a summary, it is suggested that the problem of graph learning can be broken down into $C$ problems with $C$ clusters with the most similar reference stations, thus eliminating connections between subgraphs without a large impact on the signal reconstruction performance.

## 5.5. Concluding remarks & Future work

In this chapter, we have reviewed the first question regarding the analysis of air pollution sensor networks using graphs, which addresses the first aspect to be taken into account when analyzing this type of network data:

**(R.Q.2.1):** *What different techniques are best suited to infer a graph for a sensor network? Can neighboring nodes be used to reconstruct sensor measurements using the graph and signal reconstruction techniques?*

In short, we have introduced a graph-based methodology for analyzing air pollution sensor networks, where a graph is learned to describe the existing relationships between the different sensors that compose the monitoring network. In this way, we have shown the main graph inference techniques in order to reconstruct the network measurements using both supervised (MLR) and semi-supervised (Laplacian interpolation) ML models. We have shown the importance of taking advantage of the relationships between nodes in a network, where the concentrations in the neighborhood of a sensor have been shown to provide information about the concentrations in that sensor. Three graph learning methodologies have been studied. The first one, the graphical Lasso,

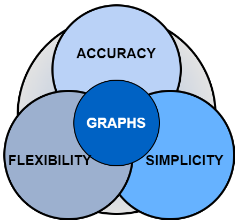


**Figure 5.10:** Graphs provide an accurate, yet flexible solution to air pollution sensor network analysis.

has been affected by the multicollinearity present in this type of network. Even so, this data-driven method has been able to obtain the sensor neighborhoods that have provided a good estimation by multiple linear regression. The method based on geodesic distances, based on prior information, has resulted in the worst performance in terms of network data reconstruction. That is because the low-cost air pollution sensor networks represent complex relationships that can be conditioned by many factors (e.g., location, emission sources, etc.) then the distances do not correctly represent the relationship between the sensors in the network. Finally, the graph signal processing method based on the smoothness of the graph signal has been the model that has obtained both the best neighborhood for the sensors and the best description of the relationships between sensors, resulting in a good reconstruction using both supervised and semi-supervised models.

As for the use of network information, through signal reconstruction, we have seen how $O_3$ and $NO_2$ can be approximated quite well using the concentrations of neighboring sensors (i.e., in nearby locations or related sensors). In addition, results obtained in the H2020 Captor Heterogeneous network have shown that by learning the sensor relationships correctly, network nodes can be reconstructed with an accuracy similar to that of a LCS. That is, if we have a reference station at a site and learn the relationships between the concentrations at that site and its neighboring sensors, they provide an estimate similar to that of a LCSs. Regarding the scalability of the framework, we have observed how graph learning can be approached cluster-wise by dividing the network into $C$ clusters to learn $C$ disjoint graphs. The bottleneck of the improvement in scalability relies on the size of the largest cluster. We have carried out an experiment where we have observed, with $C = 3$, how the average error in the signal reconstruction is very similar to using a single graph, $C = 1$, indicating how similar nodes can be grouped and dissimilar nodes do not need to participate in the signal reconstruction.

Ultimately, the GSP-based/semi-supervised methodology has been the most effective, since the smoothness-based method has been able to obtain the best graph for signal reconstruction, including the sparser solutions. The reconstruction by means of Laplacian interpolation has proved to be very practical in the case of this type of network, where any node can present missings, even the reference instrumentation. Therefore, graphs have proven to be an effective alternative for the reconstruction of network data, flexible to adapt to failures in any node of the network, and simple and interpretable.

**Graphical Lasso**

- Has been seen to suffer from multicollinearity, so that for dense graphs the results are not reliable.
- $\lambda$ hyperparameter allows for controlling the density of the obtained graphs, producing well-conditioned results.
- It has learned meaningful relationships for less dense graphs, resulting in good signal reconstruction results.

**Distance-Based**

- It has not been effective in learning relationships between air pollution network sensors since these may have complex relationships not well defined by distances.
- Sensors' neighborhoods have not been correctly discovered.

**Smoothness-Based**

- It has been able to learn relationships between significant sensors from the data. $\alpha$ and $\beta$ hyperparameters allow the inference of graphs with different degrees of connectivity and smoothness.
- It has been able to obtain sparse graphs describing correctly the relationships between sensors, being able to obtain a near-optimal signal reconstruction by means of Laplacian interpolation.
- Its performance together with the Laplacian interpolation has proven to be the most efficient model in signal reconstruction, being able to adapt to the case in which any of the nodes may present missings.

**Signal Reconstruction Performance**

- $O_3$ and $NO_2$ concentrations have been well approximated using concentrations from neighboring sensors.
- Given the learning of the correct relationships between network sensors, the results for the $O_3$ show a performance in the reconstruction of the network data similar to the low-cost sensors' performance.

**Scalability**

- The most demanding task is the graph learning optimization problem which scales quadratically with the number of network nodes $N$.
- A cluster-wise methodology can be applied to improve scalability by dividing the node set into $C$ clusters using a clustering algorithm to learn $C$ disjoint graphs.

As for future work on graph learning techniques for air pollution sensor networks, it would be interesting to study the case of mobile sensor networks where the graphs can be dynamically adapted according to the network needs to take advantage of the evolving relationships between mobile sensor measurements.

**Practical Tip !**

Right after the deployment of a sensor network, one can use a data-driven smoothness-based graph learning model to infer the graph representation of the network and benefit from sensor relationships. In the case of a large network, a cluster-based approach can be used to speed up the graph learning.

☐ *From now on, we assume that the smoothness-based graph learning method is the most suitable algorithm for this kind of network and we use this graph inferring algorithm for the rest of applications and studies.*

# 6

# Graph Signal Reconstruction Techniques for IoT Sensor Networks

*Religion is a culture of faith;*
*science is a culture of doubt.*

Richard P. Feynman

In the previous chapter, we have seen how air pollution sensor networks can be represented by graphs and how their measurements can be interpreted as signals defined over graphs. In fact, different techniques to learn a graph that represents a sensor network in order to reconstruct the measurements of the different network nodes have been tested. The next step is to explore different graph signal reconstruction (GSR) techniques that can be used to take advantage of the information from network sensors in a joint way in order to perform data quality maintenance tasks. Therefore, following the same graph-based approach, we evaluate different GSR models, coming from different fields, such as semi-supervised learning, signal processing, or kernel methods, and assess their effectiveness in the reconstruction of $O_3$, $NO_2$, and $PM_{10}$ measurements in environments where any sensor may need to be reconstructed [108, 119, 120].

This chapter is structured as follows; section 6.1 describes the need for GSR models and their applicability in air pollution sensor networks. Section 6.2 describes the different models tested and section 6.3 presents the experiments carried out. Finally, section 6.4 concludes the chapter. This chapter presents the findings made in "*Graph Signal Reconstruction Techniques for IoT Air Pollution Monitoring Platforms*", IEEE IoT-J, [82].

## 6.1. Graph signal reconstruction for sensor networks

Graphs have proven to be a good tool to represent the sensor network measurements and enable the interpretation of network measurements as signals defined over graphs [21, 100]. In this way, by means of a network topology, it is possible to take advantage of information from similar sensors in order to flow information from the network to the sensors of interest. The topology corresponds to the first cornerstone of the graph signal processing-based framework, where the relationships between sensors and the neighborhoods of each one of them are defined. The second cornerstone is the signal reconstruction model, in this case, we refer to GSR since it allows for taking advantage of the information defined by the graph and the network measurements to reconstruct sensor measurements.

There are a large number of direct applications for generating a graph with reference stations and LCSs and overlaying a signal reconstruction mechanism in which several of the network nodes participate. The principle behind this technique is to obtain the relationships between the network sensors by means of a graph learned from the data, resulting in a smooth structure with respect to the measured data. So, the data measured by the different sensors can be used by signal reconstruction methods to obtain estimates and maintain the quality of the network data, e.g., the imputation of missing values. It is known that sensors can be calibrated with supervised ML methods using arrays of sensors [15, 27, 78]. If one of the array measurements is missing, the concentration of the pollutant cannot be estimated and a gap in the measurements occurs [121, 122]. This measurement can be estimated using sensor neighbors that measure the same phenomenon and that is highly correlated with the data from the sensor that has the missing value, i.e., like a spatial interpolation. Other examples where signal reconstruction methods can be applied overlaid on a graph constructed from the data are sensor drifts [18, 123], the creation of virtual sensors [124] and the creation of proxies [69, 121, 122]. The deterioration of the data quality of these network sensors can be mitigated if during the network deployment, a sensor's signal is reconstructed using other sensors, which are potentially non-problematic, and are highly correlated with the problematic sensor. Hence, GSR for air pollution sensor networks is key to harnessing network information and maintaining data quality, where unobserved nodes (e.g., sensors with missing samples, drifting sensors, etc.) are reconstructed from a subset of observed nodes in the graph, also including places where there are no physical sensors (virtual sensors) [121, 124].

Virtual sensors are nodes in which it is difficult to deploy a physical sensor, and in which the value of the pollutant is estimated from values in the vicinity [125]. Thus, GSR can also help maintain network data quality by obtaining virtual sensor estimates when any of the nodes are under maintenance or have been relocated. This application is studied in detail in the next chapter 7.

All in all, *signal reconstruction* is, therefore, a key technique to benefit from the measurements of similar sensors, including reference stations, which can correct data for multiple applications. Indeed, this signal reconstruction can be seen as a form of spatial interpolation at the precise location of the nodes when nodes do not coincide in the same locations. Thus, a node's signal is reconstructed using other spatially distributed sensors' measurements.

### 6.1.1. Graph signal reconstruction: State Of The Art

There is a wide variety of GSR models coming from different fields, in fact, we can define GSR as a model that uses the observed network measurements and the network topology described by the graph to complete the unobserved measurements. Indeed, there are methods from the field of *semi-supervised learning* such as the Laplacian interpolation (interpolated regularization from Belkin *et al.* [108]) that use the Laplacian of the graph

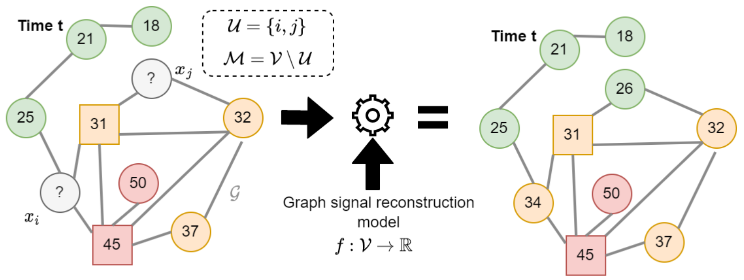

Time t
21
18
?
$x_j$
$\mathcal{U} = \{i, j\}$
$\mathcal{M} = \mathcal{V} \setminus \mathcal{U}$
25
31
32
?
50
$x_i$
$\mathcal{G}$
45
37
Graph signal reconstruction model
$f : \mathcal{V} \to \mathbb{R}$
Time t
21
18
26
25
31
32
34
50
45
37

**Figure 6.1:** Graph signal reconstruction setting, where at a given time instant $t$ a graph signal $\mathbf{x}$ defined over a learned graph topology $\mathcal{G}$ presents gaps, unobserved nodes $\mathcal{U}$, and these gaps are completed using a graph signal reconstruction model $f(\cdot)$.

to extrapolate the observed values to those that are not, maximizing the smoothness of the resulting signal. Other techniques, based on *signal processing*, use the Fourier transform assuming certain signal's transform support to recover the full signal given some measurements [119]. Some other works, address the GSR task from the *kernel* point of view, kernelizing the signal reconstruction estimator and obtaining the kernel ridge regression (KRR) [120]. Moreover, matrix completion methods are a well-known family of methods, which are transductive, that complete the missing entries of the data matrix assuming that it has a lower rank representation; e.g. the kernelized probabilistic matrix factorization (KPMF) [126] that uses graph kernels to factor the matrix. In addition, another recent field that has benefited from graph signal processing (GSP) tools are graph neural networks (GNN). Here, the definition of the convolution operator through graph filtering operations allows the development of convolutional neural networks in irregular domains such as graphs. Among the neural networks developed we find ChebyNet [127], which approximates convolution filtering through a Chebyshev polynomial filter, or the inductive graph neural network Kriging (IGNNK) [128], a 2-layer diffusion convolution neural network that allows reconstructing any of the nodes through a subnetwork selection scheme during the training. However, the two last techniques need to be further developed given that matrix completion methods do not naturally fit into the problem of signal reconstruction, as for each new sample they need to be retrained. Besides, there does not exist much literature about graph neural network architectures for GSR in cases where data is limited. We restrict the comparison of reconstruction methods to linear models since nonconvex models usually require more data, which are often unavailable in LCS deployment environments. Moreover, nonconvex models are more difficult to adapt to environments where any subset of nodes can have missing data.

### 6.1.2. GRAPH SIGNAL RECONSTRUCTION IN AIR POLLUTION SENSOR NETWORKS

Recently, GSR models have been applied to data from particulate matter (PM) sensor networks [129, 130]. The matrix completion problem has also been tackled using variational graph autoencoders (VGAE) applied to $NO_2$ and PM data sets [131]. Graph convolutional recurrent neural networks have also been used for PM data [132]. However, as mentioned above, the matrix completion approach to signal reconstruction, or the need for graph neural networks for large amounts of training data, can make it difficult to use these models in LCS network deployments with limited data. This remains an open field of research as there is little work on signal reconstruction in sensor networks, which may have their own data requirements, computational requirements, as well as the need for real-time prediction for monitoring applications. Furthermore, in most cases, GSR models for air pollution are applied to graphs created from the geodesic distances of the nodes [129, 131, 132]. However, in the previous chapter 5 we have evaluated the effectiveness of different graph learning techniques, where graphs learned from data in air pollution sensor networks resulted to be most precise technique given the complexity of air pollution correlations between different sites as well as the variability of LCSs. Thus, we chose to learn the graph from the data upon which we apply different GSR techniques.

### 6.1.3. Graph signal reconstruction scenarios & Motivation

Regarding the use of GSR models for structured measurements of air pollution monitoring sensor networks, we can differentiate two scenarios in the signal reconstruction, which we name *supervised* and *semi-supervised*:

A) **Supervised scenario** or the scenario where a subset of network nodes are to be predicted from a fixed set of observed nodes. In this case, a GSR model can be trained as if it were a supervised machine learning model. Moreover, supervised ML models could be used if no loses are assumed.

B) **Semi-supervised scenario** or the scenario in which any node may have missing data and consequently the subset of unobserved nodes or nodes to be reconstructed may vary from one time instant to another. Hence, the reconstruction model has to be flexible and transductive, in order to deal with possible missings and a variable set of observed nodes. This scenario is the one studied in this chapter because of its importance in LCS networks, where these nodes are prone to failures, and reconstruction of any node may be required.

To sum up, in this chapter, we opt for the use of linear GSR models superimposed on a graph learned from the data, given their low data and low computational requirements. This allows extending the use of these techniques to both sensor deployment environments where the data may be scarce, and to environments where any node could fail.

## 6.2. Graph signal reconstruction models

In this section, we describe the different GSP models studied in this chapter, but first we briefly review the notation necessary to understand these models. We shall define the graph upon which we will apply the different models as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$. We learn the Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N\times N}$ by means of the smoothness-based optimization problem defined in [104]. We recall from the previous chapter how the smoothness-based model has turned out to be the best graph learning model for this type of sensor network.

The goal of graph learning is to obtain the implicit relationships between sensors that form the air pollution monitoring network so that the graph can be used later to flow information between similar sensors.

Once we have defined the graph $\mathcal{G}$ on which the signal reconstruction models will be applied, we can define the purpose of using signal reconstruction. *Signal reconstruction* can be used to maintain the data quality of a network. Estimates can be obtained for a sensor that may present problems (e.g., missing value, sensor malfunction, drift, etc.) using information from its neighboring sensors. In addition, in a worst-case scenario, such estimation can also be performed when information is partial, i.e., there is more than one sensor whose signal needs to be estimated or presents missing data.

Given the set of network nodes $\mathcal{V} = \{1, \ldots, N\}$, where $N$ is the number of nodes/sensors, we consider the problem of having a subset of observed nodes $\mathcal{M} \subseteq \mathcal{V}$, for which their corresponding measures are available, being $|\mathcal{M}| = M$ the number of observed nodes. The aim is to estimate the subset of unobserved nodes $\mathcal{U} = \mathcal{V} \setminus \mathcal{M}$ using a GSR model that regresses the function $f : \mathbb{R}^M \to \mathbb{R}^U$, where $|\mathcal{U}| = U$ is the number of unobserved nodes, assuming a certain structure for the regression function $f(\cdot)$. Depending on the regression criterion the resulting graph signal $\hat{\mathbf{x}} \in \mathbb{R}^N$ will minimize the smoothness of the signal with respect to the graph $\mathrm{TV}(\hat{\mathbf{x}}, \mathbf{L})$ or minimize the *mean squared error*, among others. Now, we enumerate and describe the different GSR models analyzed:

1) ***Laplacian interpolation***: also known as graph interpolated regularization[1] by Belkin *et al.* [108]. This model estimates the set of unobserved measures $\mathbf{x}_{\mathcal{U}} = \{x_u : u \in \mathcal{U}\}$ by minimizing the quadratic form of the Laplacian given the set of observed measures $\mathbf{x}_{\mathcal{M}} = \{x_m : m \in \mathcal{M}\}$ so that the resulting graph signal $\hat{\mathbf{x}}$ is smooth with respect to the Laplacian $\mathbf{L}$. The optimization criterion is defined as:

$$\begin{aligned} \min_{\hat{\mathbf{x}} \in \mathbb{R}^N} \quad & \hat{\mathbf{x}}^\top \mathbf{L} \hat{\mathbf{x}} \\ \text{s.t.} \quad & \hat{x}_m = x_m, \quad \forall m \in \mathcal{M}, \end{aligned} \tag{6.1}$$

The estimator above corresponds to the linear combination of the neighbors' measurements, $\mathbf{x}_{\mathcal{N}(x_u)}$, of the target sensor $x_u$ weighted by the values of the Laplacian $\mathbf{L}_{u\mathcal{N}(x_u)}$. For further details about this model refer to the previous chapter 5.

2) ***GSP low-pass reconstruction***: in the field of signal processing, sparse signal recovery using an orthogonal transform is a common technique [119]. Precisely, an orthogonal transform, e.g., the discrete Fourier transform (DFT), is usually used assuming that a signal $\mathbf{x}$ is bandlimited in that transform domain and that the support of the transform is known and low-pass. This can be translated to graph signals where the notion of frequency defined for the graph discrete Fourier transform (GDFT) can be used [22]. Therefore, this technique recovers the set of unobserved values $\mathbf{x}_{\mathcal{U}}$ assuming that the GDFT coefficient vector of the signal $\mathbf{x}$ is sparse and of low-pass nature, meaning that it has $K$ non-zero components corresponding to the $K$ lowest frequencies (smallest eigenvalues $\lambda_i$ of $\mathbf{L}$). Recall that the Laplacian matrix admits the eigendecomposition $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^{-1}$, the GDFT of a graph signal $\mathbf{x}$ can be computed as:

$$\mathbf{X} = \mathbf{U}^{-1}\mathbf{x} \tag{6.2}$$

[1] Lately, this problem has been redefined within the graph signal processing context by replacing the Laplacian matrix by a graph shift operator [113].

Now, a $K$-sparse GDFT coefficient vector of the following form is to be recovered:

$$\mathbf{X} = [X_0, \dots, X_{K-1}, 0, \dots, 0]^\mathsf{T} \tag{6.3}$$

For this purpose a subset of measurements $\mathbf{x}_\mathcal{M}$ are used to recover the sparse coefficient vector by solving the following system:

$$\mathbf{x}_\mathcal{M} = \mathbf{U}_{\mathcal{M}K}\mathbf{X}_K \tag{6.4}$$

Depending on whether $M > K$ or $M < K$, eq. (6.4) corresponds to an overdetermined or undetermined system of equations. Generally, $M > K$, the system is overdetermined so the solution of the above system in the least squares sense is given by $\mathbf{X}_K = \mathbf{U}^\dagger_{\mathcal{M}K}\mathbf{x}_\mathcal{M}$, where $\mathbf{U}^\dagger_{\mathcal{M}K} = (\mathbf{U}^\mathsf{T}_{\mathcal{M}K}\mathbf{U}_{\mathcal{M}K})^{-1}\mathbf{U}^\mathsf{T}_{\mathcal{M}K}$ is the matrix pseudo-inverse of $\mathbf{U}_{\mathcal{M}K}$. The nonzero coefficients are obtained this way, and after appending the corresponding zero coefficients, the inverse graph discrete Fourier transform (IGDFT) $\hat{\mathbf{x}} = \mathbf{U}\mathbf{X}$ is computed to obtain the complete set of measurements $\hat{\mathbf{x}} \in \mathbb{R}^N$ at all vertices $i \in \mathcal{V}$. For the case, $M < K$ the solution is computed using the right pseudoinverse, while for $M = K$ the solution is computed by matrix inversion.

*3)* ***Kernelized graph signal reconstruction***: Romero *et al.* [120] introduced the kernelized ridge GSR. As in the case of classical linear regression, given a set of noisy observations $\{x_m = g(m) + \epsilon_m : \forall m \in \mathcal{M}\}$, the kernel regression estimates the underlying function $g : \mathcal{V} \to \mathbb{R}$ in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, which is a space of functions $f : \mathcal{V} \to \mathbb{R}$:

$$\mathcal{H} = \left\{ f : f(i) = \sum_{n=1}^{N} \alpha_n k(i,n), \alpha_n \in \mathbb{R} \right\} \tag{6.5}$$

Where $k : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is a graph kernel map that defines some similarity between nodes. After some manipulation and the application of the *representer theorem*, which states that the solution can be expressed as a linear combination of the kernel map values of the observed nodes $\hat{g}(i) = \sum_{m \in \mathcal{M}} \hat{\alpha}_m k(i,m)$. The kernel ridge regression (KRR) problem is defined as:

$$\hat{\boldsymbol{\alpha}}_\mathcal{M} = \underset{\hat{\alpha}_\mathcal{M} \in \mathbb{R}^M}{\arg\min} \underbrace{\frac{1}{M} \|\mathbf{x}_\mathcal{M} - \mathbf{K}_{\mathcal{M}\mathcal{M}}\boldsymbol{\alpha}_\mathcal{M}\|^2}_{MSE} + \mu \underbrace{\boldsymbol{\alpha}^\mathsf{T}_\mathcal{M}\mathbf{K}_{\mathcal{M}\mathcal{M}}\boldsymbol{\alpha}_\mathcal{M}}_{RKHS\ norm} \tag{6.6}$$

Where the mean squared error (MSE) is used as the loss function, the regularization term is the RKHS norm of the solution $g(\cdot)$. This problem has a closed-form solution:

$$\hat{\boldsymbol{\alpha}}_\mathcal{M} = (\mathbf{K}_{\mathcal{M}\mathcal{M}} + \mu M \mathbf{I}_M)^{-1}\mathbf{y}_\mathcal{M} \tag{6.7}$$

The key to the successful application of kernel-based GSR is in kernel selection. Thus, based on the assumption that the signal evolves smoothly over the graph, graph kernels that capture such prior information can be used, a common choice is the diffusion kernel (KRR-DIFF) where $r(\lambda) = e^{\sigma^2\lambda/2}2$:

$$\mathbf{K} = r^{\dagger}(\mathbf{L}) = \mathbf{U} r^{\dagger}(\Lambda)\mathbf{U}^{-1} \tag{6.8}$$

The vertex-covariance kernel (KRR-COV) can also be used, which is based on the covariance instead of graph structure, which turns out to be the local linear minimum mean squared error (LMMSE) estimator on a Markov random field (MRF) [120]. Since the actual covariance matrix $\mathbf{\Sigma}$ is unknown, the graphical lasso has been used to estimate the covariance matrix $\hat{\Sigma}$ throughout the experiments. Given the presence of multicollinearity in air pollution data (see chapter 5) and that the covariance matrix $\mathbf{\Sigma}$ has to be estimated, the result is suboptimal, i.e., $\mathrm{MSE}(\hat{\mathbf{\Sigma}}, \hat{\mathbf{x}}) \geq \mathrm{MSE}(\mathbf{\Sigma}, \hat{\mathbf{x}})$.

**Definition 14** *A linear minimum mean squared error (LMMSE) estimator is an estimator minimizing the MSE with the strict form of a linear estimator. This means, given two random variables $y$ and $x$ the conditional expectation can be expressed as $\mathrm{E}[y|x] = \beta_0 + \beta x$, where the estimator $\beta, \beta_0$ is LMMSE if it minimizes the MSE.*

Given the analytical form of the solution for the methods described above, it can be noticed how the reconstruction of any of the methods corresponds to a linear combination $\mathbf{x}_{\mathcal{U}} = \boldsymbol{\beta}\mathbf{x}_{\mathcal{M}}$ of the observed nodes $\{x_m \colon m \in \mathcal{M}\}$. Hence, Table 6.1 shows how the $\boldsymbol{\beta}$ coefficients are calculated for the different reconstruction models. As it can be seen, the main operations correspond to a matrix inversion and multiplication; in the case of Laplacian interpolation (Lap.Int.) the $\boldsymbol{\beta}$ are calculated by the Laplacian $\mathbf{L}$, in the case of GSP by the Laplacian eigenvector matrix $\mathbf{U}$, in the case of the kernelized ridge regression with diffusion kernel (KRR-DIFF) by a pre-computed kernel matrix $\mathbf{K}$, and in the case of the kernelized ridge regression with the vertex-covariance kernel (KRR-COV) by the data covariance matrix $\hat{\mathbf{\Sigma}}$.

**Table 6.1:** Equivalence of the models to a linear model. $\mathbf{x}_{\mathcal{U}}$ are the unobserved values and $\mathbf{x}_{\mathcal{M}}$ are the observed ones.

| **Model:** | $\mathbf{x}_{\mathcal{U}} = \boldsymbol{\beta}\mathbf{x}_{\mathcal{M}}$ |
|---|---|
| Lap.Int | $\boldsymbol{\beta} = -\mathbf{L}^{-1}_{\mathcal{U}\mathcal{U}}\mathbf{L}_{\mathcal{U}\mathcal{M}}$ |
| GSP | $\boldsymbol{\beta} = \mathbf{U}_{\mathcal{U}K}(\mathbf{U}^{\top}_{\mathcal{M}K}\mathbf{U}_{\mathcal{M}K})^{-1}\mathbf{U}^{\top}_{\mathcal{M}K}$ |
| KRR-DIFF | $\boldsymbol{\beta} = \mathbf{K}_{\mathcal{U}.}\mathbf{\Phi}^{\top}(\mathbf{\Phi}\mathbf{K}\mathbf{\Phi}^{\top} + \mu M\mathbf{I}_M)^{-1}$ |
| KRR-COV | $\boldsymbol{\beta} = \hat{\mathbf{\Sigma}}_{\mathcal{U}.}\mathbf{\Phi}^{\top}(\mathbf{\Phi}\hat{\mathbf{\Sigma}}\mathbf{\Phi}^{\top} + \mu M\mathbf{I}_M)^{-1}$ |

As it can be seen in the table above, the different models can be computed using linear algebra, which makes them computationally tractable for the *semi-supervised* scenario studied in this thesis, where the set the unobserved nodes $\mathcal{U}$ may change from time to time[3].

[2]There exist other graph kernels such as the Laplacian kernel ($r(\lambda) = 1 + \sigma^2\lambda$), the diffusion kernel ($r(\lambda) = e^{\sigma^2\lambda/2}$) or the p-step random kernel ($r(\lambda) = (a - \lambda)^{-p}, a \geq 2$).

[3]A python library implementing the methods mentioned above is available at https://bitbucket.org/sans-rg/ieee-graph-signal-reconstruction/.

## 6.3. Experimental evaluation

In this section, the different GSR models described in the previous section 6.2 are evaluated in a semi-supervised environment, where any node may need to be reconstructed. More precisely, in this section we: *i)* describe the data sets used in the subsequent experiments, *ii)* describe the methodology to learn the graph, as well as the results of learning the graph depending on the GSR model used, *iii)* experiment with the individual reconstruction of each node of the network ($U = 1$), and *iv)* experiment with the reconstruction of random subsets of nodes of increasing size ($U>1$), simulating different simultaneous reconstructions on different nodes, i.e., simulating unavailable nodes.

In order to carry out the different experiments, i.e., train the GSR model and learn the graph, we proceed as follows; *i)* we use 100% of the data set for cross-validation (CV), *ii)* we perform a grid search over the graph learning hyperparameters and the signal reconstruction hyperparameters via 5-fold CV, *iii)* in each of the folds we learn the graph using the training data and we reconstruct each one of the nodes in the validation set and report the average root-mean-squared error (RMSE) between all nodes, and *iv)* report the CV metrics for the best-performing hyperparameters and graph topology. From now on, all metrics reported throughout the chapter are CV metrics.

### 6.3.1. Data sets

This section introduces the data sets used for the comparison of the GSR methods described above. Nowadays, there are many open data initiatives to promote transparency and encourage research. The Spanish government carries out the measurement of air pollution levels by means of reference stations, which are worth thousands of euros given their high accuracy, and makes such data openly available. Hence, data captured by reference stations[4] in the area of Catalonia, Spain, over an area of 32,108 km$^2$ have been selected for three pollutants; tropospheric ozone ($O_3$), nitrogen dioxide ($NO_2$) and particulate matter 10 ($PM_{10}$). In this chapter, unlike the previous chapters, with the exception of the scalability experiment in the previous chapter 5, we use a reference station network deployed over a large area, such as the area of Catalonia, Spain. This allows us to evaluate the reconstruction capability of the different instruments depending on their location and the number of neighboring sensors.

**Table 6.2:** Summary of the data sets used to carry out the different experiments throughout the chapter.

| **Data Set** | **Pollutant** | **# Nodes** | **# Samples** | **Period** | **Mean** ($\mu$gr/m$^3$) | **Pooled STD.** ($\mu$gr/m$^3$) |
|---|---|---|---|---|---|---|
| 1 | $O_3$ | 46 | 1155 | 2019/01/01 - 2019/05/31 | 66.84 | 28.61 |
| 2 | $NO_2$ | 60 | 983 | 2019/01/01 - 2019/05/31 | 23.46 | 16.52 |
| 3 | $PM_{10}$ | 33 | 709 | 2019/01/01 - 2019/05/31 | 20.11 | 11.51 |
| 4 | $O_3$ | 8 | 2612 | 2017/06/18 - 2017/09/16 | 64.92 | 34.84 |

These pollutants exhibit different spatial behavior, which allows for studying the signal reconstruction under various spatial conditions. In addition, the use of reference stations over a large area makes it possible to investigate the feasibility of building a large network and the application of these methods. Table 6.2 shows the characteristics of the

[4]These data are available at `http://mediambient.gencat.cat/ca/05_ambits_dactuacio/atmosfera/qualitat_de_laire/vols-saber-que-respires/descarrega-de-dades/`.

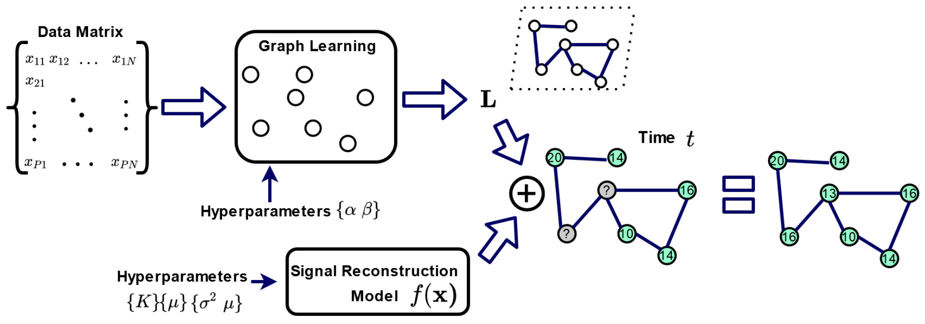

Data Matrix
$x_{11}\ x_{12}\ \dots\ x_{1N}$
$x_{21}$
$x_{P1}\ \dots\ x_{PN}$
Graph Learning
L
Hyperparameters $\{\alpha\ \beta\}$
Time $t$
20
14
?
16
?
10
14
20
14
13
16
16
10
14
Hyperparameters
$\{K\}\{\mu\}\{\sigma^2\ \mu\}$
Signal Reconstruction Model $f(\mathbf{x})$

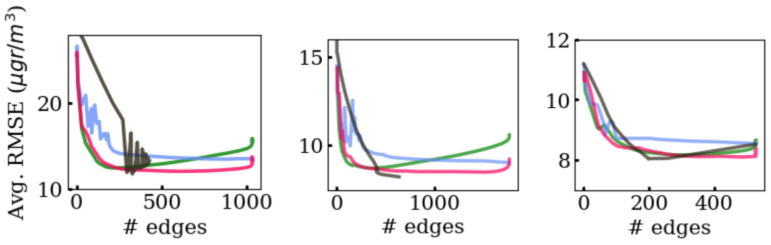
**Figure 6.2:** Graph learning and GSR pipeline. The graph learning model and the GSR model may have different hyperparameters. Both the graph and the signal reconstruction model can be used to recover an incomplete graph signal ($U \geq 1$).

data sets used, and Figure 6.4 presents the location of the reference stations for the data sets. Moreover, Table 6.2 shows the means for the pollutants as well as the pooled standard deviation to better interpret the error measures in the following sections. These three data sets will be used to show the ability to reconstruct signals in an air pollution monitoring network in a regional area such as the metropolitan area of Barcelona, Spain.

### 6.3.2. LEARNING THE GRAPH

Figure 6.2 shows the pipeline for the graph learning and posterior GSR of any graph signal. First, given a training data matrix $\mathbf{X_{tr}}$, corresponding to a set of training graph signals, and the graph learning method with its hyperparameters $\alpha$ and $\beta$, we get the Laplacian matrix $\mathbf{L}$ that describes the relationships between the nodes. Then, given a GSR model and its hyperparameters - none for the Laplacian interpolation, $K$ for the GSP, $\mu$ and $\sigma^2$ for KRR-DIFF and $\mu$ for KRR-COV - we can reconstruct a graph signal at any time $t$ given a subset of observed nodes $\mathcal{M}$. Actually, this is the methodology performed during the CV, where the graph $\mathcal{G}$ is learned using the training set of that fold and the nodes in the validation set, of the same fold, are reconstructed. The four GSR models are transductive, it is to say, the coefficients for the reconstruction can be calculated given a set of observed nodes to reconstruct the unobserved ones, but if the set of observed nodes changes (e.g. some nodes have missings) the model needs to be recalculated to reconstruct the new set of unobserved nodes. It is important to note that the different GSR models have different hyperparameters, structures, and objective functions, so it is possible that the graph selection depends on the signal reconstruction model, making its selection linked.

There are then two blocks of hyperparameters; one corresponds to graph learning and the other to signal reconstruction. Therefore, it is possible that for a given graph, with a certain number of edges, two methods perform in a different way. For this reason, CV is done with all the hyperparameters, both for the graph learning and for the signal reconstruction, since as mentioned before the performance of the reconstruction method can be associated with the obtained graph. To illustrate this, Figure 6.3 shows the average CV RMSE for each of the methods given the graph obtained. That is, the average RMSE of CV for each $\alpha$ and $\beta$ and the hyperparameters of the signal reconstruction method that have obtained the lowest error (the best $K$ for GSP, the best $\mu$ and $\sigma^2$ for KRR-DIFF and the best $\mu$ for KRR-COV). The case of the kernelized ridge regression with the vertex-covariance kernel is a special case, where the covariance matrix is estimated using the graphical Lasso algorithm with a different $\lambda$ hyperparameter values, thus obtaining an adjacency matrix (given the precision matrix $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$) but not a Laplacian matrix.



(a) $O_3$ data set. (b) $NO_2$ data set. (c) $PM_{10}$ data set.

**Figure 6.3:** Average CV RMSE obtained for graphs with a different number of edges and the best signal reconstruction hyperparameters. The line in green corresponds to the Lap.Int., the blue line corresponds to GSP, and the red and black lines correspond to KRR-DIFF and KRR-COV respectively.

The same trends can be seen in the three plots in Figure 6.3. The Laplacian interpolation gets its best error for a low number of edges, increasing later as the number of edges in the graph grows. The KRR-DIFF finds its lowest RMSE, similar to that of the Laplacian interpolation, but with a larger number of edges. The GSP reconstruction shows great instability for very sparse graphs, but its best error is obtained as the density of the graph increases. Finally, the case of KRR-COV shows great instability due to the multicollinearity present in the data, as mentioned in the previous chapter 5, in many cases, it is not possible to obtain the complete graph with the graphical Lasso due to this problem since the matrix that is the object of manipulations may be ill-posed. Also for similar graphs, the error can vary a lot, but the best error seems to be obtained with 50% of the edges. So, for a low number of edges, the Laplacian interpolation and KRR-DIFF are the best, the KRR-COV seems to obtain its best error with a relatively small number of edges, and the low-pass-based reconstruction needs a large number of edges (around 100%) to obtain its best performance. Moreover, the KRR-DIFF is the most stable method, being able to maintain a good performance independently of the graph, its number of hyperparameters favor this effect being able to adapt better. On the other hand, the GSP and the KRR-COV are the ones that show more variability for sparse graphs and the Laplacian interpolation works the best for sparse graphs. Although the KRR-COV is the best linear estimator for all three data sets, Figure 6.3 shows how for sparse graphs (<25% of the edges) the other methods produce a lower error. This is because the smoothness-based graph learning method is able to find a better set of neighbors per node for signal reconstruction than graphical Lasso, so although KRR-COV is optimal given a dense graph, it is not able to obtain the best performance for sparse graphs. In summary, each method behaves differently for a graph, so the graph learning task must be coupled with the signal

**Table 6.3:** CV performance metrics for the different air pollutant data sets and reconstruction methods.

| Method | $O_3$ | | | | $NO_2$ | | | | $PM_{10}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMSE | MAE | $R^2$ | # Edges | RMSE | MAE | $R^2$ | # Edges | RMSE | MAE | $R^2$ | # Edges |
| **Lap.Int.** | 12.41 | 9.51 | 0.66 | 216.80 | 8.72 | 6.44 | 0.42 | 369.80 | 8.16 | 5.66 | 0.26 | 297.60 |
| **GSP** | 13.43 | 10.44 | 0.56 | 1035.00 | 9.03 | 6.73 | 0.26 | 1761.20 | 8.54 | 5.99 | 0.16 | 528.00 |
| **KRR - DIFF** | 12.02 | 9.21 | 0.69 | 614.40 | 8.49 | 6.27 | 0.46 | 1514.00 | 8.10 | 5.61 | 0.29 | 481.80 |
| **KRR - COV** | 11.72 | 8.98 | 0.71 | 328.80 | 8.23 | 6.04 | 0.50 | 638.40 | 8.03 | 5.51 | 0.30 | 198.00 |

reconstruction model in the CV procedure.

## 6.3.3. SIGNAL RECONSTRUCTION PERFORMANCE: U=1

Let us see the result of learning the graph and performing signal reconstruction given that the data of all node neighbors are available, i.e., the number of unobserved nodes $|\mathcal{U}| = U = 1$ is one for each reconstruction. The average CV metrics used are the average metrics for the reconstruction of each one of the network nodes. Table 6.3 shows the minimum average CV RMSE between stations, the average CV mean absolute error (MAE), as well as the corresponding average CV coefficient of determination $R^2$, and the average number of edges for the data sets and signal reconstruction methods. The methods appear to perform similarly as in the previous experiment, with KRR-COV being the best and GSP the worst. For the $O_3$, we can observe how the GSP is the worst method obtaining an $R^2$ of 0.56 with a complete graph (1035 edges) and the KRR with vertex-covariance kernel is the best model obtaining $R^2$ of 0.71 and 328.8 edges on average, followed by the KRR-DIFF that obtains a similar performance with a denser graph (614.40 edges on average) and the Laplacian interpolation whose best performance is obtained with a sparse graph. The same trend is observed for the $NO_2$ where the KRR-COV obtains the best performance with an $R^2$ of 0.50 and 638.40 edges on average and followed by the KRR-DIFF and Laplacian interpolation that obtain a denser and a sparser graph respectively. Finally, the reconstruction methods applied to the $PM_{10}$ data set obtain the worst results with $R^2$s around 0.16-0.3, but this time the KRR-COV performs the best with an average $R^2$ of 0.30, and the sparsest graph with 198.0 edges on average. As it has been observed, the Laplacian interpolation obtains a good performance with a sparse graph (approximately density of 25%) and the KRR-COV with a slightly denser graph, but although the KRR-DIFF obtains its minimum with a dense graph, in figure 6.3, it can be observed how it obtains a similar performance for sparse graphs. Thus, the three methods work well with sparse graphs, being the Lap.Int. and the KRR-DIFF the ones that work better with sparse graphs.

$O_3$ and $NO_2$ can be estimated quite well by reconstructing the graph signal, but $PM_{10}$ can not. The reason is that $PM_{10}$ is more heterogeneous in the area of study than $O_3$ and $NO_2$, which makes it difficult to estimate using neighboring nodes. This means that $PM_{10}$ hotspots can appear instantaneously in one exact location with little repercussion in nearby locations, requiring denser sensor networks to have related sensors. Among the methods, the KRR-COV obtains the best results as it can be interpreted as the local LMMSE estimator on a Markov random field. Nevertheless, as the covariance matrix is estimated and the data suffer from multicollinearity, the result is suboptimal and the graphical Lasso has not been able to obtain the covariance matrix with all dependencies (100% of the edges). However, the KRR-DIFF performs almost as well as KRR-COV, followed by Laplacian interpolation which also shows good results.

The Laplacian interpolation has been able to obtain an error close to the optimal local linear estimator (KRR-COV) since the smoothness of the Laplacian matrix with respect to the training data is a criterion of the graph learning optimization problem. As for the resulting graphs, the GSP method obtains the best performance with 100% of the edges. On the other hand, the Laplacian interpolation obtains its best performance with 21% of the edges (with respect to the complete graph) for $O_3$, 21% of the edges for $NO_2$ and 56% of the edges for $PM_{10}$. This is an interesting result since these methods allow obtaining a low RMSE with a small number of edges, which makes the graph sparse when the number of nodes is increased. Furthermore, in the previous section 6.3.2, it has been observed that the vertex-covariance method is no longer the best model for sparse graphs, with approximately less than 25% of the edges.

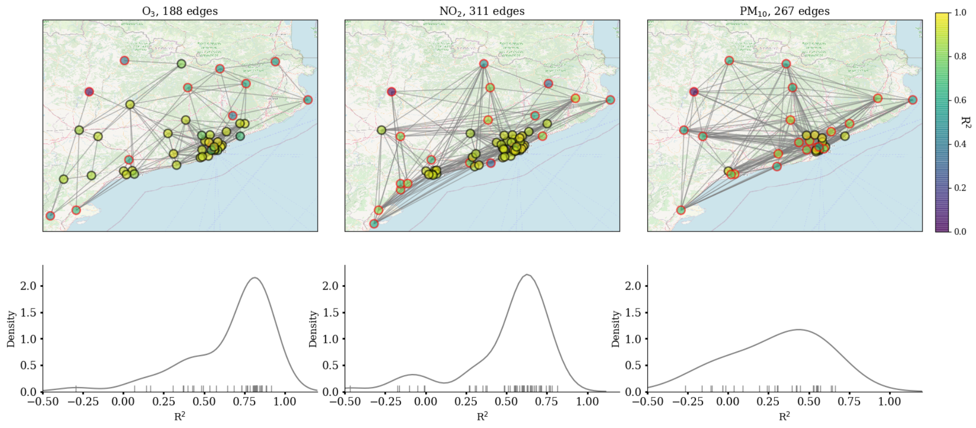


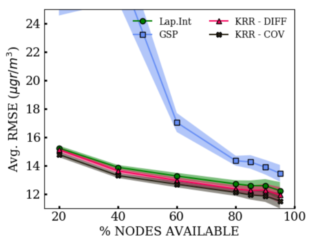
**Figure 6.4:** Plots representing the CV result for the data sets using the Laplacian interpolation method. Top figures show the graphs obtained with node color denoting the average CV $R^2$ for each station. The nodes highlighted in red denote reference stations with a CV $R^2$ below 0.5. The bottom figures show the empirical $R^2$ distribution of the graphs above.

Figure 6.4 shows the graphs obtained by the CV procedure using the Laplacian interpolation method as the reconstruction method. The color of the nodes denotes the average CV $R^2$ of the given stations. As shown in Table 6.3, despite having an average $R^2$ of 0.66, many of the stations in areas with a high density of stations (e.g. Barcelona area) have an $R^2$ around 0.8. Thus, ozone values in denser areas of similar reference stations can be estimated effectively. The same is observed for $NO_2$, although it presents a lower spatial correlation, some reference stations obtain a coefficient of determination larger than 0.7. Finally, $PM_{10}$ can not be approximated quite well using similar reference stations as most stations get an $R^2$ smaller than 0.6.

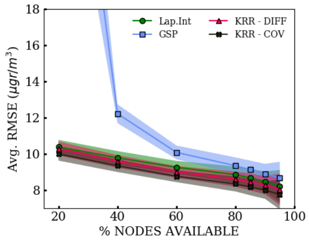
The results show a good prediction for most of the stations. Figure 6.4 shows the location of the nodes, some of them are far away from others or far from zones with a density of stations. In the same figure, we can observe the reference stations highlighted in red, which are those with a CV $R^2$ smaller than 0.5. In the case of $O_3$, these are the most distant stations, and only 10 out of 46. In the case of $NO_2$, they already represent a larger percentage of the network stations with 19 stations out of 60, and are more distributed. Finally, in the case of $PM_{10}$ the bad reconstructed stations represent the majority, 22 stations out of 33. This represents the idea that $O_3$ and $NO_2$ are more predictable due to their homogeneity. This result indicates that if we look at the individual RMSE of the stations instead of the global one, those stations that are in a high-density area achieve a high $R^2$ and, therefore, a good estimate, while those that are more distant and, therefore, do not have many neighbors, do not take advantage of the nodes of the network as much. As an example, if we considered as outliers the stations with low $R^2$ and kept them out of the calculation, the average $R^2$ with the KRR-DIFF would improve from 0.69 to 0.78 for $O_3$ and from 0.46 to 0.65 in the case of $NO_2$.

### 6.3.4. SIGNAL RECONSTRUCTION PERFORMANCE: U>1

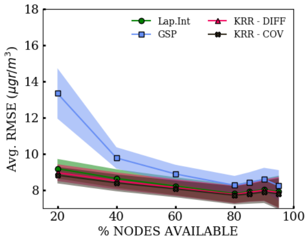
The previous section 6.3.3 showed the CV error for the reconstruction of the signal of a reference station ($U = 1$) when the others were available. In this section, we experiment with a semi-supervised setting, estimating multiple nodes at the same time ($U$>1), since we want to reconstruct several stations at once or we want to reconstruct some nodes that have missing data. This scenario is of special interest in the air pollution monitoring paradigm since in a heterogeneous network there will always exist malfunctioning nodes, nodes with data losses, nodes in maintenance, and even the presence of virtual sensors. That is why the GSR model needs to be flexible to couple with the absence of data in neighboring sensors and estimate the signal in all nodes given a learned graph topology. Given a fixed set of hyperparameters, those found in the previous section 6.3.3, we calculate the CV error as done in the previous section, but instead of reconstructing one node, selecting a random incremental percentage of nodes to be estimated simultaneously, as if they were virtual sensors during the validation set, and perform ten repetitions.



(a) $O_3$ data set.



(b) $NO_2$ data set.



(c) $PM_{10}$ data set.

**Figure 6.5:** Average CV error and its 95% confidence interval calculated as a $t$-student for several percentage of observed nodes $M$ and ten repetitions.

Figure 6.5 shows the average CV error with bands indicating its 95% confidence interval computed as a $t$-student for the different signal reconstruction methods and data sets. Figure 6.5.a) shows the results for the $O_3$ data set. First of all, we can observe how the GSP low-pass based reconstruction is the method that has the highest error with a

95% of nodes available with 13.45 $\mu$gr/m$^3$ followed by a large difference by the Laplacian interpolation method with 12.22 $\mu$gr/m$^3$. In the previous section (100% of nodes available), the Laplacian interpolation (12.41 $\mu$gr/m$^3$) already outperformed the GSP method (13.43 $\mu$gr/m$^3$). Moreover, as the percentage of available nodes decreases, the GSP low-pass-based method is the reconstruction method whose error increases faster, rising from 60% of nodes available. Thus, this method seems to be unable to generalize to the semi-supervised setting. The Laplacian interpolation and the kernelized ridge regressions are performing the best with a large difference. With 95% of the nodes available, the kernelized ridge regression with a diffusion kernel obtains a CV RMSE of 11.91 $\mu$gr/m$^3$ and the KRR with the vertex-covariance kernel obtains a CV RMSE of 11.50 $\mu$gr/m$^3$, clearly, the KRR-COV is the lower-bound of KRR-DIFF and Laplacian interpolation, and the three follow the same trend as the percentage of available nodes increases. In these latter cases, the error is slightly smaller than in the previous section probably due to the random selection of subsets of nodes, but it is noticed that with a high percentage of nodes available 95-80% the error increases from 11.91 to 12.34 $\mu$gr/m$^3$ in the KRR-DIFF case, and from 11.86 to 12.27 $\mu$gr/m$^3$ in the vertex-covariance case. Finally, the error of all the reconstruction methods seems to grow significantly from 40% of available nodes. In addition, there is also a factor that can affect the performance when several nodes are missing, which is the graph density D($\mathcal{G}$) or the number of neighbors per node $|\mathcal{N}(x_i)|$ (i.e., average graph degree), so that for sparse graphs the available number of neighbors for each node $\{\mathcal{N}(x_i) : i \in \mathcal{M}\}$ will be smaller. Each of the signal reconstruction models obtains the best performance for a different degree of graph density, so the number of edges can affect in this semi-supervised scenario. Thus, in addition to taking into account the optimal signal reconstruction performance when choosing the best graph, denser graphs can also be chosen for cases with extensive data losses.

In Figure 6.5.b) the same results can be seen for the $NO_2$ data set. The GSP low-pass-based method is the worst performing method, 8.66 $\mu$gr/m$^3$, with a high percentage of nodes available (95%). The two KRR methods obtain similar errors, 8.03 $\mu$gr/m$^3$ and 7.75 $\mu$gr/m$^3$ respectively, in the case of $NO_2$, the difference between the two is even less than in the case of $O_3$. The kernel ridge regressions and the Laplacian interpolation methods show a similar pattern, the error with 95% of the nodes (57 nodes) is less than the average error of all the nodes (section 6.3.3) due to the random node subset selection. As the percentage of available nodes decreases the error increases, although the increase is moderate, the GSP low-pass method degrades considerably from 60% nodes available, and for the kernel-based methods the error increase is produced from 40% to 20%, going from 9.57 $\mu$gr/m$^3$ to 10.19 $\mu$gr/m$^3$ and 9.33 $\mu gr/m^3$ to 9.98 $\mu$gr/m$^3$ for KRR-DIFF and KRR-COV respectively. In summary, the kernelized ridge regressions and the Laplacian interpolation perform similarly, and these methods get lower average R$^2$ for the $NO_2$ data since its concentrations are less smooth than the $O_3$.

Finally, Figure 6.5.c) shows the results for the $PM_{10}$ data set. Again, the methods follow the same trend, where as the percentage of available nodes increases the error decreases. It is worth remembering that in subsection 6.3.3 the $PM_{10}$ has been observed not to be predictable using neighboring concentrations, achieving a maximum average CV R$^2$ of 0.3 with the KRR-COV. Therefore, the same results are maintained in the semi-supervised case where the worst method is the GSP-based low-pass reconstruction and the best is the KRR-COV followed closely by its similar with diffusion kernel and the Laplacian interpolation. With a 20% of nodes available the methods get an error of 13.32 $\mu$gr/m$^3$ with GSP, 9.17 $\mu$gr/m$^3$ with Laplacian interpolation, 8.95 $\mu$gr/m$^3$ with KRR-DIFF and 8.81 $\mu$gr/m$^3$ with KRR-COV.

In summary, we can conclude that Laplacian interpolation and kernel-based methods are robust and efficient in reconstructing the signal when an acceptable percentage (>60%) of nodes are available, and that efficiency decreases as fewer nodes become available.

## 6.3.5. SIGNAL RECONSTRUCTION: SCALABILITY

Similar to the previous chapter 5 where clustering was proposed as a measure to improve the scalability of graph learning and signal reconstruction, here we show the complexity of the different reconstruction models studied in this chapter.

**Table 6.4:** Methods' cost along with their hyperparameters.

| Method | Cost | Hyperparameters | Observations |
|---|---|---|---|
| **Graph learning** | Iterative algorithm, requires solving iteratively a quadratic program and a matrix $\mathbb{R}^{N\times N}$ inverse | $\alpha$, $\beta$ | Easy to tackle cluster-wise |
| **Lap. Int** | Matrix inversion or multiplication (depending on if $\lvert\mathcal{U}\rvert << \lvert\mathcal{M}\rvert$) | None | $\mathbf{L}_{\mathcal{U}\mathcal{U}}$ may be singular |
| **GSP** | Matrix inversion or multiplication (depending on if $K << \lvert\mathcal{M}\rvert$) | K | Needs $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$ decomposition once: $\mathcal{O}(N^3)$ |
| **KRR-DIFF** | Matrix inversion or multiplication (depending on if $\lvert\mathcal{M}\rvert << \lvert\mathcal{U}\rvert$) | $\mu$, $\sigma^2$ | Kernel matrix $\mathbf{K}$ only needs to be computed once: $\mathcal{O}(N^3)$ |
| **KRR-COV** | Matrix inversion or multiplication (depending on if $\lvert\mathcal{M}\rvert << \lvert\mathcal{U}\rvert$) | $\mu$ | Kernel matrix $\mathbf{K}$ only needs to be computed once: $\mathcal{O}(N^3)$ |

Table 6.4 shows the costs and the required hyperparameters for the methods. It is important to mention that the learning of the graph is linked to the training of the reconstruction method since their hyperparameters are selected by means of a 5-fold CV. In this way, the more hyperparameters, the more dimensions the grid search will have and the more cost the CV will have. For instance, learning the graph for signal reconstruction by Laplacian interpolation will only require two hyperparameters ($\alpha$ and $\beta$) for the graph learning and none for the signal reconstruction method, and the CV will be faster than with the other GSR models. The graph learning task is solved by iteratively addressing a convex optimization problem that scales quadratically with the number of nodes $N$, so that, it is the most expensive task in the learning process. On the other hand, learning the graph for signal reconstruction with kernel ridge regression and diffusion kernel will require four hyperparameters ($\alpha$, $\beta$, $\mu$ and $\sigma^2$) and, therefore, will be the one that the grid search will take the longest. Similarly, to apply the kernel ridge regression with vertex-covariance kernel the covariance matrix needs to be estimated with the graphical lasso, which has one hyperparameter, implicitly learning a graph.

One way to speed up the CV is to fix the graph and then do the grid search on the hyperparameters of the GSR model, resulting in a suboptimal result as we have already seen how the choice of the graph is linked to the signal reconstruction model. As in the previous chapter 5, the simplest way to improve scalability is to use the cluster-based approach.

## 6.4. Concluding remarks & Future work

In this chapter, we have analyzed the performance of different GSR techniques that can be coupled to the data-driven graph learning technique. Indeed, we have focused on the particular environment of IoT sensor networks where any node can suffer losses, ergo, the subset of sampled nodes can vary at each time instant $t$. Thus, we have tackled the second research question posed regarding the application of graph techniques for air pollution sensor networks:

(**R.Q.2.2**): *How do different graph signal reconstruction techniques perform in air pollution monitoring networks? What is specific about them? What problems can they pose?*

First, we have analyzed the performance of the different GSR models depending on the graph (and its sparsity) used. The results have shown how the different techniques require a different graph, in the case of Lap. Int. the best performance is obtained for a sparse graph, the GSP obtains the best performance for a dense graph, the KRR-DIFF is quite inverse to the graph used, and the KRR-COV presents instability in the graph learning.

Secondly, we have investigated the performance of the models in scenarios where only one node is reconstructed and in scenarios where a subset of nodes is reconstructed, U=1 and U>1. First, we have observed how Laplacian interpolation and KRR have been able to obtain the best performance compared to GSP reconstruction. Furthermore, it has been observed that $O_3$ and $NO_2$ can be correctly predicted with these methods as opposed to $PM_{10}$. The reason is the higher correlation between reference station data for $O_3$ and $NO_2$, while $PM_{10}$ data are more local and with a lower correlation between stations. The KRR with the vertex-covariance kernel is the method that has worked best because it is the minimum local linear estimator, but the Laplacian interpolation and KRR with diffusion kernel work better for sparse graphs. Afterwards, we have studied the reconstruction with an increasing percentage of nodes to be estimated at the same time (e.g. nodes with missings) to simulate the semi-supervised learning paradigm. Again, the kernel methods and Laplacian interpolation outperform the other method and produce very good results when 0-40% of the nodes are missing, with little increase in the average error. Thirdly, we have also shown the complexity of the different GSR models along with their hyperparameters.

**Graph Signal Reconstruction For Air Pollution IoT Signals**

- The selection of the best graph and best graph signal reconstruction hyperparameters is coupled.
- Lap. Int. obtains the best performance for sparse graphs, GSP for dense graphs, KRR-COV for denser graphs than the Lap.Int., and the KRR-DIFF for denser graphs but its performance is quite invariant to the graph.
- $O_3$ signals can be better reconstructed than $NO_2$ signals, while $PM_{10}$ signal obtain a poor signal reconstruction given the low correlation between sensors' measurements.
- Kernel regression methods and Lap.Int. outperforms the GPS-based method.
- When more than a node is reconstructed, U>1, the average error increases a little

for 0-40% of unobserved nodes, while for >40% unobserved nodes the error increases significantly.

In short, we have seen how different linear GSR techniques adapt to the scenario of low-cost air pollution sensor network signals given their flexibility to deal with varying unobserved node sets.

As future work, it would be interesting to study the adaptation of novel nonlinear techniques such as graph neural networks for this scenario. In an environment such as LCS networks where; *i)* any node can present problems, *ii)* there are limited calibration data or training data for the models, and *iii)* as time progresses the sensors in the network may present problems. Transfer learning approaches could be useful to augment the training data sets and enhance the data requirements for the graph neural network settings.

**Practical Tip !**

In the case of signal reconstruction using related sensor measurements, kernel ridge regression and Laplacian interpolation are the most recommended techniques, with Laplacian interpolation being the simplest (in terms of the number of hyperparameters) and most effective technique for sparse graphs.

# 7

# Maintaining Sensor Network Data Quality Via Data Reconstruction

*What you learn from a life in science*
*is the vastness of our ignorance.*

David Eagleman

So far we have seen how most of the research has been based on the application of *in-situ* sensor calibration techniques to improve the accuracy of low-cost sensors (LCSs) [15, 16, 133]. Data quality has been reported as an important issue in air pollution LCSs, however, their low cost allows to considerably increase the spatial resolution of governmental networks formed by reference stations [6]. Therefore, there are many data quality problems that occur in LCSs during the lifetime of the deployment of these networks, problems such as the appearance of missing values, potentially caused by communication problems or temporary failures in the nodes, or the presence of virtual sensors, either because the sensor in question is under maintenance, has been relocated for maintenance, or because there is no physical sensor in that location. For all these reasons, post-processing techniques for heterogeneous sensor network data are important in order to deal with this variety of applications and provide accurate measurements so that the monitoring can be more reliable and these data can be used for different final applications.

Previously, graphs have allowed for describing the relationships between the different sensors in a heterogeneous sensor network. Consequently, in this chapter, we study how a framework based on learning a data-driven graph and using a graph signal reconstruction (GSR) model provides the necessary flexibility to deal with a wide variety of post-processing applications that may arise in heterogeneous air pollution sensor network deployments. Firstly, section 7.1 introduces the problem and poses the main contribution and goals of this chapter. Then, section 7.2 explains different post-processing applications studied in the literature. Section 7.3 describes the proposed framework as well as its range of applications. Section 7.4 explains the experiments carried out and the corresponding results. Finally, section 7.5 concludes the chapter. This chapter presents the methodology and results explained in "*Data reconstruction applications for IoT air pollution sensor networks using graph signal processing*", Elsevier JNCA, [134].

## 7.1. DATA QUALITY IN HETEROGENEOUS AIR POLLUTION SENSOR NETWORKS

GOVERNMENTS deploy high-precision instrumentation, called reference stations, in different cities to monitor pollution and to be able to determine different kinds of measures. These stations are capable of measuring and providing data for multiple regulated pollutants such as $O_3$, $NO_2$, NO, $PM_{10}$, etc. The problem lies in the price of the stations, which can cost tens of thousands of euros, which makes the number of stations deployed in a territory small. Thus, the spatial resolution of the monitoring network, defined as the number of stations available in an area, is small (e.g., measurements at neighborhood or district level).

There are different ways to increase the spatial resolution of an air pollution monitoring network in an economically viable way but at the cost of a worse accuracy in the measurements. LCSs in conjunction with enabling internet of things technologies (IoT) have presented an economically viable solution to this problem since these sensors cost around tens of euros. Therefore, the main line of research during the last years has been based on the improvement of the data quality of this type of sensors [4, 53]. In this way, nodes mounting LCSs can coexist with reference stations creating heterogeneous networks with a higher spatial resolution but with higher care of the network data quality [6]. The main challenge in the deployment of LCSs has been the *in-situ* calibration of these sensors, since they have to be calibrated at stations close to their deployment location due to their high dependence on environmental conditions, e.g., temperature and relative humidity [10, 11, 27].

Another alternative to increase the spatial resolution of a monitoring network is the creation of virtual sensors, thus increasing the resolution without the need to physically have more sensors [125]. Virtual sensing techniques can be based on the use of machine learning models or physical models. Some examples would be the use of machine learning techniques, graph-based models, consensus algorithms, physical models, etc. Besides, in the context of low-cost air pollution sensor networks, the use of virtual sensing is not limited to increasing the spatial resolution, but in providing measurements for sensor applications such as sensor recalibration, drift or aging correction, or the creation of boundary data for numerical simulation models.

**Definition 15** *A **virtual sensor** for air pollution monitoring can be defined as a mathematical model that estimates the air pollutant concentrations of interest without the need to have a sensor physically located in a location. Virtual sensors are also named soft sensors and digital twins in some fields, these terms have lately been used to denote algorithms capable of simulating an entity.*

In conclusion, monitoring and maintaining data quality in heterogeneous networks is very important and challenging. In fact, this type of network can present a large number of data quality issues, such as data loss, the presence of virtual sensors, node maintenance, node relocation, etc. The loss of contamination measurements at a specific

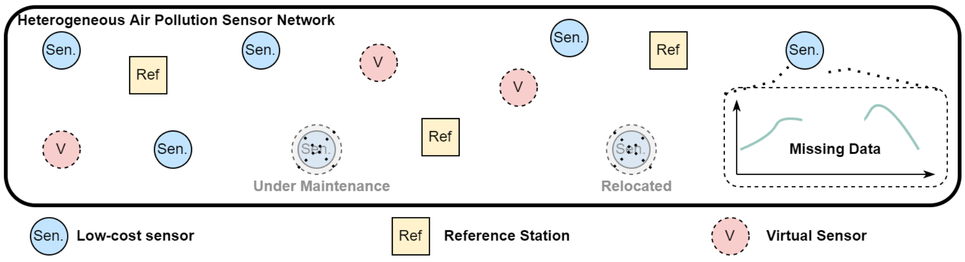


**Figure 7.1:** Scheme of a heterogeneous air pollution sensor network where different data quality issues may arise, such as missing data, the presence of virtual sensors, the relocation of sensors, or the presence of sensors under maintenance.

location can be caused by several issues, such as loss of sensor data, a node under maintenance, the node has been relocated to another site, communication or hardware failures, etc. Thus, there is a great need to deal with all these post-processing applications that may arise in such heterogeneous networks. Figure 7.1, shows an example of a heterogeneous sensor network deployed in an area of interest where the nodes may present different challenges.

The advent of *graph signal processing* (GSP) has posed the opportunity to use several signal processing tools on signals defined over graphs [21, 100]. Besides, graphs have been successfully applied in air pollution sensor networks (see chapter 5). Thus, the use of graphs over networks containing correlated sensors allows different post-processing applications to be addressed using the network data instead of using only the sensor data itself. Thus, in the previous chapters, we have seen different techniques to create a graph for an air pollution sensor network. Consequently, it is possible to use GSR to reconstruct data even if there are missings in the network. In this chapter, we propose a *graph-based data reconstruction framework*, the parts of which have been discussed in the previous chapters, that is able to perform different *post-processing applications* to guarantee the quality of the network data. In this way, we pose the different post-processing applications as data reconstruction problems and show how the graph-based framework is able to deal with this range of applications.

## 7.2. LOW-COST SENSOR POST-PROCESSING APPLICATIONS

IN this chapter, we place special emphasis on the two most important post-processing applications in air pollution sensor networks; missing value imputation and virtual sensing. These two tasks provide robustness and resilience to the sensor network by providing measurements in scenarios where there are data losses, there are no physical sensors, there are nodes under maintenance, etc.

### 7.2.1. MISSING VALUE IMPUTATION

The missing value imputation task can be seen as a way to fill the data gaps through estimations. Quinteros *et al.* [135] compare approaches such as mean imputation, conditional mean imputation, K-nearest neighbor imputation, multiple imputations, and Bayesian principal component analysis imputation for reconstructing an incomplete air quality data set. The objective was to use the data from the data set itself to perform the imputation. Based on this idea, many methods are based on the use of the entire data set through matrix completion methods. Liu *et al.* [136] propose the use of low-rank matrix completion methods for missing imputation of air pollutants given their strong spatial correlation. Okafor *et al.* [137] compare different machine learning-based imputation methods applied to sensors' time series and evaluated the impact of the imputation on the posterior sensor calibration, showing the superiority of Variational Autoencoders (VAE). Mondal *et al.* [138] develop a missing value imputation method for sensor networks based on spatio-temporal GSR via Sobolev smoothness. Matrix factorization techniques have also been used for data reconstruction for missing value imputation [139].

In general, matrix completion methods may not be well suited for real-time missing imputation since these techniques use the entire data set for gap estimation. In addition, models based on supervised machine learning may also present problems in cases where any node in the network may have missing data, making the network measurements incomplete.

### 7.2.2. Virtual sensing

The most popular trend in virtual sensor creation has been the application of supervised machine learning models. Zhang *et al.* [140] use k-means to cluster highly correlated nodes, identifying potential relationships for the creation of virtual sensors. Matusowsky *et al.* [124] create a machine learning-based virtual sensor to impute data or to use it as a substitute for a faulty sensor. Thus, this virtual sensor created from the other nodes using a multi-layer perceptron (MLP) algorithm served as a substitute for a sensor, providing high-accuracy results. Aiello *et al.* [141] show a machine learning-based algorithm for creating virtual sensors where a physical air pollution sensor cannot be placed. They trained a land use (LUR) regression-based framework to increase the number of sensors in a network so that the spatial resolution is increased. Fung *et al.* [121] define a proxy for black carbon as a virtual sensor using a feature selection scheme. Besides, they added robustness to the model by greedily adapting the covariates present in the model depending on the available set of sensors. Zaidan *et al.* [122] create a proxy based on a Bayesian neural network and a mutual information-based sensor selection scheme to avoid overfitting. Finally, Zaidan *et al.* [69] show a general methodology to calibrate LCSs and create virtual sensors for black carbon and carbon dioxide.

Overall, virtual sensors can be used for different applications, e.g., missing value imputation in the case of creating a virtual sensor for a faulty sensor, and are usually based on supervised machine learning techniques. Therefore, the main limitation of supervised machine learning-based models for data imputation or virtual sensor creation is their lack of flexibility, given that any sensor of the network can present missings, several models should be trained to fit several sensor availability scenarios. In the worst case, a complete graph, this would result in the creation of $2^{N-1}-1$ models per network sensor, although feature selection schemes could be used. In addition, some models may not use information from other nodes that have correlated data or that have more accurate data such as reference stations.

## 7.3. GRAPH-BASED DATA RECONSTRUCTION FRAMEWORK

IN this section, the graph-based data reconstruction framework is introduced as well as the different sensor post-processing applications.

This data reconstruction framework is composed by two elements; a graph $\mathcal{G}$ and a GSR model $f : \mathbb{R}^M \to \mathbb{R}^U$. As explained in the previous chapter 5 there are different techniques to infer a graph for a sensor network, some based on training data $\mathbf{X} \in \mathbb{R}^{N\times P}$ (being $N$ the number of network nodes and $P$ the number of graph signals) and some based on prior information (e.g., geodesic distance between nodes $d_{ij}$). Thus, any valid shift matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$ can be learned, either the Laplacian $\mathbf{L} \in \mathbb{R}^{N\times N}$, the weight matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$, or the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. From one of them, the rest can be computed (chapter 5, section 5.2). These matrices define the topology and set the performance of the GSR model $f(\cdot)$. The graph signals $\mathbf{x} \in \mathbb{R}^N$ are defined as the concentrations predicted or measured by the different nodes of the network at a time instant. In the case of the LCSs, these concentrations are the result of the machine learning model trained *in-situ* before the sensor network deployment [8, 10, 11, 14].

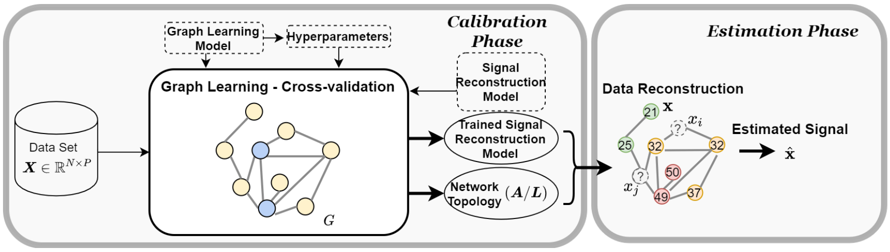

Calibration Phase
Graph Learning Model
Hyperparameters
Signal Reconstruction Model
Graph Learning - Cross-validation
Data Set
$\boldsymbol{X} \in \mathbb{R}^{N\times P}$
$G$
Trained Signal Reconstruction Model
Network Topology $(\boldsymbol{A}/\boldsymbol{L})$
Estimation Phase
Data Reconstruction
$\mathbf{x}$
$x_i$
$x_j$
21
25
32
32
50
49
37
Estimated Signal
$\hat{\mathbf{x}}$

**Figure 7.2:** Graph-based data reconstruction pipeline; from graph learning and hyperparameter selection to data reconstruction applications.

Figure 7.2 shows a general methodology for training the graph-based data reconstruction framework. Using a graph learning model, a GSR model $f(\cdot)$, the respective hyperparameters of these models **hyp**, and a training set of graph signals $\mathbf{X} \in \mathbb{R}^{N\times P}$[1], the best graph for the reconstruction of the network sensors' signals can be obtained. This is done in the following way, a grid search of two sets of hyperparameters is performed using a 5-fold CV procedure. On the validation set, each network node $x_i$ is reconstructed one by one assuming that all its neighbors are available, and the average RMSE is calculated, the same methodology as in previous chapters 5,6. The result of the CV procedure is a graph that on average has the lowest RMSE in the reconstruction of each of the nodes of the network. In fact, the CV procedure is used to find the minimization arguments in eq. (7.1):

$$\mathcal{G}, f = \underset{\mathcal{G}, f}{\operatorname{argmin}} \ \mathrm{RMSE}(\mathbf{X}, \mathbf{f}(\mathbf{X})) \tag{7.1}$$

In the following subsections, we describe the graph learning and GSR models as well as the details of the application of the framework.

[1]In the case of using a graph based on prior information (e.g., distances) training data are not necessary.

### 7.3.1. GRAPH LEARNING

As studied in the previous chapter 5, there are different techniques for graph construction, some based on data such as graph signal smoothness-based or graphical lasso and others based on prior information such as distances [23, 24]. Thus, the goal is to create an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$, which describes the existing relationships between the measurements of the different nodes. We focus on undirected graphs since the relationships between measurements are reciprocal. An undirected graph implies a symmetric shift matrix $\mathbf{S}^\top = \mathbf{S}$. Each graph learning model has a set of hyperparameters, which we call **hyp**, shown in the previous chapter 5. The distance-based graph allows for adding a new node at any time, simply by relating it to the others in the network. However, we opt for a *data-driven method* based on graph signal smoothness, as in the previous chapters this model together with a semi-supervised learning signal reconstruction model has been the most effective model for this type of network. Thus, we learn a graph $\mathcal{G}$, whose structure makes the training signals $\mathbf{X}$ smooth with respect to it. Dong *et al.* [104] uses the graph signal smoothness criteria to define the following non-convex optimization problem:

$$
\begin{aligned}
\min_{\mathbf{L},\mathbf{Y}} \quad & \|\mathbf{X}-\mathbf{Y}\|_F^2 + \alpha\, tr(\mathbf{Y}^\top \mathbf{L}\mathbf{Y}) + \beta \|\mathbf{L}\|_F^2 \\
\text{s.t.} \quad & tr(\mathbf{L}) = N, \\
& L_{ij} = L_{ji} \leq 0, \quad i \neq j, \\
& \mathbf{L}\cdot\mathbf{1} = \mathbf{0}.
\end{aligned}
\tag{7.2}
$$

Where $||\cdot||_F$ denotes the Frobenius norm and $tr(\cdot)$ denotes the trace of a matrix. In addition, [104, 105] show how these methods scale well in sparse matrix applications, which are the cases considered in this chapter. Indeed, Dong *et al.* [104] mention the use of alternating direction method of multipliers (ADMM) optimization methods to overcome possible scalability issues and learn larger graphs. Besides, we have already shown in chapter 5 how a cluster-wise strategy alleviates possible scalability problems.

### 7.3.2. SIGNAL RECONSTRUCTION

The reconstruction of the signal or the concentrations of a node $x_i$ can be approached from different perspectives (see chapter 5). More intuitively, a supervised machine learning model can be applied, where the value of the node of interest $x_i$ is regressed on the node's neighbors $\mathcal{N}(x_i)$, obtaining the function $f : \mathbb{R}^{|\mathcal{N}(x_i)|} \to \mathbb{R}$. An example is the use of multiple linear regression (MLR) to predict the values at each of the nodes $x_i$ in the network. In the case of having to reconstruct two nodes $x_i, x_j$ simultaneously, whose neighborhoods overlap $\mathcal{N}(x_i) \bigcap \mathcal{N}(x_j) \neq \emptyset$ and $i \in \mathcal{N}(x_j)$, a multivariate multiple linear regression can be performed with the two neighborhoods as covariates. This approach has a couple of drawbacks; *i)* the need to discover the neighborhoods of each of the nodes of the network, as they mark the covariates of the models, and *ii)* in the case of missings, with the need to have different combinations of covariates depending on the availability of sensors or an imputation mechanism. This MLR approach is the LMMSE in case the neighborhood is correctly selected, but has problems in case of missing data. The other alternative that best fits this scenario, in which any node can have data gaps, is the graph-based semi-supervised learning or GSR paradigm. Where a graph signal is defined as $x : \mathcal{V} \to \mathbb{R}$, and the objective is to regress a GSR function $f : \mathbb{R}^M \to \mathbb{R}^U$, where $U$

is the number of nodes to reconstruct and M is the set of nodes observed. We define the set of nodes that have missings in a time step $t$ as $\mathscr{U} \subset \mathscr{V}$ and the set of nodes that have available values as $\mathscr{M} = \mathscr{U}^c$. Then, we can use for example the Laplacian interpolation since it uses as shift matrix **S** the Laplacian matrix **L** learned earlier [108]. This method allows reconstructing any set of nodes $\mathscr{U}$:

$$\begin{aligned} \min_{\hat{\mathbf{x}} \in \mathbb{R}^N} \quad & \hat{\mathbf{x}}^\mathsf{T} \mathbf{L} \hat{\mathbf{x}} \\ \text{s.t.} \quad & \hat{x}_i = x_i \ , \forall i \in \mathscr{M} \end{aligned} \tag{7.3}$$

The above expression is convex and has closed form solution:

$$\hat{\mathbf{x}}_{\mathscr{U}} = f(\mathbf{x}_{\mathscr{M}}, \mathbf{L}) = -\mathbf{L}_{\mathscr{U}\mathscr{U}}^{-1} \mathbf{L}_{\mathscr{U}\mathscr{M}} \mathbf{x}_{\mathscr{M}} \tag{7.4}$$

As it can be noticed in case only one node is to be reconstructed $|\mathscr{U}| = 1$ and there are no missing data, only the neighborhood of that node $\mathscr{N}(x_i)$ is required for the reconstruction. However, if several nodes are to be reconstructed (e.g., because of missing data) $|\mathscr{U}| > 1$, then the required input data is the union of the neighborhoods $\bigcup_{i \in \mathscr{U}} \mathscr{N}(x_i)$. Hence, practically, the necessary data to reconstruct a node's signal is bounded since all nodes may not be required, so the signal reconstruction could be implemented in a distributed manner.

### 7.3.3. APPLICATION AND DEPLOYMENT

The implementation and deployment details of this framework are important to be applied in real-time on an air quality monitoring network[2]. We can divide the different tasks into three stages:

1. LCSs are calibrated *in-situ* at nearby reference stations of the deployment site. Thus, before the sensor network deployment, the machine learning models are trained to predict air pollution concentrations from LCSs' raw measurements in a centralized server [10, 11, 14, 16, 42, 133]. Given that applying these calibrated models is computationally inexpensive, these trained calibration models can be stored in the central server or in the nodes, so that, the nodes can send air pollution concentration measurements instead of raw measurements, following an edge computing paradigm [10].

2. Since the data collection nodes send samples to a central server, the graph can be computed during the first weeks of deployment, where it is assumed that the deployed sensors work correctly. The graph is learned by solving the optimization problem of eq. (7.2) using a CV procedure explained previously.

3. During the lifetime of the sensor network deployment, different applications (e.g., missing value imputation or virtual sensor creation for sensor maintenance) can be carried out in real-time using the data reconstruction framework. This set up would correspond to a data monitoring platform in a centralized server or implemented in a distributed manner.

[2]A python implementation of the proposed graph-based data reconstruction framework is available at https://bitbucket.org/sans-rg/jnca-graphdatareconstruction.

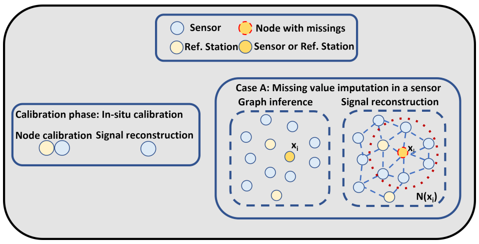
*Distributed approach*: the graph construction and signal reconstruction can be performed in a centralized or a distributed manner. The nodes report hourly air pollution concentrations, so to build the graph in a distributed manner, each node needs to do hourly flooding of the measured air pollutant concentrations (e.g., using a low-duty cycle flooding protocol [142]) during the training period of the graph, which is about three weeks. Then, the graph can be learned at any of the nodes or the central server and redistribute the learned topology to all the nodes. After the graph learning, flooding is no longer necessary, and the nodes in the distributed solution only report their hourly air pollutant concentration to the neighbors defined by the graph so that these measures are available in case of required signal reconstruction. The typical and most efficient solution in this type of sensor network is for the signal reconstruction operations to be performed by a centralized network monitoring tool, so the cost in bandwidth and energy consumption is minimal for the sensor nodes.

*Framework effectiveness*: it should be noted that the performance of data reconstruction will always depend on the existing relationships in the network. In other words, a network with sensors that have no relationships between them cannot benefit from using data from other deployed sensors. Along the same lines, Heimann *et al.* [143] stated that in order to distinguish between local emissions and the global emission pattern a dense network of sensors is needed. Moreover, during the CV process used for graph selection, it can already be observed whether any sensor in the network can be reconstructed less efficiently by looking at its CV metric. In addition, as more sensors in the network have gaps in the data, the reconstruction performance may also deteriorate as more missing sensors means fewer neighbors available for reconstruction.

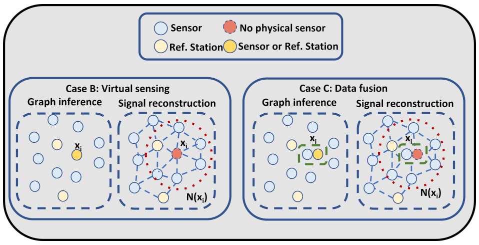
### 7.3.4. POST-PROCESSING APPLICATIONS

In this section, we explain the different applications that can be carried out using the data reconstruction framework. We explain the different stages of the deployment of an air pollution monitoring sensor network; from *in-situ* calibration to sensor post-processing applications. Prior to deployment of heterogeneous sensor network for air pollution monitoring, LCSs are calibrated *in-situ* by being collocated at a reference station or reference instrument near the location of the node deployment for a calibration period, Figure 7.3.a). After this calibration process, the node is deployed in an area participating in the monitoring network. For more information about the machine learning-based *in-situ* calibration refer to chapter 3.

From now on, we assume a monitoring network in which reference stations that give accurate values of the pollutants coexist with nodes mounting LCSs that give less accurate values, and these LCSs have been pre-calibrated using *in-situ* calibration [6]. In this sensor network scenario, we are interested in performing a signal reconstruction at a given node or location for several reasons: *i)* there is a LCS or a reference station that has not captured all the values of a time series, and we want to estimate the missing values, *ii)* there is a LCS node that needs to be recalibrated due to drifts or changes in atmospheric conditions from when it was calibrated, *iii)* to provide robustness when the physical sensor fails or is under maintenance, *iv)* to estimate a virtual sensor because we need values at points where we do not have a LCS, for example, to feed a computational fluid dynamics (CFD) model that builds a simulated air pollution transport model at small or large scale.



(a) *In-situ* calibration and missing value imputation cases.



(b) Virtual sensor and data fusion cases.

**Figure 7.3:** Description of different sensor pots-processing applications, and its corresponding graph-based data reconstruction setting, that arise in heterogeneous sensor networks; *in-situ* calibration, missing value imputation, virtual sensing, and data fusion.

We now explain the different applications, and their configuration in the graph-based data reconstruction framework, which allow us to address the problems explained in the previous paragraph. First of all, we address the missing value imputation via signal reconstruction, which allows for filling data gaps produced in a network sensor. Secondly, virtual sensing via data reconstruction is addressed to obtain estimates in a place where there is no physical sensor or to obtain estimates of a sensor presenting some kind of problem (e.g., under maintenance or a drifted sensor). Finally, we explore a data fusion case, in which two nodes are placed in the same location and the estimation of one sensor can be improved by reconstructing the data of the sensor located at the same place, i.e., creating a virtual sensor at the same place where there exists a sensor. Among the different applications we find:

**i)** ***Missing value imputation*:** in a monitoring network, both reference stations and LCSs usually do not report all data [122, 135], resulting in gaps in the sensors' time series. In the previous chapter 5, we reported a variable percentage of losses in the reference instrumentation of Barcelona (Spain) in the range of 1.5-3.0% of the data. Thus, heterogeneous sensor networks, including LCSs, are expected to produce the same or a large number of incomplete network measurements. The two most common types of missings in sensor measurements are *missings completely at random* (MCAR) and *missings at random* (MAR). MCARs are the ones addressed in most of the literature where the missings pattern is random. In contrast, the MAR corresponds to missings due to some cause, these missings usually depend on the other sensors, such as the loss of consecutive samples because the sensor is under maintenance, which can be imputed by creating a virtual sensor to fill the data. This scenario can also be seen as a *missing not at random* (MNAR) since the missings will not always depend on the other observed sensors but on some underlying cause, e.g., maintenance or sensor removal. Figure 7.3.a), case A, shows the architecture for completing the missing data $\mathbf{x}_i$ in the $i$-th node from its neighboring values $\mathbf{x}_{\mathcal{N}(x_i)}$. To estimate the missing data, either from a LCS or reference station, we need to apply a GSR model using the learned graph topology.

**ii)** ***Virtual sensing*****:** we consider as virtual sensor a vertex of the graph in which we want to estimate air pollution concentrations without having a physical sensor or to replace the physical sensor estimates. Figure 7.3.b), Case B, shows the virtual sensing setting. To do this we have to add a vertex $x_i$ in the graph in the place where we would place the target sensor. To train the graph we need to place a LCS or mobile reference station to train the graph and calculate the graph $\mathcal{G}$ that will relate the vertex representing the virtual sensor to its neighbors. In case you do not have a sensor to learn the graph, you could use a distance-based graph using a function that depends on the distance to the other nodes [109]. Once the graph $\mathcal{G}$ is obtained, we can remove the sensor or the mobile reference station to estimate the values of the virtual sensor. A supervised or semi-supervised method can now be used to reconstruct the signal $\mathbf{x}_i$ on the virtual sensor $x_i$ from its neighboring values $\mathbf{x}_{\mathcal{N}(x_i)}$. A special case is a proxy in which we consider that the estimation of the target node is made from the measurements of other pollutants. A key difference with case A is that in case B there is no data at the target node during the network deployment (i.e., there may not even be a physical sensor at the node where the signal is reconstructed, or the sensor is under maintenance) or the sensor provides erroneous data, whereas in case A, there is sensor data, with gaps, during the network deployment.

**iii)** ***Data fusion*****:** in the data fusion scenario, the value of the concentration at one point is estimated from several sensors, including a sensor that may be located at the target location. An example is a LCS, which gives inaccurate data or drifted data, and improves its estimates with sensors in its neighborhood. Figure 7.3.b), Case C, show the data fusion setting in which two nodes $x_i, x_j$ share the same location, so that they can be included in each other's neighborhood $j \in \mathcal{N}(x_i)$. Note that this scenario reassembles the virtual sensing case since it can be seen as the creation of a virtual sensor, yet, in this case, two nodes share the same location so estimates of the virtual sensor are likely to include measurements from the location-sharing node. For instance, we could locate a mobile reference station next to a LCS during the graph learning stage and create a virtual sensor for this reference station, which will no longer be available during the network deployment. Finally, in this case, both supervised and semi-supervised methods can be used to estimate the fusion of sensors that give the final concentration value. The main difference between this case and case A, is that in this case there can be two nodes at the same geographical location (e.g., case of a mobile reference station and a LCS, or several LCSs; chapter 4) and the target sensor can have data gaps or no data at all (i.e., as a virtual sensor).

The methodology for constructing the graph and reconstructing the signal is the same for all three cases, but the role and participation of the nodes in each phase is different. In the graph inference phase, we need the network nodes to train the graph, whether they are reference stations, mobile high-precision instruments, or a LCSs. Whilst, in the reconstruction phase, we reconstruct the data of a network node, regardless of its purpose or application.

## 7.4. EXPERIMENTAL EVALUATION

THE performance of the proposed data reconstruction network is evaluated for the three different post-processing applications. To do so, we use a data set collected from the heterogeneous H2020 Captor network during the 2017 summer. In the next section 7.4.1, we explain in detail the characteristics of this data set. This data set is similar to the one used in the previous chapters, but here we also use the LCSs placed at the reference stations.

For the different data reconstruction applications evaluated, we perform three different scenarios in which a subset of the network nodes is reconstructed to evaluate different data availability cases:

(A) **Scenario 1**: the nodes located at the reference stations are reconstructed one by one, assuming the other reference stations and sensors are available.

(B) **Scenario 2**: the nodes located at two of the reference stations are reconstructed simultaneously, assuming the other reference station and sensors are available.

(C) **Scenario 3**: the three nodes located at the reference stations are reconstructed simultaneously, assuming only LCS information is available.

These scenarios are performed to evaluate the possible case in which different nodes of the network must be estimated, either because they have data gaps, malfunction, or they are virtual sensors. In addition, they also allow for evaluating how the presence of reference stations affects the reconstruction of LCS nodes. To evaluate the different scenarios and applications, the experiments have been performed offline. We compare our approach with two data reconstruction techniques. The reconstruction by probabilistic matrix factorization (PMF) [139] and the spatio-temporal missing value imputation by Sobolev-based GSR using a distance-based graph [138].

### 7.4.1. DATA SET

For the analysis of the different applications described above, we use the data set of the heterogeneous network H2020 Captor [9]. This data set consists of three reference stations and eight LCSs measuring $O_3$ deployed during the summer of 2017. Three of these sensors remained at the reference stations throughout the summer. Table 7.1 summarizes the different data set characteristics.

This data set is unique since it includes data taken from a real deployment of low-cost air pollution sensors in conjunction with official reference stations, and therefore allows the study of all three applications mentioned above. Although the size of the network is small, the aim of this chapter is to show the feasibility of these techniques to solve the three cases. Figure 7.4 shows the location of these nodes over the deployment area: blue dots denote the reference stations -Manlleu is located at the top, Vic at the middle, and Tona at the bottom- and yellow dots denote LCS nodes. All sensing nodes underwent an *in-situ* calibration process at the nearest reference station of the deployment location using multiple linear regression (MLR) algorithm [8].

**Table 7.1:** Data set summary statistics.

| | |
|---|---|
| **Sensors** | 8 LCSs & 3 reference stations |
| **Period** | 07/2017 - 09/2017 |
| **# Samples** | 2612 samples |
| **Time Resolution** | 30 minutes |
| **Avg. Concentration** | 80 $\mu$gr/m$^3$ with peaks ~150$\mu$gr/m$^3$ |

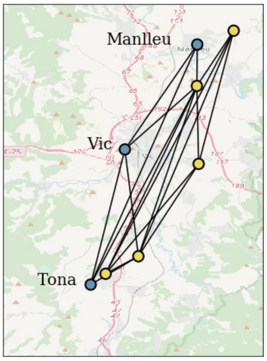


**Figure 7.4:** Graphical representation of the connectivity resulting from a Laplacian matrix.

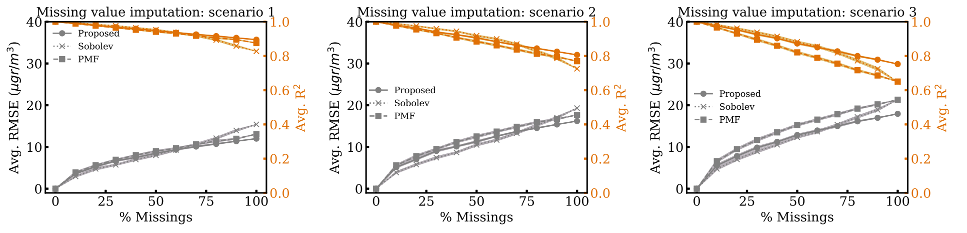
## 7.4.2. GRAPH LEARNING

The graph is learned using the methodology explained in the previous section 7.3 and previous chapters 5 and 6, where a 5-fold CV is performed on the network data to find the hyperparameters $\{\alpha, \beta\}$ that produce the minimum average CV RMSE. Then, in the data reconstruction case, the learned graph can be used both to impute the values of a node of the network and to produce an estimate of a virtual sensor corresponding to a graph node. The next section, delves into the details of the application of the graph-based data reconstruction framework on the H2020 Captor data set for the different post-processing applications described previously. Figure 7.4 shows the graph obtained for the network composed of the three reference stations and five LCSs, case B, with a total of 18 edges and a density of 64.29%.

## 7.4.3. MISSING VALUE IMPUTATION

In this first experiment, we explore the framework's ability to reconstruct missing measurements. An example can be the real-time reconstruction of measurements where the network data is displayed in a monitoring application and in the case of missings, an estimation must be given. Therefore, in this experiment we focus on case A, missing value imputation. To do so, we simulate a completely at random (MCAR) missings scenario where a percentage of sensor data is randomly lost. This type of missing is the most commonly addressed in the literature [137, 138]. In addition, we also explore the case where this type of missing occurs simultaneously in different nodes of the network (scenarios 1, 2, and 3). Specifically, we use the network configuration with LCSs at the reference stations (eight LCSs) and we miss a percentage of the testing data. Then, we compare the reconstructed values with the values of the sensors that have lost data.

Figure 7.5.a) shows the missing value imputation performance for the three methods when a sensor (in this case Manlleu sensor) loses data completely at random. The RMSE compares the predicted data with the model against the data obtained by the LCS, so if there are no losses, this RMSE has to be zero. As we can see in Figure 7.5.a) the RMSE value increases with the percentage of losses for all methods. The extreme case is a 100%



(a) Imputation results for Manlleu sensor.

(b) Imputation results when both Manlleu and Vic LCSs have data gaps.

(c) Imputation results when the three LCSs present data gaps.

**Figure 7.5:** Missing value imputation performance with respect to the percentage of missing data. The curve in orange is the average $R^2$ and the gray curve is the average RMSE obtained with the three different methods. The shaded area corresponds to the mean 95% confidence interval. The horizontal axis indicates the total percentage of samples lost by the sensor.

of lost data that would correspond to the case of obtaining the values for a virtual sensor, and its error corresponds completely to the imputation method performance. In general, the actual percentage of data lost in a reference station is usually low, 2-3% of data loss per station (chapter 5), but as the size of the network increases the chances of having an incomplete sample increase. However, there may be situations where the sensor reports incorrect data involving a higher percentage, such as a malfunction that lasts until the node is repaired. The PMF has a similar performance to the proposed method with worse performance for large missing ratios. The Sobolev-based method is able to improve the imputation performance of our proposed framework for low missing rates. This result makes sense since the Sobolev-based method takes into account the spatio-temporal correlations, so for small missing completely at random gaps (i.e., potentially non-consecutive) the temporal relationships allow to improve the estimation with respect to the proposed framework. Nevertheless, the disadvantage of these methods (matrix completion methods) is that they do not adapt naturally, or their performance is not as good, in real-time imputation where all the data are not available a priori. As the percentage of missings increases the proposed framework performs better since it does not include the temporal component. In the extreme case, 100% of missings, these consecutive missings can be seen as a case of missings at random (MAR) or missings not at random (MNAR), where the proposed framework is able to improve the RMSE of the Sobolev-based method by 3.50 $\mu$gr/m$^3$. Thus, in the case of consecutive missings (e.g, sensor failure for some time), the proposed framework works better. In case of 100% of losses, the reconstruction using the proposed framework obtains RMSEs of 12.00, 11.09, and 12.85 $\mu$gr/m$^3$ and MAEs of 9.26, 8.62, and 10.08 $\mu$gr/m$^3$, for Manlleu, Vic and Tona sensors.

Figures 7.5.b) and c) show the cases where more than one sensor have losses during the same instant of time, which makes the signal reconstruction difficult because it decreases the number of sensors in the vicinity that can participate in the reconstruction. With two sensors with missing data, Figure 7.5.b), the RMSE still remains similar to scenario 1, while with three missing sensors, Figure 7.5.c), the difficulty of the imputation increases. As for the performance of the three methods, the same trends as in the case of missings in a single sensor are observed, where as the ratio of missings increases the gap between the error of the Sobolev-based and the proposed framework increases, being the proposed framework the most effective. Nevertheless, it should be noted that this worsening in the reconstruction would be compensated by the average neighborhood size. We consider this situation a challenge to investigate with a real data set with a greater number of LCSs.

### 7.4.4. Virtual sensing

In this experiment, we produce estimates to create a virtual sensor, this means that there may not be a physical sensor at the node where we reconstruct the signal, or there may be a malfunctioning node, under maintenance or relocated, where the virtual sensor measurements replace those of the sensor. This case can also be seen as a case of MAR or MNAR, where the data loss is caused by some factor such as sensor malfunction, data loss due to sensor maintenance, sensor relocation, or the absence of a sensor.

For this case, we use the network consisting of five LCSs and the three reference stations. This setting allows us to estimate virtual sensor measurements at the reference station nodes and compare them with the measurements provided by the LCSs located at the same stations.

**Table 7.2:** Virtual sensing (case B) test set results for the different scenarios and different methods.

| **Scenario** | **Nodes** | **Method** | **Target Location** | | | | | | | | | **# Edges** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ***Manlleu*** | | | ***Vic*** | | | ***Tona*** | | | |
| | | | **RMSE** | **MAE** | **$R^2$** | **RMSE** | **MAE** | **$R^2$** | **RMSE** | **MAE** | **$R^2$** | |
| 1 | 5 LCSs + 2 Ref.Stat. | *Proposed* | 11.92 | 8.67 | 0.88 | 10.02 | 7.70 | 0.92 | 11.38 | 8.99 | 0.87 | 18 |
| | | *Sobolev* | 20.35 | 16.91 | 0.64 | 21.09 | 17.75 | 0.63 | 18.75 | 15.46 | 0.65 | 19 |
| | | *PMF* | 14.51 | 11.15 | 0.82 | 11.74 | 9.15 | 0.89 | 13.04 | 10.37 | 0.83 | - |
| 2 | 5 LCSs + *Tona* Ref.Stat. | *Proposed* | 15.02 | 11.34 | 0.81 | 12.86 | 10.30 | 0.86 | - | - | - | 18 |
| | | *Sobolev* | 24.63 | 20.74 | 0.48 | 23.57 | 19.87 | 0.54 | - | - | - | 19 |
| | | *PMF* | 17.20 | 13.55 | 0.75 | 14.39 | 11.13 | 0.83 | - | - | - | - |
| 2 | 5 LCSs + *Manlleu* Ref.Stat. | *Proposed* | - | - | - | 11.59 | 8.34 | 0.89 | 13.03 | 10.10 | 0.83 | 18 |
| | | *Sobolev* | - | - | - | 24.04 | 20.33 | 0.52 | 23.00 | 19.35 | 0.47 | 19 |
| | | *PMF* | - | - | - | 13.41 | 10.15 | 0.85 | 14.71 | 11.32 | 0.78 | - |
| 2 | 5 LCSs + *Vic* Ref.Stat. | *Proposed* | 12.00 | 8.76 | 0.88 | - | - | - | 11.61 | 9.31 | 0.87 | 18 |
| | | *Sobolev* | 20.48 | 17.04 | 0.64 | - | - | - | 18.84 | 15.52 | 0.64 | 19 |
| | | *PMF* | 15.20 | 11.66 | 0.80 | - | - | - | 13.98 | 11.34 | 0.80 | - |
| 3 | 5 LCSs | *Proposed* | 24.25 | 18.66 | 0.49 | 23.29 | 18.53 | 0.55 | 21.57 | 16.76 | 0.53 | 18 |
| | | *Sobolev* | 26.27 | 22.78 | 0.41 | 27.47 | 23.35 | 0.38 | 23.77 | 20.04 | 0.43 | 19 |
| | | *PMF* | 25.69 | 20.44 | 0.43 | 25.70 | 20.91 | 0.46 | 23.50 | 19.27 | 0.45 | - |

Table 7.2 shows the results obtained for the reconstruction of each of the reference stations, indicating the available nodes for the data reconstruction, and the corresponding data reconstruction method used. Regarding the proposed method, RMSE values of 11.92, 10.02, and 11.38 $\mu$gr/m$^3$ and MAE values of 8.67, 7.70, and 8.99 $\mu$gr/m$^3$, are obtained for the reference stations of Manlleu, Vic, and Tona respectively. These values are similar to the values obtained by the LCSs placed in those stations, case of *in-situ* calibration (Table 7.3), with RMSE values of 10.85, 11.30, and 12.21 $\mu$gr/m$^3$, and MAE values of 8.04, 8.58 and 8.48 $\mu$gr/m$^3$. Therefore, air pollution concentrations at one station can be approximated using neighboring nodes. In this case, the state-of-the-art Sobolev-based reconstruction model obtains a much lower performance, obtaining test RMSEs of 20.35, 21.09, and 18.75 $\mu$gr/m$^3$, while the PMF is able to obtain a good performance

obtaining test RMSEs of 14.51, 11.74, and 13.04 $\mu$gr/m$^3$. In the case of virtual sensing, including the temporal component worsens the estimation since the reconstruction is performed at consecutive times. In addition, the construction of the data-driven graph also gives an advantage in terms of reconstruction performance over the distance-based graph used by the Sobolev-based method since heterogeneous air pollution sensor networks contain complex relationships which are not well defined by distances.

For scenario 2, Table 7.2 shows the results of predicting two virtual sensors simultaneously with only one reference station available. The obtained test RMSE values are still in the same range as in scenario 1 for the proposed framework. In the worst case, Manlleu and Vic estimate, the RMSE increases by about 3.00 $\mu$gr/$m^3$. Whereas in the best case, Manlleu and Tona estimation, the error increases only about 0.30 $\mu$gr/m$^3$, since Vic station is available and close to both Tona and Manlleu. Finally, for scenario 3, Table 7.2 shows the performance of estimating three virtual sensors simultaneously. In this case, the performance is worse as there are fewer neighbors with which to predict the signal and all are LCSs. Logically, as more nodes in the network lose data, less information is available and data reconstruction is less accurate (see chapter 6, section 6.3.4).

In short, some state-of-the-art methods have been able to obtain similar performance in the case of MCAR missings, case A, with superior performance of the Sobolev-based model for low data losses, but in the case of virtual sensing, the Sobolev-based model has not been able to obtain a performance similar to the proposed framework. Thus, learning a graph from the data and reconstructing the graph signal without considering the temporal trend allows for dealing with a larger variety of post-processing applications that arise in heterogeneous networks of LCSs for air pollution monitoring, where there can be random losses, consecutive losses, and even no data at all. In the following case, data fusion, we only show the results for the proposed framework since it also corresponds to the case of estimating a node with no data at all, and a distance-based graph does not allow for multiple sensors at the same location.

### 7.4.5. DATA FUSION

Finally, in the case of data fusion (Case C), we evaluate the signal reconstruction when the neighborhood of a node $\mathcal{N}(x_i)$ includes a sensor located at the same location ($d_{ij} = 0$). Thus, for the reconstruction, information from other locations and from the same location of the target node is fused, leading to faulty sensor compensation and improvement of LCS estimates. As an example, when calibrating a LCS with a mobile instrument we could add a node in the graph representing the instrument and reconstruct its signal using the LCS located at the same place and other neighboring sensors. We recall that the neighborhood selection is performed by learning the graph, then the use of the LCS $x_j$ from the same site will depend on whether it is included in the neighborhood of the mobile instrument ($j \in \mathcal{N}(x_i)$).

Data fusion results in Table 7.3 show the advantage of using a physical sensor placed at the target location merged with neighboring sensors. The effect of fusing sensor data is better when some of the neighbors are reference stations that have reference air pollution concentration values. These true values allow the correction of the LCS error. For example, it can be observed in Table 7.3, scenario 1, how data fusion outperforms the *in-situ* calibration performance, obtaining RMSEs of 9.55, 8.59, and 9.73 $\mu$gr/m$^3$ respec-

**Table 7.3:** Data fusion (case C) test set results for the different scenarios using the proposed framework and the *in-situ* calibration.

| Scenario | Nodes | Target Location | | | | | | | | | # Edges |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *Manlleu* | | | *Vic* | | | *Tona* | | | |
| | | **RMSE** | **MAE** | **$R^2$** | **RMSE** | **MAE** | **$R^2$** | **RMSE** | **MAE** | **$R^2$** | |
| **1** | 5 LCSs + Manlleu LCS + *Vic* and *Tona* Ref.Stat. | 9.55 | 7.11 | 0.92 | - | - | - | - | - | - | 22 |
| **1** | 5 LCSs + *Vic* LCS + *Manlleu* and *Tona* Ref.Stat. | - | - | - | 8.59 | 6.68 | 0.94 | - | - | - | 22 |
| **1** | 5 LCSs + *Tona* LCS + *Manlleu* and *Vic* Ref.Stat. | - | - | - | - | - | - | 9.73 | 7.05 | 0.91 | 11 |
| **2** | 5 LCSs + *Manlleu* and *Vic* LCSs + *Tona* Ref.Stat. | 10.28 | 7.91 | 0.91 | 9.79 | 7.89 | 0.92 | - | - | - | 14 |
| **2** | 5 LCSs + *Vic* and *Tona* LCSs + *Manlleu* Ref.Stat. | - | - | - | 9.22 | 7.07 | 0.93 | 10.16 | 7.29 | 0.90 | 14 |
| **2** | 5 LCSs + *Manlleu* and *Tona* LCSs + *Vic* Ref.Stat. | 9.38 | 7.02 | 0.92 | - | - | - | 9.79 | 7.18 | 0.90 | 14 |
| **3** | 5 LCSs + *Manlleu*, *Vic* and *Tona* LCSs | 12.06 | 8.93 | 0.87 | 12.15 | 9.45 | 0.88 | 11.81 | 8.09 | 0.86 | 16 |
| **In-situ calibration** | *Manlleu*, *Vic* and *Tona* LCSs | 10.85 | 8.04 | 0.90 | 11.30 | 8.58 | 0.89 | 12.21 | 8.48 | 0.85 | - |

tively, improving up to 2.70 $\mu$gr/m$^3$.

For scenario 2, Table 7.3, the data fusion also shows good results, improving the *in-situ* calibration although it does not manage to reduce the error as much as in the case where all the neighbors are available. Despite that a neighboring station may not be available, we do have the LCS that is located at the target node's location. For instance, estimating Vic and Tona simultaneously produces RMSEs of 9.22 and 10.16 $\mu$gr/m$^3$, so it does not improve as much as in the first scenario (8.59 and 9.73 $\mu$gr/m$^3$) but it reduces the *in-situ* test RMSE at least 1.60 $\mu$gr/m$^3$.

Finally, Table 7.3 scenario 3 (when no reference stations are available), data fusion performs worse than the *in-situ* calibration given that none of the reference stations are available and the estimation relies on the LCSs placed there. However, the obtained RMSEs of 12.06, 12.15, and 11.81 $\mu$gr/m$^3$ respectively, and MAEs of 8.93, 9.45, and 8.09 $\mu$gr/m$^3$ respectively, are not much larger than in the *in-situ* calibration. However, only taking into account neighboring LCSs worsens the signal reconstruction. In conclusion, the use of reference stations improves data fusion in the LCS estimation. Nevertheless, the use of neighboring LCSs along with the target sensor does not improve the estimation if the neighboring sensors are not more accurate than the collocated sensor.

### 7.4.6. DATA FUSION APPLICATION: ERROR COMPENSATION

The occurrence of aging or noise is an important source of error in LCSs [12]. Here, we show how the data fusion scenario can help mitigate the error or noise present in a faulty LCS. As an example scenario, we use the sensor network composed of all the LCSs (eight sensors) and simulate a mobile reference station in Manlleu using this reference station. Then, we introduce 30% white Gaussian noise to the LCS in Manlleu in the test set. Using the data reconstruction framework, Manlleu's reference station signal can be reconstructed using the faulty sensor and the related neighboring LCSs. These estimates can replace the measurement obtained by the faulty sensor placed in the same location. Thus, it can be checked whether the combination of a faulty sensor with potentially non-faulty LCSs can mitigate the sensor's noise.

Recall that under optimal conditions, the Manlleu LCS obtained a test RMSE of 10.85

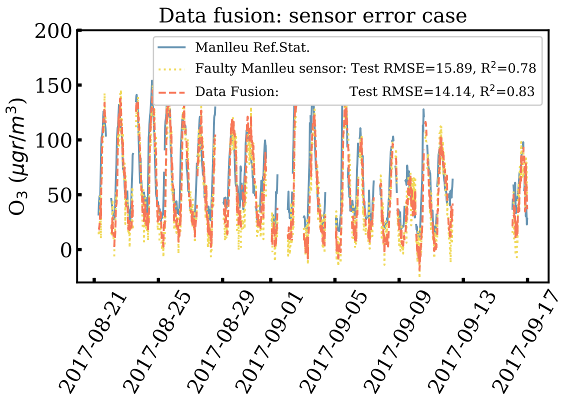


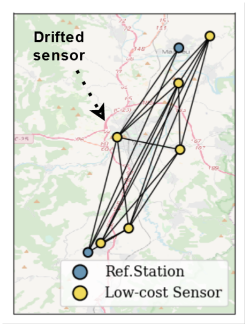
**Figure 7.6:** Manlleu's LCS Gaussian error compensation using the data fusion scenario.

$\mu$gr/m$^3$ (Table 7.3). As shown in Figure 7.6, the faulty LCS obtains an RMSE of 15.89 $\mu$gr/m$^3$. Using the data fusion scenario, the RMSE is reduced from 15.89 to 14.14 $\mu$gr/m$^3$, thus, neighboring LCSs can partially compensate for the error introduced in the LCS. Note that the graph learned with this scenario connects the Manlleu reference station with the LCS in Manlleu and the LCSs in Tona and Vic, allowing the data fusion to be effective. Otherwise, in the case of not having relationships between other sensors apart from the faulty sensor, the fusion would not improve the accuracy of the estimates produced by the faulty sensor.

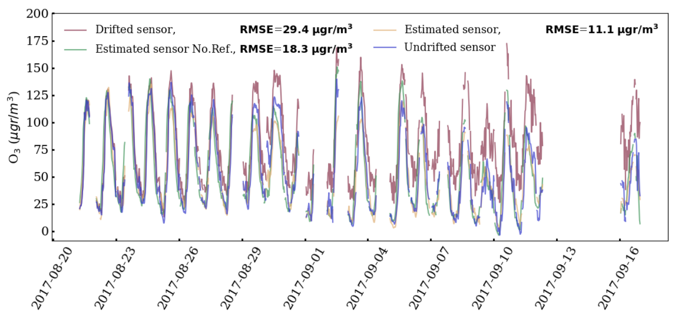
### 7.4.7. VIRTUAL SENSING APPLICATION: DRIFT COMPENSATION

The presence of drifts, either produced by the sensor itself or by an out-of-date calibration model (concept drift, see chapter 3), is a potential source of long-term error in a LCS network deployment. In this section, we show an example where the data reconstruction framework can be used to create a virtual sensor that produces estimates that can replace a faulty LCS or can be used to recalibrate the faulty sensor. In this case, the data reconstruction framework helps to maintain the quality of the network data by providing virtual sensor estimates that can replace or correct the drifting measurements.

For this experiment, we use the sensor network composed of two reference stations (Manlleu and Tona) and six LCSs (including Vic LCS). Afterwards, we introduce Gaussian noise of increasing magnitude with time, $\epsilon_t \sim N(\frac{t}{T}\bar{x}, 0.25)$ to Vic's LCS. We investigate two possible scenarios; both neighboring reference stations and LCSs are available, and only neighboring LCSs are available. These scenarios allow for investigating the impact of having precise instrumentation in the neighborhood available. Figure 7.7.a) shows the disposition of the network nodes as well as the graph's edges learned using the data reconstruction framework. As it is observed, the drifting LCS is connected to the two reference stations and four LCSs. Figure 7.7.b) shows the estimations of the created virtual sensor when all neighboring nodes are available (including reference stations) and when only LCSs are available. The drifted LCS produces an RMSE of 29.40 $\mu$gr/m$^3$, the RMSEs are computed with respect to the normal behavior of the sensor. The performance of the



(a) 19-edge graph learned from the H2020 Captor data.



(b) Results for simulated drift compensation in two scenarios; using reference stations and LCSs, and using only LCSs.

**Figure 7.7:** Example of a virtual sensor creation to replace a faulty LCS using the graph-based data reconstruction framework. Both scenarios, reference stations available and only LCSs available, are simulated.

virtual sensor created by using all its neighbors, including the two reference stations, is good obtaining an RMSE of 11.10 $\mu$gr/m$^3$, showing the benefits of having precise instrumentation nearby to correct possible seasonal changes. Finally, it has been simulated the unavailability of the two reference stations. Even in this case, the virtual sensor is able to improve the RMSE from 29.40 to 18.30 $\mu$gr/m$^3$ only using neighboring LCSs, indicating the possibility of mitigating drifts using only neighboring LCSs. It is worth mentioning how in the case of correcting drifting sensors, the inclusion of the sensor measurements itself could result in an error accumulation over time. Therefore, in the case of creating virtual sensors, it is better to use only neighboring nodes.

## 7.5. Concluding remarks & Future work

While throughout the previous chapters we have evaluated different techniques to create graphs and reconstruct signals for air pollution LCS networks, in this chapter, we have seen how different post-processing applications that arise in heterogeneous LCS networks can be carried out using a graph-based data reconstruction framework. Thus, we have approached the third question posed about graph-based sensor network analysis:

(**R.Q.2.3**): *For which applications can the signal reconstruction using the sensor network data be used?*

We have proposed a graph-based data reconstruction framework composed of a graph learning stage based on a graph signal processing smoothness-based criterion and a graph signal reconstruction stage based on the Laplacian interpolation. Both stages provide the framework with high flexibility to deal with different sensor network data post-processing applications that appear in heterogeneous networks of LCSs for air pollution. Among the different possible applications, we have investigated the use of the framework in; missing value imputation, virtual sensing, and data fusion. To summarize, in the case of missings completely at random, the performance of the proposed framework has similar performance to the state-of-the-art models, although these have better accuracy in the case of low missing rate percentages. This is due to the fact that the matrix completion-based models use the temporal component of the sensor with missings, so in small data gaps they are superior. However, matrix completion methods can present more difficulties in their real-time application. In the case of larger data gaps, approximating to the case of virtual sensing, the proposed framework performs obtains a good performance. In the case of virtual sensing, the proposed model performs better than the other models, being slightly better than the probabilistic matrix factorization. This case may represent the case of 100% loss, so models including the temporal component of the sensor itself tend to accumulate the error in the reconstructed sensor. Finally, in the case of data fusion, we have presented a scenario that allows combining information from a sensor at one location and sensors from other locations in order to improve the sensor estimation. The results show how combining information from different sensor locations improves individual sensor estimation.

In short, the proposed framework, although it may not be optimal for some applications (e.g., missing value imputation) presents the necessary flexibility to deal with different applications that arise in this type of network that may require real-time processing. Therefore, this framework could be applied in monitoring platforms to guarantee the quality of the network data in real time. More precisely, we can present the conclusions as follows:

**Missing Value Imputation**

- The proposed framework has obtained similar performance to two state-of-the-art models for MCAR missings.
- For a low percentage of MCAR, the state-of-the-art model has performed slightly better than the proposed method, while for large percentages the proposed framework performed better.
- The proposed method has shown the ability to deal with simultaneous missings with a performance similar to the state-of-the-art models.

**Virtual Sensing**

- The creation of virtual sensors has shown a quality of estimation similar to that of LCSs.
- The proposed framework has shown a better performance than the other models, as this case can be seen of a 100% of losses, with the ability to deal with simultaneous virtual sensors.
- The availability of the neighborhood has had a great impact on the data reconstruction quality.
- Virtual sensing using the data reconstruction framework has successfully produced estimates for a drifting sensor.

**Data Fusion**

- The data fusion results have been shown to outperform the estimation of *in-situ* calibrated LCSs.
- The data fusion scenario has been successfully applied to the compensation of a noisy LCS.

As a future work, it would be interesting to investigate the creation of air pollution proxies, having sensor networks composed of different air pollutant sensors and with sensors coinciding at the same location, using the graph-based approach. It would also be interesting to study how the virtual sensing case can be used to recalibrate and relocate sensors, as well as the study of more advanced graph-based data reconstruction techniques that could fit this particular scenario.

**Practical Tip !**

In order to monitor and maintain the quality of data of a sensor network, one can use the graph-based data reconstruction framework as a real-time monitoring tool. Thus, missing samples can be completed, virtual sensors can be obtained during maintenance/recalibration periods, and other situations where it is necessary to obtain estimates for the affected sensors.

# 8 Graph-Based Sensor Network Outlier Detection

*An experiment is a question which science poses to Nature,*
*and a measurement is the recording of Nature's answer.*

Max Planck

In previous chapters, we have seen how low-cost sensors (LCSs) tend to suffer from errors and inaccuracies, which must be considered in the analysis of sensor network data [4, 8]. Firstly, *in-situ* calibration techniques have been used to improve the data quality of these sensors [14, 15]. Once calibrated and deployed, we have investigated how graphs can describe the complex relationships that exist between the different sensors of a heterogeneous sensor network. In fact, we have proposed a graph-based approach to different applications that arise in this type of network such as; missing value imputation, virtual sensing, or drift compensation. However, there remains another very important aspect of the monitoring of the measured network data, which is the detection of erroneous measurements that may indicate a malfunctioning sensor or corrupt measurements. Recently, this aspect has gained interest given the attention paid to LCSs and especially their performance in terms of data quality in order to be able to carry out regulated measurement campaigns [144]. Actually, this outlier detection aspect is of special interest in this type of network in order to increase the reliability of the data and monitor how the network is working and carry out maintenance operations if necessary. Moreover, in addition to guaranteeing the performance of the data reconstruction by means of the graph, we need to detect those measurements that are considered incorrect or of very low quality. In this chapter, we approach the monitoring of the sensor network data from the graph-based *outlier detection* perspective, detecting anomalous measurements using the graph describing the relationships between the different network nodes.

This chapter is structured as follows; section 8.1 presents the need for outlier detection models for LCS networks. Then, section 8.2 presents the proposed *Volterra graph-based outlier detection* model and section 8.3 evaluates its performance. Finally, section 8.4 concludes the chapter and presents future research directions. This chapter presents the findings made in "*Volterra Graph-Based Outlier Detection for Air Pollution Sensor Networks*", IEEE TNSE, [145].

## 8.1. Heterogeneous sensor network data quality issues

Heterogeneous low-cost sensor networks are the future of fine-scale air pollution monitoring networks, but they face data quality problems [6, 146]. Specifically, throughout this thesis we have approached the data quality issue from two points of view; the improvement of LCSs data quality by *in-situ* machine learning-based calibration, and the maintenance of sensor network data quality by means of graphs. Thus, throughout chapters 1, 2, 3, and 4, we have studied the improvement of the data provided by LCSs using ML techniques to ensure good data quality at sensor level. Afterwards, during chapters 5, 6, and 7 we have studied how many problems can exist in heterogeneous monitoring networks given the reliability of LCSs, and how graphs can help to exploit the existing correlations between network nodes and provide tools to deal with a wide variety of post-processing applications (e.g., missing value imputation, virtual sensing, drift compensation).

In the following, we review the LCSs data quality problems and their possible consequences, and also highlight the need for outlier detection tools as well as enumerate different state-of-the-art outlier detection techniques.

### 8.1.1. Low-cost sensors data quality

LCSs represent an economically feasible alternative to increase the spatial resolution of air quality monitoring networks. As it is well known, high-precision instruments deployed by governments have a high economic cost, being impossible to measure pollution at a very fine scale. Therefore, the first limitation of these sensors that is their need for calibration has been studied in great detail, making use of both linear and nonlinear supervised ML techniques [14–16, 31, 147]. For more information on improving the data quality of LCSs using machine learning, see chapters 1, 2, 3, and 4.

In an air quality monitoring campaign using LCSs, once these LCSs have been calibrated *in-situ*, they are deployed in the different locations of interest forming a sensor network. So much effort has been put into improving data quality so that these sensors can meet minimum quality guarantees to be used in a regulated manner so that governments can take preventive measures and action against pollution episodes [4, 83, 148]. Although *in-situ* calibration improves the sensor data quality substantially, there are still more problems concerning the quality of these sensors. These problems arise when IoT nodes mounting LCSs form a sensor network for an expected long-term air quality monitoring campaign.

During the sensor network deployment is when other data quality issues arise. Even though the sensors have been calibrated *in-situ*, these sensors are known to suffer from aging, drift, and concept drift problems as time passes and environmental conditions differ from those observed during the calibration period [12, 18, 144, 149]. Therefore, LCSs tend to produce errors in such network deployments. In addition, it should be noted that these sensors mounted on IoT devices are of low-cost nature or simply not very resilient to the environment where they are deployed, generating data quality problems such as the occurrence of missings due to communication system or sensing system failures, erroneous measurements, etc.

In addition, as we have seen in the previous chapter 7, the sensor network data can be used to carry out post-processing applications such as data imputation [138] or the creation of virtual sensors, among others, and in different applications such as the creation of air pollution maps [19, 150]. That is, the network can work cooperatively to correct information using the own sensor network measurements, so it is essential to detect whether a sensor has anomalous measurements in order not to take it into account and not to propagate this error to other measurements in the network.

In short, it is necessary to identify anomalous measurements and malfunctioning sensors in order to carry out replacement measures, estimation, or recalibration using the other sensors in the network. In the following section, we explain the use of *outlier detection* techniques for air pollution sensors as well as different state-of-the-art techniques.

### 8.1.2. DETECTION OF OUTLIERS & SENSOR ERRORS

So far, we have highlighted the need to detect anomalous measurements and erroneous sensors in order to prevent the performance of various applications from degrading and to carry out replacement, maintenance, estimation, and recalibration actions if necessary. Therefore, we study outlier detection processes to detect anomalous measurements in air pollution sensors.

**Definition 16** ***Outlier detection*** *is the process of detecting measurements that are different from the rest of the data, which we call outlying or anomalous measurements, in some statistical sense. For instance, an outlying measure may be an extreme value of the statistical distribution observed during a period of time. In the case of multivariate measurements, it may be that the joint distribution of the measurements differs from the previously observed distribution, e.g., the distance between the observed values and historic data is large. And in the case of spatial data, i.e., data from a sensor network measuring a phenomenon, a measurement may be an extreme value of the spatial distribution considered normal.*

There exists a great variety of algorithms for the detection of outliers of different natures, Table 8.1 describes some of them. In our scenario, unsupervised models are the cornerstone since there is no prior information on which measurements may be outliers and it is also assumed that the sensors work well at least for a period after the calibration and deployment of the sensor network. Hence, this task can be seen as *unsupervised novelty detection* since the training data is assumed to not include outliers, so the training distribution is learned. We also distinguish between local or univariate detectors and global or multivariate detectors. The local ones make use of the distribution of the sensor itself or other sensors and are able to detect which sensor has an anomalous measurement, as is the case of z-score techniques [151]. In the case of global or multivariate models, it is no longer detected which is the sensor with an anomalous measurement but it is known that the joint measurements (seen as a sample) of the network are anomalous, such as the use of principal components analysis (PCA), widely used in this field [152–154], or other ML techniques [155–158]. There are also local models based on residuals, i.e., the difference between a sensor measurement and a model's prediction, that allow identifying the anomalous sensor, models based on spatial statistics [159] and others based on graphs [160].

In our case, and given the proposal of this thesis, we opt for the graph signal processing (GSP) paradigm. Indeed, the growing field of GSP has shown its flexibility in describing this type of network as well as providing classical signal processing techniques for their analysis [21, 103]. This field mainly relies on the assumption of signal smoothness, assuming that similar sensors will be strongly connected while non-similar sensors will be weakly connected or disconnected. The interpretation of the measurements as a signal defined over a graph allows the calculation of the Fourier basis and the interpretation of the different frequency components through the graph discrete Fourier transform (GDFT)[22]. The search for high frequencies to detect outliers has already been used for outlier detection, as the magnitude of the high frequencies is increased due to abrupt changes in similar nodes [161]. That is why the description of the topology by means of a graph and the subsequent application of filtering or anomalous frequency detection techniques are good candidates for this type of sensor network. More recently, Xiao *et al.* [160] developed a third order nonlinear polynomial graph filter (NPGF) to implement a residual-based outlier detector, with good results in the detection and localization of daily mean temperature outliers. These residual GSP-based techniques offer great outlier detection capabilities in the sensor network realm since they can locate which is the abnormal sensor measurement.

**Table 8.1:** Summary of different outlier detection methods used in the literature. "Local" refers to the capability of locating the outlying sensor while "global" refers to identifying an outlying network measurement as a whole.

| Class | Realm | Method | Definition | References |
|---|---|---|---|---|
| Local | Statistics | z-score | $z_i = \frac{\lvert x_i - \text{mean}_{\mathbf{x}}\rvert}{\sigma_{\mathbf{x}}}$ | [155, 162] |
| Local | Spatial | Median Algorithm | $z_i = \frac{\lvert x_i - \text{median}_{\mathbf{x}}\rvert}{\sigma_{\mathbf{x}}}$ | [162] |
| Local | Spatial | Weighted z-algorithm | $\begin{cases} S(x_i) = \lvert x_i - \frac{1}{\lvert N(x_i)\rvert}\sum_{j\in N(x_i)} w_j x_j\rvert \\ z_i = \frac{\lvert S(x_i) - \text{mean}_S\rvert}{\sigma_S} \end{cases}$ | [163] |
| Local | Graph-based | Graph-Spatial outlier | $\begin{cases} S(x_i) = \lvert x_i - \text{mean}_{j\in N(x_i)} x_j\rvert \\ z_i = \frac{\lvert S(x_i) - \text{mean}_{\mathbf{S}}\rvert}{\sigma_{\mathbf{S}}} \end{cases}$ | [164] |
| Local | Statistics | Residual-Based | $\begin{cases} R(x_i) = \lvert x_i - f(x_i)\rvert \\ z_i = \frac{\lvert R(x_i) - \text{mean}_{\mathbf{R}}\rvert}{\sigma_{\mathbf{R}}} \end{cases}$ | [159] |
| Local | Machine Learning | Autoencoders | $\mathbf{R}(\mathbf{x}) = \lvert\mathbf{x} - \mathbf{f}_{\text{decoder}}(\mathbf{f}_{\text{encoder}}(\mathbf{x}))\rvert$ | [157, 158] |
| Local | GSP | NPGF | $\mathbf{R}(\mathbf{x}) = \mathbf{f}_{\text{NPGF}}(\mathbf{x}) - \mathbf{x}$ | [160] |
| Global | Spectral | PCA Residual-Based | $\lVert\mathbf{x}_i - \mathbf{P}_k\mathbf{P}_k^\top\mathbf{x}_i\rVert$ | [152, 154] |
| Global | Machine Learning | LOF | $\text{LOF}_k(\mathbf{x}) = \frac{\sum_{y\in N_k(x)} lrd_k(y)}{\lvert N_k(\mathbf{x})\rvert lrd_k(\mathbf{x})}$ | [165] |
| Global | Machine Learning | KNN | $\text{KNN}_k(\mathbf{x}) = \frac{1}{k}\sum_{i\in\mathcal{N}(\mathbf{x})} d(\mathbf{x},\mathbf{x}_i)$ | [144] |
| Global | GSP | High-Frequencies | $\lVert\mathbf{U}\mathbf{h}(\boldsymbol{\Lambda})\mathbf{U}^\top\mathbf{x}\rVert$ | [161] |
| Global | GSP | Total Variation | $\text{TV}(\mathbf{x},\mathbf{L}) = \mathbf{x}^\top\mathbf{L}\mathbf{x}$ | [166] |

## Statistical Outlier Detection in Air Pollution Monitoring Networks

Univariate outlier detection models are the most simple and commonly used models in most applications to filter out outliers and ensure good data quality. In particular, unsupervised detection of outliers is difficult in air quality sensors given their data variability and the underlying air pollution patterns [167, 168]. The simplest univariate models are based on statistics, such as the *z-score*, evaluating whether a sensor measurement corresponds to an extreme value of the distribution observed during training [155, 159]. These models only take into account the distribution of the investigated sensor itself, thus explicitly identifying which sensors have anomalous measurements. Instead of inspecting the data distribution, the spatial distribution of the data has also been proposed to compare values at different sensors which are related in some manner. In fact, in the field of spatial outlier detection, a commonly used statistic is the difference between the value of one sensor and the mean or median value of its neighboring sensors, without the need for a training stage [151]. Shekhar *et al.* [164] extended this idea to the graph setting, where the spatial relationships between nodes are described with a graph, instead of using the nearest nodes. Kou *et al.* [163] proposed a spatial outlier detection statistic based on a weighted average of the defined nodes' neighborhoods. A combination of these techniques are residual-based methods, where a reconstruction model is fitted, and the observed value is compared to the value predicted by the model (potentially from neighboring sensors) [159]. The benefit of all these techniques is that they compute a statistic per sensor, so the identification of the sensor that is causing the anomaly is implicit. Yet, most of these models are too simple to capture outliers that depend on other sensors jointly deployed (or other variables).

### MACHINE LEARNING-BASED OUTLIER DETECTION IN AIR POLLUTION MONITORING NETWORKS

More recently, outlier detection models based on ML have gained interest, where multivariate models take into account all features or sensors' measurements at the same time. In multivariate methods, the measurements of all sensors in the network are regarded as a data sample. In this way, ML methods such as local outlier factor (LOF) [169] or k-nearest neighbors (KNN) [144] have been used to detect whether an observation is anomalous. Basically, the distance between the different multivariate samples is used as outlierness statistic since normal samples should be close to observed samples during a training phase. In addition, within the field of neural networks, many studies have been carried out using autoencoders to detect anomalies [157, 158]. As a residual-based model, the outlier detection via autoencoders is performed by inspecting the difference between the reconstructed vector with the observed values. Another common approach is that of spectral decomposition, where principal component analysis (PCA) assumes that normal data patterns are contained in the components that explain more information, and anomalous changes affect the components with less information. Furthermore, Harkat *et al.* [152] showed how to identify which sensor is anomalous from the vector norm of the principal components corresponding to the noise. However, most multivariate models do not naturally identify which sensor (or feature) is causing the anomaly, thereby limiting their use in this field.

### GRAPH SIGNAL PROCESSING-BASED OUTLIER DETECTION

In this particular case, we want to explore outlier detection techniques based on GSP given its applicability to air pollution sensors (see previous chapters 5, 6, and 7). Although there is no specific literature for GSP-based outlier detection techniques for air pollution sensors, there are for other types of sensor networks measuring other phenomena. The main assumption to be used in the GSP setting is that the measurements are smooth with respect to the graph, where similar nodes will be strongly connected by the graph and non-similar nodes will be weakly connected or even disconnected, so anomalous measurements reduce the signal smoothness and increase the amplitude of the high frequencies.

In particular, Egilmez *et al.* [161] showed the analogous GPS-based approach to outlier detection via PCA. They used graph signal filtering and the graph Fourier transform to detect an increase in high frequencies. Therefore, anomalous sensor measurements tend to increase the high-frequency GDFT components' amplitudes. Similarly, Gopalakrishnan *et al.* [166] directly used the signal smoothness, represented by the total variation (TV), as a statistic to determine whether a graph signal is anomalous, or at least different from the signals already observed in the training phase. Unfortunately, as with the multivariate models, those models that treat graph signals as a sample (i.e. treating all measurements observed at a time step as one sample) indicate that the entire sample is anomalous, but give no clue of the sensors that are producing the anomalous values. Graph neural networks have also been used for the detection of outliers in the context of telemetry data [170]. Despite their prediction capability, neural networks are nonconvex models that present optimization difficulties when being fed by little training data [171, 172], requiring specific training methodologies (e.g., transfer learning schemes). Hence, their use is limited in the field of low-cost air pollution sensor networks, where the data available for training are scarce. To overcome this problem, Xiao *et al.* [160] used a convex nonlinear polynomial graph filter (NPGF) to reconstruct graph signals (temperature) and used a threshold on the differences of the reconstructed and the original signal to detect and locate the outliers. Therefore, this residual-based method has proven to be a good alternative to other graph signal processing (GSP) methods, as it is able to locate the abnormal sensors by inspecting the errors produced. Moreover, the NPGF has proven to be a convex alternative to neural networks with better outlier detection capabilities [173]. A shortcoming of these methods lies in the way the graph is constructed. Most of the proposed methods use as shift matrix a matrix whose weights are calculated using a function that decays exponentially with the distance between nodes, and does not take into account the correlation of the measured data [22, 160].

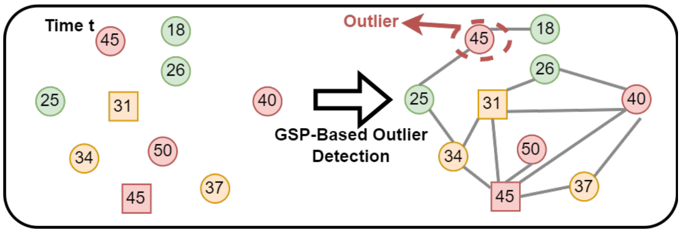


**Figure 8.1:** Graph-based sensor outlier detection for air pollution monitoring networks. A necessary feature is the identification of the sensor that is producing the anomaly in order to carry out different actions, e.g., virtual sensing or sensor replacement.

### 8.1.3. Motivation & Approach

As we have seen previously, graphs offer great capabilities when applied to air pollution sensor networks. In addition, there exist a wide variety of outlier detection methods based on GSP. Therefore, a proposal for unsupervised outlier detection for the detection and localization of outliers in this type of sensor based on GSP is feasible. Moreover, residual GSP-based techniques offer great outlier detection capabilities in the sensor network realm since they can locate which is the abnormal sensor measurement.

However, heterogeneous air pollution monitoring sensor networks present their own challenges. Table 8.2 describes the different needs and characteristics of these networks and the proposed approach to deal with each one of them. For instance, sensors are first calibrated and then deployed, so they may report data at the granularity of the reference instruments, e.g., hourly, which makes the amount of data to train an anomaly detection model limited. Signals may also depend on emission sources such as vehicle traffic or industry. Most studies build graphs based on the geographical distance between nodes, although this approach performs well for some phenomena, networks that measure air pollution and other phenomena can be very complex. Therefore, as shown in chapter 5, the use of graphs learned from the data, resulting in a smooth structure with respect to the measured data, is a good candidate for these air pollution sensor networks, and the one we use for the outlier detection process. This approach is based on the fact of having a network of sensors where there are implicit relationships between the sensors that compose it, so that the different sensors can benefit from the information of other sensors. This idea is in line with Heimann *et al.* [143], where they explained that a dense network of sensors is needed to distinguish local air pollution emissions from regional emissions. To sum up, we rely on having a graph that is smooth with respect to the sensor measurements and that relationships between sensors exist, so sensors deployed in sparse areas without any information from nearby sensors cannot benefit from the network data nor from the graph modeling the network.

**Table 8.2:** Heterogeneous LCS networks for air pollution needs and requirements for outlier detection.

| | **Requirements** | | **Proposed Solution** |
|---|---|---|---|
| **1** | Identification of the anomalous value/sensor. | $\longrightarrow$ | Residual-based model with an indicator function. |
| **2** | Complex relationships between air pollution sensors | $\longrightarrow$ | Graph learned from the data |
| **3** | Little data availability for training | $\longrightarrow$ | Convex GSR model |
| **4** | Non-stationarity of air pollution signals | $\longrightarrow$ | Adaptive outlier detection procedure |

In conclusion, in order to benefit from the advantages of the GSP field and the intrinsic topology defined by a sensor network, we approach the problem of unsupervised sensor outlier detection from a graph-based perspective. Thus, we propose the *Volterra graph-based outlier detection* (VGOD), an unsupervised outlier detection process that first learns the graph encoded by the sensor network data, and then detects the outliers using a residual-based method based on a Volterra-like GSR model [174]. Indeed, this model poses three advances with respect to the literature:

1. Outlier detection methods for air pollution sensor networks are scarce, previously used methods include LOF, KNN, and statistical methods [144, 155, 159]. Here, we propose a more complex graph-based outlier detection mechanism with localization capabilities, which allows the identification of outliers as in the case of drifting

LCSs in heterogeneous air pollution sensor networks.

2. While most previous work on graph-based outlier detection has used a graph distance-based graph [160, 161], we propose to use a graph learned from data. As discussed in chapter 5, graphs learned from network data best describe complex networks than those using functions that decay exponentially with distance, such as low-cost heterogeneous sensor networks for air pollution monitoring. The choice of the shift matrix $\mathbf{S}$ defines the nodes' neighborhoods $\mathcal{N}(x_i) = \{j : \mathbf{S}_{ij} \neq 0\}$ and has impact on the signal reconstruction model as it participates in the shifting of the graph signals.

3. We apply a GSR model based on the classical Volterra series defined by Xiao *et al.* [174]. In fact, Volterra-like models have already been successfully applied to graphs [174, 175]. This model is similar to the NPGF model [160], which has proven a good performance in outlier detection but requires fewer parameters to learn. This means a better computational response when reconstructing the signal.

## 8.2. VOLTERRA GRAPH-BASED OUTLIER DETECTION (VGOD) PROCESS

IN this section, the *Volterra graph-based outlier detection* process is described; from the smoothness-based graph learning, the GSR model based on the Volterra series, to the residual-based outlier detection and the adaptation to deal with non-stationary signals. The sensor network is described by means of a graph $\mathcal{G}$ defined as the triplet $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{S}\}$, where $\mathcal{V} = \{1, \ldots, N\}$ and $N$ is the number of sensors, $\mathbf{S} \in \mathbb{R}^{N\times N}$ is the shift matrix (e.g., $\mathbf{L}$ or $\mathbf{W}$), and $\mathcal{E} = \{e_{ij} : i, j \in \mathcal{V} \wedge S_{ij} \neq 0\}$ is the corresponding set of edges defined by the shift matrix $\mathbf{S}$. Recall that a graph signal is defined over the set of vertices as $x : \mathcal{V} \to \mathbb{R}$.

### 8.2.1. GRAPH LEARNING: SMOOTHNESS-BASED

The first step consists of learning the underlying implicit relationships between the sensors $\{x_i : i \in \mathcal{V}\}$ composing the network by discovering the graph $\mathcal{G}$ that best fits the network data. So, given a set of training graph signals in matrix form $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_P]$, where $\mathbf{X} \in \mathbb{R}^{N\times P}$ and $P$ is the number of signals, the goal is to learn a graph shift matrix $\mathbf{S}$ (e.g., $\mathbf{S} = \mathbf{L}$, $\mathbf{S} = \mathbf{W}$, or $\mathbf{A}$) that fits the training data according to some criterion. As mentioned in chapter 5, it is interesting to learn the Laplacian matrix $\mathbf{L}$ since its eigendecomposition provides the Fourier basis for performing the GDFT, and its quadratic form $\mathrm{TV}(\mathbf{x}, \mathbf{L}) = \mathbf{x}^\top \mathbf{L}\mathbf{x}$ provides the signal smoothness criterion. Thus, signal smoothness is a good criterion for learning a graph that maintains coherence between the measurements of the different sensors given their complex relationships, so a graph smooth with respect to the measures. Therefore, edges between sensors measuring similar or related measures are encouraged, while edges between dissimilar sensors are penalized. As we have seen throughout the thesis, we have opted for the model defined by Dong *et al.* [104] since it promotes smooth graphs according to the training data. Therefore, the graph learning optimization problem[1] is defined as:

$$\begin{aligned} \min_{\mathbf{L},\mathbf{Y}} \quad & \|\mathbf{X}-\mathbf{Y}\|_F^2 + \alpha\, tr(\mathbf{Y}^\top \mathbf{L}\mathbf{Y}) + \beta \|\mathbf{L}\|_F^2 \\ \text{s.t.} \quad & tr(\mathbf{L}) = N, \\ & L_{ij} = L_{ji} \leq 0, \quad i \neq j, \\ & \mathbf{L}\cdot\mathbf{1} = \mathbf{0}. \end{aligned} \tag{8.1}$$

Where $\mathbf{Y} \in \mathbb{R}^{N\times P}$ is a filtered version of the data matrix $\mathbf{X}$, and $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$ are the model's hyperparameters (see chapter 5 for further details). The graph $\mathcal{G}$ is only required to be learned once, using the training data $\mathbf{X}$, given that it is assumed that deployed sensors will work well for a certain period of time right after the deployment and those relationships are considered to be the correct ones given that no sensor worsen is present. Thus, the temporal distribution of the measurement may change, but the relationship between the sensors, when working properly, is expected to be maintained over time, so a change in anthropogenic emissions may result in outliers.

It is important to emphasize that in the case of LCSs, which present quality variability and need to be calibrated before deployment, a data-driven graph has an advantage over a distance-based graph. The graph where the shift matrix $\mathbf{S}$ is constructed from the distances $d_{ij}$ can actively include a poor quality sensor $x_i$, since the weight $S_{i\cdot}$ does not depend on how well the sensor is calibrated. In contrast, a poorly calibrated sensor $x_i$ will participate little in the outlier detection since it will be poorly related to the other well-performing sensors ($L_{ij}\approx 0$), resulting in a better definition of the neighborhood $\mathcal{N}(x_i)$ and weights $\mathbf{L}_{i\cdot}$ by a Laplacian $\mathbf{L}$ learned directly from the data.

## 8.2.2. GRAPH SIGNAL RECONSTRUCTION: VOLTERRA SERIES

An important part of the outlier detection models based on the residuals, let us define the residual of a signal $\mathbf{x}$ as $\mathbf{R}(\mathbf{x}) = \mathbf{x} - \mathbf{f}(\mathbf{x})$, is the signal reconstruction model $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$. In the VGOD mechanism, we propose the use of a model based on the Volterra series[2], recently defined by Xiao *et al.* [174]. For understanding purposes, we will now explain the relationship of the used model with the classical Volterra discrete model. The classical discrete Volterra model can be defined as:

$$y(t) = h_0 + \sum_{d=1}^{D} \sum_{\tau_1=a}^{b} \cdots \sum_{\tau_d=a}^{b} h_d(\tau_1,\ldots,\tau_d) \prod_{j}^{d} x(t-\tau_j) \tag{8.2}$$

Where $x(t) \in \mathbb{R}$ is a discrete signal defined at different time steps $t \in \mathbb{N}$, $h_d(\cdot) \in \mathbb{R}$ are the different learnable parameters of the model, $D \in \mathbb{N}$ is the order of the Volterra series, and $x(t-\tau_j)$ can be seen as a signal shift by $\tau_j$ as in classical discrete signal processing. This model is known for being nonlinear and memory-based since the output $y(t)$ depends on the inputs at previous times $x(t-\tau)$ in a nonlinear way. Similarly, Franz *et al.* [176] proposed the kernelized version of the Volterra series, where the vector

[1] The authors of [104] provide the implementation of the graph learning problem.

[2] A python implementation of the proposed VGOD mechanism along with an implementation of the Volterra model [174] and the graph learning problem [104] using the CVXPY library are available at https://bitbucket.org/sans-rg/ieee-tnse-outlier-detection/.

$\mathbf{x} = [x(t-a),\dots,x(t-b)]^\top$ contains all shifted versions of the signal to take into account, being $0 \le a \le b$. Then, the Volterra series can be represented in form of a basis expansion $\boldsymbol{\phi}(\mathbf{x}) = [\phi_0(\mathbf{x}),\dots,\phi_D(\mathbf{x})]^\top$, where $\boldsymbol{\phi}_D(\mathbf{x})$ are the $D$-th order monomials of the signal and its shifted versions.

Equivalently, the notion of signal shift [103] has been extended to the GSP paradigm by applying a graph shift matrix $\mathbf{S}$ to a graph signal $\mathbf{x}$, $\mathbf{x}^{[1]}$=$\mathbf{Sx}$. The graph weight matrix $\mathbf{W}$ and the Laplacian matrix $\mathbf{L}$ have been widely used as the graph shift operator in the literature [21, 103]. In the specific case of a circular graph, the graph shift is equivalent to the signal shift in discrete signal processing. In this way, we define the Volterra-like graph model in terms of a linear combination of nonlinear basis functions $\boldsymbol{\psi}_{.....} : \mathbb{R}^N \to \mathbb{R}^N$, this model describes the interactions between the signal at node $x_i$ with the shifted versions of the signal at that node $(\mathbf{L}^j\mathbf{x})_i$ in the following way:

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}_0 + \sum_{d=1}^{D} \sum_{k_d=0}^{K-1} \cdots \sum_{k_1=0}^{K-1} h_d(k_d,\dots,k_1)\boldsymbol{\psi}_{d,k_d,\dots,k_1}(\mathbf{x}) \tag{8.3}$$

Where $D$ is the order of the Volterra series, $K \in \mathbb{N}$ is the maximum number of shifts to be applied (model depth), $\mathbf{h}_0 \in \mathbb{R}^N$ and $h_d(k_d,\dots,k_1) \in \mathbb{R}^{K^d}$, for $d = 1,\dots,D$, are the parameters to be learned, and $\boldsymbol{\psi}_{d,k_1,\dots,k_D}(\mathbf{x})$, for $d = 1,\dots,D$, are the basis functions representing the interactions defined as:

$$\begin{cases} \boldsymbol{\psi}_{1k_1}(\mathbf{x}) = \mathbf{L}^{k_1}\mathbf{x} \\ \boldsymbol{\psi}_{2k_2k_1}(\mathbf{x}) = (\mathbf{L}^{k_2}\mathbf{x}) \odot \boldsymbol{\psi}_{1k_1}(\mathbf{x}) \\ \quad\vdots \\ \boldsymbol{\psi}_{Dk_D\dots k_1}(\mathbf{x}) = (\mathbf{L}^{k_D}\mathbf{x}) \odot \boldsymbol{\psi}_{(D-1)k_{D-1}\dots k_1}(\mathbf{x}) \end{cases} \tag{8.4}$$

Where $\odot$ is the Hadamard product and $k_i$= $0,..,K-1$. For instance, the second order interactions take into account the interactions between the values at one node $x_i$ (and its shifted versions) and the values at that node in its shifted versions $(\mathbf{L}^j\mathbf{x})_i \in \mathbb{R}$. As an example, the value at the i-th vertex $x_i$ for the third order expansion is computed as:

$$\begin{aligned} x_i =& h_{0i} + \sum_{k_1=0}^{K-1} h_1(k_1)(\mathbf{L}^{k_1}\mathbf{x})_i \\ &+ \sum_{k_2=0}^{K-1}\sum_{k_1=0}^{K-1} h_2(k_2,k_1)(\mathbf{L}^{k_2}\mathbf{x})_i(\mathbf{L}^{k_1}\mathbf{x})_i \\ &+ \sum_{k_3=0}^{K-1}\sum_{k_2=0}^{K-1}\sum_{k_1=0}^{K-1} h_3(k_3,k_2,k_1)(\mathbf{L}^{k_3}\mathbf{x})_i(\mathbf{L}^{k_2}\mathbf{x})_i(\mathbf{L}^{k_1}\mathbf{x})_i \end{aligned} \tag{8.5}$$

The expression above is the same for all vertices $i \in \mathcal{V}$, it actually resembles the expression of the classical Volterra model for discrete signals, but in this case, since we are dealing with a graph signal $\mathbf{x}$, the different parameters $h.(\cdot)$ are shared across all nodes and applied to graph signals $\mathbf{x}$.

Finally, we need to define a training scheme and a loss function Loss$(\mathbf{x},\mathbf{f}(\mathbf{x}))$. The GSR model used for outlier detection is trained to recover the original signal $\mathbf{x}$ given a perturbed version of it $\tilde{\mathbf{x}}=\mathbf{x}+\epsilon$, acting as a denoising model, and the following convex objective function is minimized to find the model's coefficients $\mathbf{h}$:

$$\min_{\mathbf{h}} \sum_{i}^{N_{\text{train}}} \|\mathbf{x}_i - \mathbf{f}(\tilde{\mathbf{x}}_i)\|_2^2 \tag{8.6}$$

Where $N_{\text{train}} \in \mathbb{N}$ is the number of training signals. Thus, the loss function used corresponds to the residual sum of squares (RSS). When an unusual perturbation is present in a signal the model will incur a larger error, being capable of identifying the anomalous node given the residuals $\mathbf{R}(\mathbf{x})=|\mathbf{x}-\mathbf{f}(\mathbf{x})|$. This problem constitutes a convex optimization problem since it is linear with respect to the coefficients of the model $\mathbf{h}$, so its optimization is easier than in nonconvex models such as neural networks. This is of special interest in cases such as the monitoring of air pollution heterogeneous sensor networks, where the training periods used to learn the models may be relatively small. We also expect the choice of the shift operator $\mathbf{S}$ to have a very high impact on the model. In general, for many signals, the distance between nodes does not have to be a good choice for generating the shift operator.

This training scheme could be seen as a form of denoising autoencoder, where we have an overcomplete autoencoder whose code dimension is the number of basis expansions, assumed to be larger than the number of nodes, and noise is introduced so that the model learns the latent patterns of the data and avoids learning the identity function, acting as a regularization. Figure 8.2 shows an example of the use of the Volterra-like GSR for the detection of outliers in a real $O_3$ graph signal.

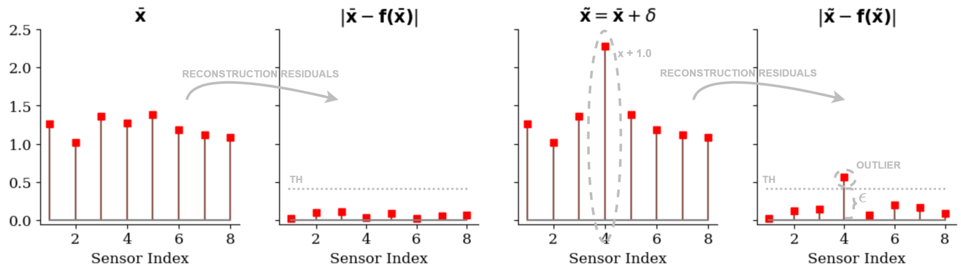


**Figure 8.2:** These figures represent an example of the use of the Volterra-like GSR to detect outliers. The left figure shows a standardized graph signal $\bar{\mathbf{x}}$ while its next figure shows the residuals of reconstructing the unaltered graph signal $\bar{\mathbf{x}}$. Then, a perturbation ($\delta = 1.0$) of one standard deviation is introduced to sensor number four and now (right figure) the reconstruction residuals indicate the introduced outlier since the error is above the predefined threshold $TH$.

## 8.2.3. OUTLIER DETECTION: INDICATOR FUNCTION

Once we have learned the graph Laplacian $\mathbf{L}$ and we have trained the GSR model $\mathbf{f}(\cdot)$, we need to identify and locate outliers. The objective is not only to evaluate whether the graph signal is an outlier $\mathbf{R}(\mathbf{x})$ but to locate which sensor/measurement is an outlier $R_i(\mathbf{x})$.

The identification of outliers is done through the inspection of the signal reconstruction residuals $\mathbf{R}(\mathbf{x}) \in \mathbb{R}^N$, i.e., the difference between the observed signal $\mathbf{x}$ and the reconstructed signal $\mathbf{f}(\mathbf{x})$:

$$R_i(\mathbf{x}) = \left| x_i - f_i(\mathbf{x}) \right| \qquad , \forall i \in \mathcal{V} \tag{8.7}$$

Normal samples are supposed to have small residuals since the model has been trained with a similar pattern, while abnormal samples tend to have larger residuals, as they deviate from the normal pattern seen during the training. Then, using a thresholding value, $TH \in \mathbb{R}$, an indicator function can be implemented:

$$I_i(\mathbf{x}) = \begin{cases} 1 & , R_i(\mathbf{x}) > TH \\ 0 & , R_i(\mathbf{x}) \leq TH \end{cases} \qquad , \forall i \in \mathcal{V} \tag{8.8}$$

Where $I_i(\mathbf{x})$=1 indicates that the sensor $x_i$ is the suspicious one. Other outlier scoring metrics can be used to detect outliers, for instance, if we were not interested in locating the error, but in indicating whether the whole sample is an outlier, using the $\mathrm{l}_2$-norm of the residual $\|\mathbf{R}(\mathbf{x})\|_2$ could be useful to find abnormal graph signals. The threshold value $TH$ can be defined depending on the application target performance, i.e., the maximum false positive rate (FPR) or the minimum true positive rate (TPR) required by the application [160, 161].

In order to define the optimal threshold, once the GSR model $f(\cdot)$ has been trained, we select a set of training samples $\mathbf{X}_{th}$, which are normal, and we add a small perturbation $\epsilon$ to a random subset of samples selecting a random perturbed node per sample, $\tilde{\mathbf{X}}_{th} = \mathbf{X}_{th} + \boldsymbol{\epsilon}$. Five repetitions are performed, then, inspecting the residuals $\mathbf{R}(\tilde{\mathbf{X}}_{th})$ the threshold $TH$ is selected via a grid search so that the true positive rate is maximized without exceeding a desired maximum false positive rate $fpr$:

$$\begin{aligned} \max_{TH \in \mathbb{R}} \quad & \mathrm{TPR}(\mathbf{R}(\tilde{\mathbf{X}}_{th}), TH) \\ \text{s.t.} \quad & \mathrm{FPR}(\mathbf{R}(\tilde{\mathbf{X}}_{th}), TH) \leq fpr \end{aligned} \tag{8.9}$$

## 8.2.4. Adaptive algorithm: VGOD

Once we have all the algorithm components we can define the VGOD. We emphasize that both for graph learning and signal reconstruction the data are standardized. Finally, although the GSR model can be trained correctly using a few weeks of data, air pollution data can suffer from a problem known as data set shift, commonly known as non-stationarity in the field of time series analysis. This is because the data present in the training set may not be representative of the testing set (or the posterior deployment conditions), e.g., mean concentrations may vary from month to month. This phenomenon is also known as *concept drift* in the field of sensor calibration [12]. Therefore, special care must be taken when applying the GSR model. A common approach to overcome this problem consists of updating the detection model periodically with new data [177]. For example, we can incorporate into the training set the samples predicted as normal during the test phase, and retrain the signal reconstruction model every time we have $W$ normal samples. This increases the computational burden, but as we are solving a convex optimization problem the increase remains bounded. However, it is important to keep the complexity of the models limited, i.e., their depth or number of learnable parameters, so that it is feasible to retrain periodically. Recall that the threshold $TH$ used by the indicator function can also be recomputed along with the signal reconstruction method to better adapt to the non-stationarity nature of the data.

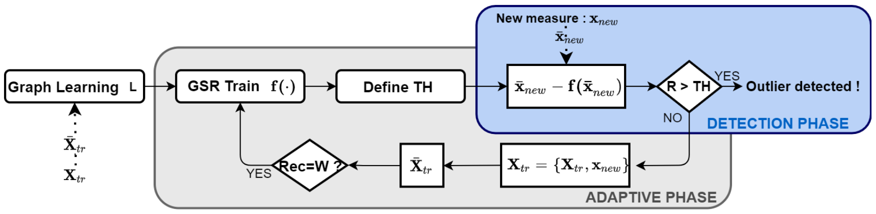


**Figure 8.3:** General view of the VGOD process. First, the graph is learned, then the GSR model and the threshold $TH$ are updated adaptively. Finally, outliers are detected by inspecting the signal reconstruction residuals $\mathbf{R}(\mathbf{x})$.

**Definition 17** *A* ***non-stationary*** *time series is one that is not strictly stationary nor weakly stationary. Lets us define the sensor's readings $x_t$ for $t = 1,\dots,T$ as a realization of a stochastic process, where each $x_t$ is a realization of a random variable. Thus, a weak-sense or wide-sense stationary process is the one whose first moment and correlation function is invariant to time, and the second moment remains bounded. Thus, in order for a process to be non-stationary it must not meet any of these conditions:*

$$E[x_{t_1}] = E[x_{t_1+\Delta}], \qquad \forall t_1, \Delta \in \mathbb{N} \tag{8.10}$$

$$E[x_{t_1} x_{t_2}] = E[x_{t_1+\Delta} x_{t_2+\Delta}] \qquad \forall t_1, t_2, \Delta \in \mathbb{N} \tag{8.11}$$

$$E[x_t^2] < \infty \tag{8.12}$$

**Algorithm 4** Volterra Graph-Based Outlier Detection(VGOD).

```
Input: $\{\alpha, \beta, \mathbf{X}_{tr}, K, D, fpr, \epsilon, W\}$
1: $\bar{\mathbf{X}}_{tr} \leftarrow$ Standardization($\mathbf{X}_{tr}$)
2: $\mathbf{L} \leftarrow$ Graph_Learning($\alpha, \beta, \bar{\mathbf{X}}_{tr}$)
3: $\mathbf{f}(\cdot) \leftarrow$ Reconstruction_Model($\bar{\mathbf{X}}_{tr}, K, D$)
4: $TH \leftarrow$ Define_Threshold($\mathbf{f}(\cdot), \bar{\mathbf{X}}_{tr}, fpr, \epsilon$)
5: $rec \leftarrow 0$
6: while $\mathbf{x}_{new}$ do                                  ◁ Detection Phase
7:     $\bar{\mathbf{x}}_{new} \leftarrow$ Standardization($\mathbf{x}_{new}$)
8:     $\mathbf{R}(\bar{\mathbf{x}}_{new}) \leftarrow |\bar{\mathbf{x}}_{new} - \mathbf{f}(\bar{\mathbf{x}}_{new})|$
9:     if $R_i(\bar{\mathbf{x}}_{new}) > TH$ then
10:        $x_{new,i}$ is outlier !
11:    else                                                   ◁ Adaptive Phase
12:        $\mathbf{X}_{tr} \leftarrow \{\mathbf{X}_{tr}, \mathbf{x}_{new}\}$
13:        $rec \leftarrow rec + 1$
14:        if $rec = W$ then
15:            $\bar{\mathbf{X}}_{tr} \leftarrow$ Standardization($\mathbf{X}_{tr}$)
16:            $\mathbf{f}(\cdot) \leftarrow$ Reconstruction_Model($\bar{\mathbf{X}}_{tr}$,$K$,$D$)
17:            $TH \leftarrow$ Define_Threshold($\mathbf{f}(\cdot), \bar{\mathbf{X}}_{tr}, fpr, \epsilon$)
18:            $rec \leftarrow 0$
19:        end if
20:    end if
21: end while
```

Figure 8.3 summarizes the overall outlier detection process developed while algorithm 4 gives a precise description of the process[3]. The outlier detection process parameters are; $\{\alpha, \beta\}$ hyperparameters to control the graph learning algorithm, the training data $\mathbf{X}_{tr}$, model depth $K$, model order $D$, acceptable maximum false positive rate $fpr$ to define the threshold, the perturbation $\epsilon$ to be introduced to define the threshold, and the model updating window size $W$. $\{\alpha, \beta, K, D\}$ are hyperparameters that are obtained based on the training data $\mathbf{X}_{tr}$, while $\{fpr, \epsilon, W\}$ are user-defined parameters that depend on the specific data domain on which the algorithm is used.

## 8.3. EXPERIMENTAL EVALUATION

THIS section evaluates the performance of the proposed VGOD algorithm (Section 8.2) applied to real air quality monitoring data sets. The proposed model is compared with other state-of-the-art outlier detection methods, some with and some without localization capabilities. Actually, for comparison purposes we use; *i)* outlier detection algorithms that do not allow localization, such as the frequency-based GSP [161], the local outlier factor (LOF) [169], and the k-nearest neighbors (KNN) [144], and *ii)* models based on reconstruction residuals and GSP, such as the linear graph filter (LGF, $\mathbf{f}(\mathbf{x})=\mathbf{h_0}+\sum_{i=0}^{K-1} h_{1i}\mathbf{S}^i\mathbf{x}$) [22] and the third order NPGF model with a distance-based graph [160]. NPGF is the model that most closely resembles VGOD, since it is a residual-based model based on GSP. In this case, it is also an adaptive method, which uses a shift matrix $\mathbf{S}$ based on geodesic distances, and uses a more complex optimization model in terms of the number of learnable parameters. Check Table 8.1 to review how the different outlier scores are computed.

The following sections describe the data and the different experiments performed:

(A) *Experimental data*: the different data sets used for outlier detection experiments are introduced. These data sets include a real heterogeneous LCS network and two governmental reference station sensor networks of small and medium size.

(B) *Model training & Selection*: the different models' hyperparameters are described for both global and residual-based models, as well as the selection process of their hyperparameters and the different training schemes used.

(C) *Outlier detection over the training set*: outliers are simulated on the training of the data set *D.1*. The different models are applied non-adaptively on the training data set, i.e., they are trained using the training data set and the detection is also performed on the training data set. This simulates the best case, where the data distribution in the detection phase does not change. Such experiments allow us to analyze which parameters internal to the models will be used later in testing, e.g., the model depth K.

(D) *Outlier detection over the testing set*: outliers are simulated in the testing of the data sets *D.1* and *D.2*. The models are applied adaptively, as shown in the VGOD

[3]A python implementation of the VGOD as well as the graph learning method used are available at https://bitbucket.org/sans-rg/ieee-tnse-outlier-detection/.

algorithm 4 since the data distribution changes over time (i.e. non-stationary time series).

(E) *Sensor drift detection*: a sensor drift is simulated in the testing set of data set *D.3*. A malfunctioning sensor, which is a common situation with sensor network deployments, can be simulated since the data set *D.3* contains both LCSs and reference stations. Again, the models are applied adaptively to detect and locate the drifting sensor.

(F) *VGOD scalability*: the scalability of the two best performing graph-based models, the GSR model using the Volterra-based model and the NPGF model, are compared.

The data sets are divided into 60% of the data for training, and the remaining 40% for testing. Thus, mimicking the real case where the outlier detection model is trained just after the sensor network deployment and applied sequentially throughout the deployment lifetime. The perturbations $\delta$ added to simulate the outliers have no units since these perturbations are introduced to the standardized data. Indeed, it is fairer to add perturbations proportional to the standard deviation of each of the sensor nodes, which can be quite different.

### 8.3.1. EXPERIMENTAL DATA

To study the performance of the proposed outlier detection process for air pollution data sets, we use two different types of data. First, we use two data sets provided by the Spanish government consisting of forty-three nodes deployed in the Catalonia area, and fourteen nodes deployed in the Barcelona city area. These data are public and can be downloaded at the Catalonia open data web page[4]. In this way, we can simulate outliers and assess how outlier detection works for tropospheric ozone sensor networks. Secondly, we use a data set collected by a heterogeneous network consisting of five LCSs and three reference stations, deployed during the summer of 2017 for the H2020 Captor project, to detect drifting sensors. Summarizing, we experiment with the following three data sets:

1. *Spanish air pollution reference station network for $O_3$ for Barcelona city area*: this data set is composed of fourteen nodes, capturing hourly tropospheric ozone data between the months of January and May of 2019, with a total of 2798 samples.

2. *Spanish $O_3$ reference station network for Catalonia*: this data set is made up of forty-three nodes in the area of Catalonia capturing hourly tropospheric ozone data between the months of January and February of 2021, with a total of 1076 samples.

3. *H2020 Captor network [9]*: this data set is formed by eight nodes, five LCSs and three reference stations, deployed in the area of Catalonia (Spain) during the summer of 2017 to capture half-hourly tropospheric ozone concentration levels. This data set has a total of 2368 samples.

[4]https://analisi.transparenciacatalunya.cat/en/Medi-Ambient/Qualitat-de-l-aire-als-punts-de-mesurament-autom-t/tasf-thgu

**Table 8.3:** Statistics of the data sets used.

| Data Set Label | Pollutant | # Nodes | # Samples | Period | Resolution | Mean ($\mu$gr/m$^3$) | Pooled STD. ($\mu$gr/m$^3$) |
|---|---|---|---|---|---|---|---|
| *D.1* | $O_3$ | 14 | 2798 | 2019/01/01 - 2019/05/31 | 1 h | 45.32 | 31.78 |
| *D.2* | $O_3$ | 43 | 1076 | 2021/09/01 - 2021/11/01 | 1 h | 50.79 | 25.72 |
| *D.3* | $O_3$ | 8 | 2368 | 2017/06/18 - 2017/09/01 | 30 min | 68.82 | 35.14 |

These data sets are similar to those used in previous chapters and are representative of air quality monitoring networks. The first two data sets correspond to governmental reference stations, while the third corresponds to a hybrid network of governmental reference stations and low-cost sensors. Table 8.3 shows the statistical characteristics of the three data sets.

In addition, heterogeneous data from the Captor network allows us to explore one of the most important outlier detection applications in sensor networks, the detection of drifting or malfunctioning sensors. Large air pollution monitoring sensor networks can be reduced to smaller subnetworks using clustering techniques (as seen in chapter 5). This reduces the computational cost, without losing the ultimate goal of the graph-based method, which is to detect anomalies using neighboring nodes selected with an algorithm that learns the connectivity of the graph based on a smoothness criterion.

## 8.3.2. Model training & Selection

In this section, we explain the hyperparameters required for each model, how to train the different models, from the graph learning and the signal reconstruction, to how to define the thresholds. As mentioned earlier, we assume that the sensors have no outliers during the graph learning and signal reconstruction training phases since these are expected to work well at least during a period of the sensor network deployment, i.e., poorly calibrated sensors may not be deployed. This allows casting the outlier detection problem to an unsupervised novelty detection problem, where we know the distribution of normal data. The different resulting hyperparameters for the different models are summarized in Table 8.4.

**Table 8.4:** Models' hyperparameters along with the input data required for each of them.

| Model | Inputs | | Hyperparameters |
|---|---|---|---|
| | Shift Matrix S | Data | |
| Linear Graph Filter (LGF) [22] | **W** | $\mathbf{X}_{tr}$ | Depth ($K$) |
| Third Order NPGF [160] | **W** | $\mathbf{X}_{tr}$ | Depth ($K$) |
| VGOD | **L** | $\mathbf{X}_{tr}$ | $\{\alpha,\beta\}$, Depth ($K$), Order($D$) |
| Frequency-based GSP [161] | **L** | $\mathbf{X}_{tr}$ | Variance kept ($\tau$) |
| Local Outlier Factor (LOF) [169] | | $\mathbf{X}_{tr}$ | Neighbors ($N_{lof}$) |
| K-nearest Neighbors (KNN) [144] | | $\mathbf{X}_{tr}$ | Neighbors ($N_{knn}$) |

### Graph learning

Residual-based models require a shift matrix **S** to perform the GSR, for example, the weight matrix **W** or the Laplacian matrix **L**. The state-of-the-art models LGF and NPGF use a distance-based weight matrix **W** as defined in [22], to define the relationships between the different network sensors:

$$W_{ij} = \frac{e^{-d_{ij}^2}}{\sqrt{\sum_{n\in\mathcal{N}(x_i)} e^{-d_{in}^2} \sum_{m\in\mathcal{N}(x_j)} e^{-d_{jm}^2}}}, \quad \forall i,j \in \mathcal{V} \tag{8.13}$$

The VGOD process uses a Laplacian matrix learned from the network data using a signal smoothness criterion [104], i.e., based on the data collected during the training. As seen in chapter 5, air pollution sensor networks encode highly complex relationships, which are best described by a data-driven graph, thus learning the Laplacian matrix implies learning more meaningful relationships. The Laplacian matrix $\mathbf{L}$ is learned from the data using the training set $\mathbf{X}_{tr}$ and the values of the hyperparameters $\{\alpha,\beta\}$, which control the sparsity of the graph in the smoothness-based graph learning criterion. In this case, we choose a graph with a medium density, so that smoothness is promoted and all nodes have enough neighboring information to detect the outliers. In fact, different graph topologies have been tried with different levels of sparsity and the one with medium sparsity performed the best although there is little difference. Depending on the data set, the user can perform a grid search over the graph learning hyperparameters to fine-tune the graph to be learned. For further information on how to select the learn the graph Laplacian $\mathbf{L}$ and the effect of the hyperparameters $\{\alpha,\beta\}$ refer to chapter 5.

### GRAPH SIGNAL RECONSTRUCTION

The second step is to train the signal reconstruction models to learn to remove noise, as if it were a signal denoising task, by taking the training set $\mathbf{X}_{tr}$ and adding artificial noise $\epsilon$ to a variable percentage of nodes for each signal, so using as input an artificially perturbed version $\tilde{\mathbf{X}}_{tr}$ of the training data, as mentioned in section 8.2.2. Regarding the hyperparameters of the GSR models, we have the filter depth $K$, which indicates the maximum number of shifted versions of the signal taken into account and therefore controls the model complexity. As a rule of thumb, the maximum value of $K$ is set at most to the degree $N_m$ of the minimal characteristic polynomial of the shift matrix $\mathbf{S}$, that is $K \leq N_m \leq N$. As the graph diameter in dense graphs is low, and we want the models to extrapolate to the test set, we explore simple models with small depths. The model based on the Volterra series [174] also includes the model order $D$, which indicates the maximum degree of interactions to take into account. This parameter drastically affects the complexity of the model as well as the number of parameters of the model. But in this experiment, we only take into account the third order model, $D$=3, so that we can fairly compare this model with the third order NPGF shown in [160]. The best filter depth $K$ value is found by adding artificial outliers randomly in the training, and obtaining the $K$ that corresponds to the best true positive rate, best false positive rate, and the least complex model so that it generalizes better, a common procedure used in the literature [160, 161] and similar to the threshold $TH$ selection procedure. The selection of the best $K$ is done according to the outlier detection results over the training set, next section 8.3.3.

Once the GSR models are trained, we find the corresponding threshold $TH$ for each model above which the residual $R_i(\mathbf{x})$ considers that sensor $x_i$ is a possible outlier. The selection of the threshold can be done in different ways, but the most common choice depends on the false positive rate (FPR) or false alarm, and the true positive rate (TPR) or probability of hit required by the application. Since the outlier detection process is used to maintain the network data quality, the TPR is maximized and the acceptable rate of false positives is set to 10%. This is achieved by introducing artificial outliers in a set of training samples and selecting the $TH$ that maximizes the TPR (section 8.2.3). In the paradigm of sensor data quality, it is important to have a high sensitivity (true positive rate) and the fact of having any false positive does not imply any costly action (e.g. sensor replacement). The decision on the value of $W$ is shown in section 8.3.4, where the adaptive application of the different models (adaptive phase, algorithm 4) is explained.

$$\mathbf{X}_{tr} \overset{(1)}{\Longrightarrow} \tilde{\mathbf{X}}_{tr} = \mathbf{X}_{tr} + \boldsymbol{\epsilon} \overset{(2)}{\Longrightarrow} \min_{\mathbf{h}} \text{RSS}(\mathbf{X}_{tr}, \mathbf{f}(\tilde{\mathbf{X}}_{tr})) \overset{(3)}{\Longrightarrow} \begin{array}{rl} \max\limits_{TH} & \text{TPR}(R(\mathbf{X}_{tr} + \boldsymbol{\epsilon}_2), TH) \\ \text{s.t.} & \text{FPR}(R(\mathbf{X}_{tr} + \boldsymbol{\epsilon}_2), TH) \leq 0.1 \end{array}$$

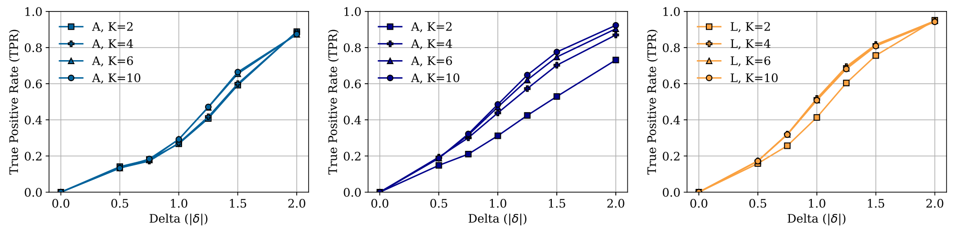
### Training of global models

As for the global models, the frequency-based GSP needs the Laplacian $\mathbf{L}$, the training data $\mathbf{X}_{tr}$, and the $\tau$ hyperparameter, which indicates the amount of variance retained by the selected frequencies as the normal components of the signal. The LOF uses the training data $\mathbf{X}_{tr}$, and the number of neighbors $N_{lof}$ to take into account to compute the outlier score. Finally, the KNN uses the training set $\mathbf{X}_{tr}$ as a dictionary to compute distances, and the number of neighbors $N_{knn}$ to consider when computing the distance. The hyperparameters and the thresholds for these models are selected in the same way as in the case of residual-based models, by introducing artificial outliers to a set of training samples and selecting the best set of hyperparameters promoting models with lower complexity to avoid overfitting.

## 8.3.3. Training set performance

This section shows the results of applying the different models in a non-adaptive way over the training so that the models are trained and applied to the same data. This process allows exploring the best case, where the data distribution does not change as the opposite of the adaptive case, and allows different hyperparameters to be examined and set as a baseline for testing. This step is important since it is similar to the adaptive process where the models are trained adaptively and the threshold $TH$ is also recomputed periodically.

The different models are trained with different possible values for their hyperparameters, Table 8.4. As mentioned in the previous section 8.3.2, some of the VGOD hyperparameters have already been fixed to reduce the complexity of the model; *i)* a graph with medium density, which can be defined a priori and *ii)* $D = 3$ to be able to compare with the NPGF model and to bound the complexity of the model. Thus, the only tested parameter for the VGOD is the filter depth $K$. To evaluate the outlier detection performance over the training set, 30% of the training set is perturbed by adding different outlier perturbations $\delta$ at random, and ten repetitions are performed. The sign (+1,-1) of these perturbations is also selected at random. We also use this perturbation mechanism for the testing set results. For illustrative purposes, only the results over the first data set *D.1* are shown given that some global models are known to have difficulties with high-dimensional outlier detection problems (data set *D.2*).



(a) Outlier detection using the LGF. (b) Outlier detection using the third order NPGF. (c) Outlier detection using the VGOD process with the third order Volterra-based graph model.

**Figure 8.4:** Average true positive rates results for ten different repetitions and different perturbation magnitudes ($|\delta|$) using the residual-based models.

Figures 8.4.a), b), and c) show the average TPR for the different residual-based models and a FPR of ~10%. As for the depths of the models, it can be seen how from $K$=4 onwards the improvement for all three models is very little. Therefore, $K$=4 seems to be a good choice to keep the models' complexity bounded. In fact, for perturbations of one standard deviation ($|\delta|$=1) the LGF obtains a 27% TPR, the NPGF obtains around a 44% TPR, and in the case of the VGOD the TPR is 52%. Therefore, using the combination of a graph learned from the data and the Volterra-based reconstruction outperforms the other two residual-based models using the distance-based weight matrix. Looking at the VGOD results, Figure 8.4.c), it is observed that with a perturbation of $|\delta|$=1.0 standard deviation obtains a TPR of 52%, with perturbations of $|\delta|$=1.25 standard deviations the TPR reaches 69%, and with perturbations of $|\delta|$=2.0 standard deviations it can detect almost all the outliers with a TPR of 95%. Recall that in this case, one standard deviation is approximately 31 $\mu$gr/m$^3$, this means that measurements with deviations around 39 $\mu$gr/m$^3$ (1.25 $\sigma$) are effectively detected with a 69% TPR. This value is particularly good since air pollution sensing nodes exhibit large variability both with respect to their own measurements and with respect to the measurements of the other sensors in the network, so stating that a measurement is an outlier can be challenging. Hence, this outlier detection process provides the necessary tools to detect outlying measures and sensors. Indeed, the proposed model can slightly improve the results of the NPGF even when this has larger depths ($K > 4$).

Now, Figure 8.5 compares the TPR for the residual-based models, with $K$=4, with the TPR for the three global models (frequency-based GSP, LOF, and KNN) with their best hyperparameters. It is clearly seen that the nonlinear residual-based models perform better than the global models. Indeed, VGOD is able to improve the detection rate by more than a 10% for perturbations greater than $|\delta|$=0.75 standard deviations. The NPGF improves the results of LOF by a 5% of TPR for perturbations in the range of $|\delta|$=0.5-1.5 standard deviations. The frequency-based GSP has a similar performance to the LGF, since they both use the high-frequency components of the signal to detect outliers, but the LGF performs slightly better than the frequency-based GSP for high-magnitude perturbations. The KNN is observed to have a similar performance to GSP and LGF, with

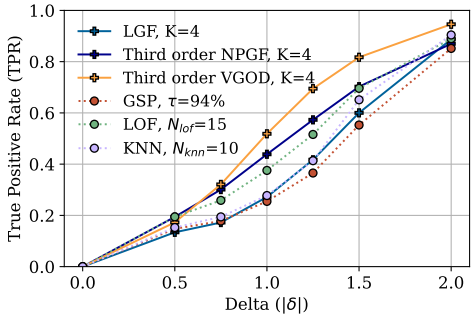


**Figure 8.5:** Average true positive rate for the different models for data set *D.1*.

slightly higher detection capabilities for large perturbations. In addition to their lower detection capabilities, global models are not able to localize which one of the sensors in the network is producing the outlier, and this limits their application in real sensor network deployment scenarios. In the following section, we show the detection results using the adaptive algorithm, as well as the localization abilities of the models for two different data sets, *D.1* representing a small-size network and *D.2* representing a mid-size network.

## 8.3.4. Test set performance

Once we have seen how the different models work on the training, let's check how they work when applied adaptively, as for algorithm 4, on the test set. As already mentioned, in non-stationary environments it is necessary to update the models by introducing samples with the new data distribution to adapt them. To this end, we use a time window of 10 samples ($W$=10), which is equivalent to recalculating the model once ten samples are considered normal. This approach is feasible for hourly measurements since, in the best case, the model would need to be recomputed every ten hours. Smaller time windows (e.g. $W$=1) could lead to problems depending on the data availability, the model's complexity, and the required training time. However, when feasible, smaller time windows can increase the detection capabilities of the models. In the adaptive approach, we add the new samples considered non-outliers to the training set. This is a user-defined parameter since its value will always depend on the specific data domain of the application and data resolution. In addition to recalculating the model, we recompute the threshold $TH$ using the latest samples collected. We apply the same adaptive procedure for all models, global and residual-based. Similarly to the previous experiment, perturbations of different magnitude $\delta$ are applied to 30% of the test set, and five repetitions per perturbation magnitude are performed.

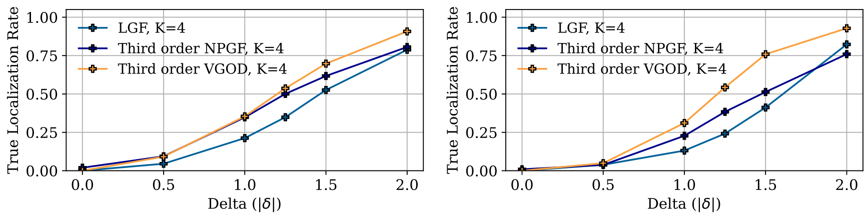
Table 8.5 shows the average TPRs and FPRs for the selected models and perturbations of different magnitude $\delta$. The same trend is observed as in the training results but with slightly lower TPRs in general. Firstly, in the case of the true positive rates, the VGOD process is the best method followed by the NPGF, in particular, the VGOD is able to improve the NPGF by about 2.5-11% TPR for outliers in the range between $|\delta|$=1.0-2.0 standard

**Table 8.5:** Average outlier detection results over the test set with $\delta$ standard deviation perturbations for data set *D.1*.

| **Model** | **$\|\delta\|$=0.0** | | **$\|\delta\|$=0.5** | | **$\|\delta\|$=1.0** | | **$\|\delta\|$=1.25** | | **$\|\delta\|$=1.5** | | **$\|\delta\|$=2.0** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TPR | FPR | TPR | FPR | TPR | FPR | TPR | FPR | TPR | FPR | TPR | FPR |
| **LGF** [22] | 0.0 | 0.19 | 0.20 | 0.18 | 0.31 | 0.15 | 0.41 | 0.14 | 0.56 | 0.13 | 0.80 | 0.12 |
| **Third order NPGF** [160] | 0.0 | 0.19 | 0.24 | 0.17 | 0.40 | 0.13 | 0.54 | 0.13 | 0.64 | 0.12 | 0.82 | 0.12 |
| **Third order VGOD** | 0.0 | 0.18 | 0.23 | 0.16 | 0.41 | 0.13 | 0.56 | 0.12 | 0.71 | 0.12 | 0.91 | 0.13 |
| **LOF** [169] | 0.0 | 0.15 | 0.20 | 0.15 | 0.32 | 0.15 | 0.45 | 0.14 | 0.61 | 0.15 | 0.84 | 0.16 |
| **KNN** [144] | 0.0 | 0.15 | 0.18 | 0.16 | 0.26 | 0.16 | 0.36 | 0.16 | 0.49 | 0.16 | 0.83 | 0.16 |
| **Frequency-based GSP** [161] | 0.0 | 0.15 | 0.19 | 0.15 | 0.34 | 0.15 | 0.48 | 0.15 | 0.61 | 0.15 | 0.78 | 0.16 |

deviations. In general, VGOD and NPGF are better able to detect outliers with smaller perturbations, e.g. in the range of $|\delta|$=1.0, thus showing better sensitivity in these ranges than LGF, LOF, KNN, and frequency-based GSP. For large perturbation values ($|\delta|$=2.0), all methods show a similar high ability, in the order of 78-84%, to detect outliers, except VGOD which goes up to 91%, outperforming all other methods. Then, the NPGF is able to improve the GSP model and the LGF by about 10% TPR for perturbations in the range 1.0-1.5 standard deviations, but this difference in performance is smaller with respect to the LOF, where it only improves by 5% in TPR. Finally, all global models as well as the LGF behave in a similar way, with the LOF being the model that obtains a better TPR for large disturbances.

On the other hand, Table 8.5 also shows the FPR committed by the different models. In fact, given the non-stationarity of the test set, we can see FPRs around 15-19% for $|\delta|$=0, although in the case of the residual-based methods (LGF, NPGF, and VGOD) these FPRs are reduced to 12-14% as the magnitude of the outliers increases. That is, as outliers are larger in magnitude the residual-based models have better sensitivity and also produce fewer false positives. Depending on the characteristics of the data set and the computational capabilities, the adaptive window $W$ can be reduced to improve the models' results.



(a) Localization rate test set results for the three residual-based models for data set *D.1*.

(b) Localization rate test set results for the three residual-based models for data set *D.2*.

**Figure 8.6:** Average localization rate test set results for the three residual-based models and the small-sized sensor network *D.1* and mid-sized network *D.2*.

Now, let's see how the residual-based models work with respect to the localization of the sensor that has the outlier. This step is very important in sensor networks in order to carry out actions to mitigate the effects of the outlier, actions such as the imputation of the sensor measurement using a virtual sensor, the removal of the measure, and even the replacement of the malfunctioning sensor. Figure 8.6.a) shows the true localization rate results for the test set, this rate is defined as the precise detection of the outlying sensor, where the localization rates are slightly smaller than the detection rate, meaning that sometimes the models fail in locating the specific outlying sensor. However, the results show how VGOD outperforms the NPGF and the LGF. For perturbations of 1.0 standard deviation ($1\sigma$) the two nonlinear models behave similarly with a location rate of about a 37%. Nevertheless, as the perturbation magnitude increases the performance gap between the three models also increases, leading to a localization rate of 70% with 1.5 standard deviation perturbations with the VGOD process, more than 10% higher location rate than the others.

Figure 8.6.b), shows the outlier localization results for the data set comprising the Catalonia reference stations (*D.2*). In this case, we have focused on a subset of the network nodes given that some violate the assumption made for the model to work, i.e., nodes that are not related to any of the network nodes due to their geographical location. The same trend as in the previous case is verified, where the nonlinear models have a better localization for outliers of magnitude in the range $|\delta|$=1.0-1.5 standard deviation. However, the gap between the localization performance of the VGOD and the NPGF is larger in this case for outliers of magnitude in the range $|\delta|$=1.0-2.0 standard deviations since the network is very heterogeneous, and a graph learned from the data captures better the relationships between nodes. Besides, distance-based graphs worsen the performance in this case given that when there are nodes that although they are close they are not correlated, then the relationships between these sensors are not well-defined by distance-based graphs.

The sensor network represented by the *D.2* data set includes sensors whose relationships are not well defined by distances, which is a common scenario in air pollution sensor networks whose nodes are deployed in specific locations with high concentrations of air pollution, and whose signals are highly dependent on ambient conditions and other pollutants. Therefore, outlier detection models using distance-based graphs do not work well in that case given that the GSR stage is distorted by erroneous sensor-to-sensor relationships. This problem does not happen with a graph learned from the data since uncorrelated sensors are weakly connected to other sensors or disconnected, proving to be a more robust alternative for these air pollution monitoring networks.

### 8.3.5. Sensor error detection: Drift Detection

We evaluate how the proposed model can be used to detect a common type of error in LCS networks, the sensor drift. In this way, we observed the pattern generated by the outlier detection models and how this can enable replacement or recalibration actions to maintain the quality of the network. Since an outlier detection model detects samples that have unusual behavior, this technique can be further used to detect specific sensor errors by the inspection of the outlier detection results. Here, we use the data set of a real heterogeneous network deployment, *D.3*, composed of three reference stations (high-precision nodes) and five LCS nodes. To this end, we add an offset of increasing magnitude in one of the LCSs, as $\epsilon_t \sim \mathrm{N}(2.0/t, 0.1)$ and $t \in (0, 2.0]$. Figure 8.7 shows the result of applying the VGOD mechanism, where sensor 3 is the drifted sensor, and the model is able to detect the simulated drift after its magnitude nearly becomes 0.5 standard deviations. Although some false positives can also be observed (13%), the pattern observed for sensor 3 is very different from the others, thus identifying a malfunctioning sensor. In addition, a filter could be used to reduce the false positive rate and help detecting this specific type of error, an example is the use of the exponentially weighted moving average (EWMA) to eliminate false positives and filter this type of pattern as a malfunctioning sensor but it would introduce a delay in the detection [152]. The obtained TPR for the VGOD is 78% and a FPR of 13%, the linear graph filter obtained a TPR of 58% and a FPR of 10%, and finally, the NPGF obtained a TPR of 70% and a FPR of 13%. Again our proposed model outperforms the other two graph-based models. This example shows the importance of these graph-based techniques that enable the localization of the faulty sensor. In fact, given the localization capability of this model, this type of sensor malfunction can be detected, and the sensor can be replaced or can undergo a recalibration process.

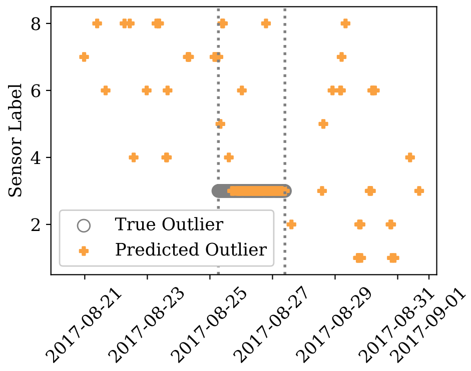


**Figure 8.7:** Outlier detection results for a drifted sensor (sensor 3) over the H2020 Captor data set (data set *D.3*) using the VGOD algorithm.

## 8.3.6. SCALABILITY

In previous sections, we have observed how the graph and residual-based nonlinear models, VGOD and NPGF, have obtained a good performance in the detection of outliers. In addition, both methods allow the localization of which sensor produces outliers, an aspect of great interest in this type of sensor network. Given the adaptive nature of the algorithms and the need to retrain the GSR models, the scalability and complexity of the reconstruction models are important for their application in this real scenario. Therefore, in this section we compare the scalability of the core element of the VGOD, the Volterra-based GSR model [174], with the third order NPGF [160].

There are two main differences between the VGOD and the third order NPGF for outlier detection: *i)* the authors in [160] use a shift operator built from a distance-based function between nodes, and we propose to use a shift operator that is built using the data measured by the nodes, and *ii)* the authors in [160] use the NPGF, whose structure is similar to the Volterra-based model but the higher order interactions differ. Indeed, the number of NPGF parameters for a degree of interaction $D$ and depth $K$ is $N_{model} = (N + K + K^2 + \sum_{i=1}^{D-2} K^2 N^i)$, while the number of the parameters for the Volterra-based model is $(N + \sum_{i=1}^{D} K^i)$. Table 8.6 compares the number of parameters for third order models ($D$=3) with models' depth four ($K$=4), for different network sizes $N$.

**Table 8.6:** Number of parameters for third order NPGF and third order Volterra model, with $K = 4$.

| **Model** | **N=10** | **N=50** | **N=100** |
|---|---|---|---|
| Third order NPGF | 190 | 870 | 1720 |
| Third order Volterra model | 94 | 134 | 184 |

In order to show how the number of model parameters of both, the GSR Volterra model and NPGF model, affect the solving time of the convex problem in equation (8.6) we perform the following experiment: given a certain depth $K$=4, we simulate data sets with an increasing number of nodes $N$ and an increasing number of samples $P$. To do this, we simulate the sensors coordinates in the unit square as $c_x, c_y \sim U(0,1)$, and define a weight matrix by the distance-based radial basis function $W_{ij} = e^{\frac{-d_{ij}^2}{2*0.5}}$, where $d_{ij}$ is the distance between sensor $x_i$ and sensor $x_j$. The samples are generated as a zero-mean multivariate Gaussian with precision matrix equal to the Laplacian pseudoinverse with noise injected on the diagonal, $\mathbf{x} \sim N(\mathbf{0}, \mathbf{L}^\dagger + \sigma \mathbf{I}_N)$. Then, for each pair ($N$,$P$) we perform five repetitions to calculate the average solving time. Figure 8.8 shows the optimization solving times for both models. In fact, the Volterra-based model is invariant to the number of nodes in the network, resulting in much lower solving times as the number of network nodes increases. The opposite happens with the third order NPGF, where the solving time increases dramatically as the number of nodes increases. For instance, for 1000 samples and 46 nodes the third order NPGF takes almost eight minutes, while the third order Volterra model takes just over one minute.

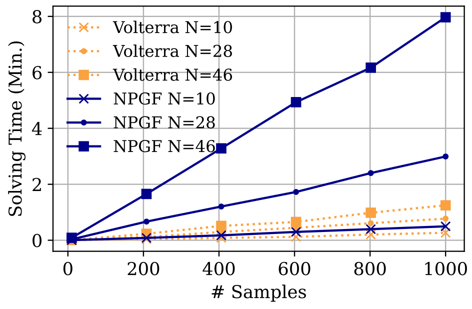


**Figure 8.8:** Optimization problem solving times, using the Splitting Conic Solver (SCS) for the third order Volterra-based and the third order NPGF models.

Moreover, in this particular case, where the objective function of the GSR model is set to the residual sum of squares (RSS) the model can be trained in the least squares sense using matrix pseudoinverse. Hence, we can avoid the use of a generic convex solver that although they are easy to use they may be less efficient than a matrix pseudoinverse. We can formulate the problem in matrix form, where $\mathbf{y} = \text{vec}(\mathbf{X})$, $\mathbf{y} \in \mathbb{R}^{(N \cdot N_{train})}$, we recall $\mathbf{X} \in \mathbb{R}^{N \times N_{train}}$ and $\tilde{\mathbf{X}}_{exp} \in \mathbb{R}^{(N \cdot N_{train}) \times N_{model}}$ where $N_{model}$ is the number of parameters (here we use $D = 3$) and $\tilde{\mathbf{X}}_{exp}$ is defined as:

$$\tilde{\mathbf{X}}_{exp} = \begin{bmatrix} \mathbf{I}_N & \mathbf{I}_N & \cdots & \mathbf{I}_N \\ \boldsymbol{\psi}_{1,1}(\mathbf{x}_1)^\top & \boldsymbol{\psi}_{1,1}(\mathbf{x}_2)^\top & \cdots & \boldsymbol{\psi}_{1,1}(\mathbf{x}_{N_{train}})^\top \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{\psi}_{1,K}(\mathbf{x}_1)^\top & \boldsymbol{\psi}_{1,K}(\mathbf{x}_2)^\top & \cdots & \boldsymbol{\psi}_{1,K}(\mathbf{x}_{N_{train}})^\top \\ \boldsymbol{\psi}_{2,1}(\mathbf{x}_1)^\top & \boldsymbol{\psi}_{2,1}(\mathbf{x}_2)^\top & \cdots & \boldsymbol{\psi}_{2,1}(\mathbf{x}_{N_{train}})^\top \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{\psi}_{2,K}(\mathbf{x}_1)^\top & \boldsymbol{\psi}_{2,K}(\mathbf{x}_2)^\top & \cdots & \boldsymbol{\psi}_{2,K}(\mathbf{x}_{N_{train}})^\top \\ \boldsymbol{\psi}_{3,1}(\mathbf{x}_1)^\top & \boldsymbol{\psi}_{3,1}(\mathbf{x}_2)^\top & \cdots & \boldsymbol{\psi}_{3,1}(\mathbf{x}_{N_{train}})^\top \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{\psi}_{3,1}(\mathbf{x}_1)^\top & \boldsymbol{\psi}_{3,1}(\mathbf{x}_2)^\top & \cdots & \boldsymbol{\psi}_{3,1}(\mathbf{x}_{N_{train}})^\top \end{bmatrix}^\top \tag{8.14}$$

Where $\boldsymbol{\psi}_{\cdot,\cdot}(\mathbf{x}_i) \in \mathbb{R}^N$ are column vectors representing the basis functions defined in section 8.2.2 and $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ is the identity matrix. Now, we can define the RSS problem in vector-matrix form as:

$$\begin{aligned} \mathbf{y} &= \tilde{\mathbf{X}}_{exp}\mathbf{h} \\ \hat{\mathbf{h}} &= \tilde{\mathbf{X}}^{\dagger}_{exp}\mathbf{y} \end{aligned} \tag{8.15}$$

Where $\mathbf{h} \in \mathbb{R}^{N_{model}}$ is the vector of concatenated parameters $\mathbf{h} = [\mathbf{h}_0^\top, \mathbf{h}_1^\top, \ldots, \mathbf{h}_D^\top,]^\top$ and $\mathbf{h}_i \in \mathbb{R}^{K^i}$. Recall, that the pseudoinverse can be faster than a generic convex solver but it can also struggle with large matrices since the pseudoinverse, which may be computed via singular value decomposition (SVD), can be challenging in a high-dimensional setting.

In short, it has been observed how the difference in the number of parameters of the Volterra model and the NPGF really affects the computation time. And this computation time can be an important parameter given the need to adaptively retrain the model to deal with the non-stationarity of the data.

## 8.4. CONCLUDING REMARKS & FUTURE WORK

In this chapter, we have proposed an outlier detection algorithm based on graph signal processing, called *Volterra graph-based outlier detection* algorithm. The results have shown the ability of residual-based and graph signal processing-based models to detect and locate sensor outliers, which is important in this particular field, where locating sensors with abnormal measurements can lead to further replacement or recalibration actions. In this way, we have tackled the following research question throughout the chapter:

**(R.Q.2.4):** *Can we approach one of the most important data quality problems, such as the detection and location of outliers or erroneous sensors, using the graph information?*

The proposed outlier detection process consists of three stages; *i)* learning a graph based on the measured data, *ii)* a GSR model based on the Volterra series, and *iii)* the subsequent inspection of the residuals of the signal reconstruction task to identify and locate the outlying measurements. This process allows not only for detecting an outlier in a sensor network sample but also localizing the sensor that produces the outlier, which is of great importance in the air pollution sensor network realm so that measurement replacement or recalibration actions can be done.

In summary, the VGOD method uses a shift matrix that is constructed using the measured data, unlike other graph-based methods that use shift matrices based on functions that decay exponentially with the distance between nodes. This aspect is key to the method as the shift matrix actively participates in two modules of the outlier detector, *i)* the selection of the graph edges and therefore of the neighbors of a node, and *ii)* in the GSR model. This feature improves the detection and localization of outliers. The second differential aspect is the use of the Volterra series as a signal reconstruction method, which improves the computational performance by requiring fewer parameters than other nonlinear methods, such as the NPGF.

The VGOD process has been compared to three state-of-the-art global outlier detection methods that detect but do not allow localization, the frequency-based GSP, the k-nearest neighbors (KNN), and the local outlier factor (LOF), and to two models based on reconstruction residuals and graph signal processing that detect and allow localization, the linear graph filter (LGF) and the third order NPGF model with a distance-based graph. The results show that VGOD increases the detection rate by at least 10% over the other models and has better localization of the sensors producing the outliers than the other two graph-based models. In addition, it is shown that the VGOD reconstruction model requires less training time than its closest graph-based competitor, the NPGF. Therefore, the proposed mechanism improves both outlier detection and model scalability with respect to NPGF.

Finally, the VGOD graph-based detection model has been applied to sensor drift detection in a low-cost heterogeneous sensor network. The results show the ability of the proposed method to detect the outlying samples and locate the drifting sensor, thus allowing the identification of the drifting sensor for a possible replacement or sensor recalibration. The results show hoe this method can be used as monitoring tool to asses the quality of the measures and the health of the sensors.

The most relevant aspects and advantages of the VGOD are defined below:

**Sensor Outlier Detection and Localization**

- The VGOD performs well with respect to the state of the art in the detection and localization of outliers in air pollution sensor networks.
- The graph learned from the data offers better detection and robustness of the VGOD for highly heterogeneous sensor networks with complex relationships and possible sensors without relationships with other sensors.

**Low-Cost Sensor Drift Detection**

- VGOD has been successfully applied to the detection of a drifting sensor in a heterogeneous network of LCSs. The results have shown better detection and localization metrics than the state-of-the-art models. In this way, the VGOD can also help in the detection of erroneous sensors for further actions, such as recalibration and sensor replacement.

**Scalability**

- The signal reconstruction element of the VGOD, the Volterra-like model, has a smaller number of parameters than the NPGF and a lower complexity, which makes it more suitable for the online/adaptive training of the model, allowing more frequent retraining without the need for large computational resources. In addition, the reduced complexity of the model avoids possible overfitting problems.

The weaknesses of the method refer to the two main assumptions for the model to work: *i)* in the case of having a network with sensors deployed in sparse areas without significant relationships, the method may not be able to detect outliers in those sensors, and *ii)* the mechanism needs the graph to be learned from correct sensor values, so it is assumed that the sensors will function well during the training phase.

As future work, it would be interesting to study the applicability of graph neural networks for air pollution LCS network outlier detection, with specific training methodologies to deal with small training data sets and the need for periodic retraining.

**Practical Tip !**

In the case of wanting to detect anomalous measurements or malfunctioning sensors to carry out replacement or recalibration actions, one can run the VGOD adaptively in real time in order to carry out actions to maintain the quality of the network data. Outlier detection can be seen as monitoring tool to assess the data quality and health of the sensors. This can be used in conjunction with the data reconstruction framework to mitigate data quality issues and provide data robustness and completeness.

# 9
# CONCLUSIONS

THROUGHOUT the different chapters, the conclusions and future work specific to the problems dealt with have been given. Here, we describe the more general conclusions to list the contributions made during the thesis. Thus, we see how the studies carried out contribute to the field of low-cost sensors for air pollution monitoring.

The paradigm of air quality monitoring using low-cost sensors is a field in constant progression and has seen great advances over the last decade. The main focus of attention has been the use of mature enabling technologies, such as machine learning and the internet of things, to improve the quality of data that these sensors provide. In particular, the main focus has been on improving the quality of the data in order to meet minimum accuracy requirements for using low-cost sensors in conjunction with regulated governmental instrumentation to measure regulated pollutants.

In this line, many researchers have started using machine learning techniques to calibrate low-cost sensors. It has also been established that the future of air quality monitoring networks lies in the creation of heterogeneous networks where high-precision instruments coexist with a large number of low-cost sensors to improve the spatial resolution of measured pollutants.

Throughout the course of this thesis, we have studied the improvement of the quality of the data obtained in a network of low-cost sensors for air quality monitoring. From the study of sensor sampling and the improvement of sensor calibration using machine learning techniques, to the use of the data obtained by a sensor network to perform post-processing tasks that allow for monitoring and maintaining the quality of the data measured by the network. In this way, we have addressed both the quality improvement of the individual sensors prior to deployment and the maintenance of the quality of the sensor network estimations over the lifetime of the network deployment. Figure 9.1 shows the topics of the different chapters as well as the findings produced.

**PART I: Machine Learning-Based *In-Situ* Sensor Calibration**

This first part of the thesis has been focused on individual sensor data quality improvement using machine learning-based calibration. Firstly, the pre-processing and sampling of low-cost sensors have been studied. Secondly, given the raw values of the sensors, the application of machine learning techniques for the calibration of these sensors has been studied. And finally, refining this calibration, the multisensor data fusion calibration by means of machine learning, introducing more than one target sensor in the calibration model, has been proposed.

- ***Sensor sampling & pre-processing***: we have analyzed which are the pre-processing stages of the sensors before obtaining air pollution estimates. We have evaluated the effect that *sampling* has on the quality of calibration and the data quality, allowing us to establish duty cycle techniques to efficiently measure air pollution and reduce energy consumption. The results have shown how data quality and representativeness can be maintained with an energy-saving duty cycle.
- ***Machine learning-based in-situ calibration***: once the phenomenon has been sampled and the sensor measurements have been collected, we have evaluated the use of supervised *machine learning* techniques to calibrate *in-situ* the sensors against reference instrumentation. Both linear and nonlinear models have allowed the improvement of the data quality and estimates, with the nonlinear models being more accurate at the expense of needing more data to train. In addition, it has been observed the need to recalibrate the sensors, i.e., to retrain the calibration models periodically so that the estimates do not worsen due to model outdatedness.
- ***Multisensor in-situ sensor calibration***: apart from calibrating a sensor by means of a sensor array containing the sensor itself and other sensors to compensate for cross-sensitivities and cross-correlation, low-cost sensors contain imperfections that make it possible to introduce more than one sensor of the same family in the calibration. Thus, we have introduced the *multisensor data fusion calibration* where more than one sensor of the same pollutant is used to improve the estimation of the pollutant by using machine learning. The results have shown how introducing up to four sensors of the same pollutant or combining sensors of different technologies can improve the calibration of the sensors and the subsequent pollution estimation.

Ultimately, as a result of the various points mentioned above, the improvement of data quality and the estimation of pollution concentrations by means of machine learning-based techniques has been deepened. Now, once we have been able to produce more accurate pollution estimation values, we have left the individual study of the sensors and moved on to the study of the measurements of the sensor network as a whole. Thus, we have studied how to use network data to maintain the quality of the data over the lifetime of the deployment.

**PART II: Graph-Based Analysis of Air Pollution Sensor Networks**

In this second part of the thesis, we have focused on the monitoring of the pollution measurements produced by a sensor network. For this purpose, we have used a novel approach based on graph signal processing, describing the relationships between sensors distributed in a region by means of a graph, and interpreting each measurement obtained by the network at an instant of time as a signal defined on the graph. Given the growing interest in graph signal processing, this approach has enabled the application of different techniques from the field of signal processing and machine learning to perform tasks such as data-driven graph learning or network data reconstruction. This approach has allowed addressing different post-processing applications that appear in this type of sensor deployment, where the quality of the measured data can be affected by long-term problems, such as node malfunctioning, node maintenance or relocation. Therefore, this approach has made it possible to take advantage of the data measured by the network to maintain the quality of the data, thus overcoming possible data quality issues during the lifetime of the network deployment.

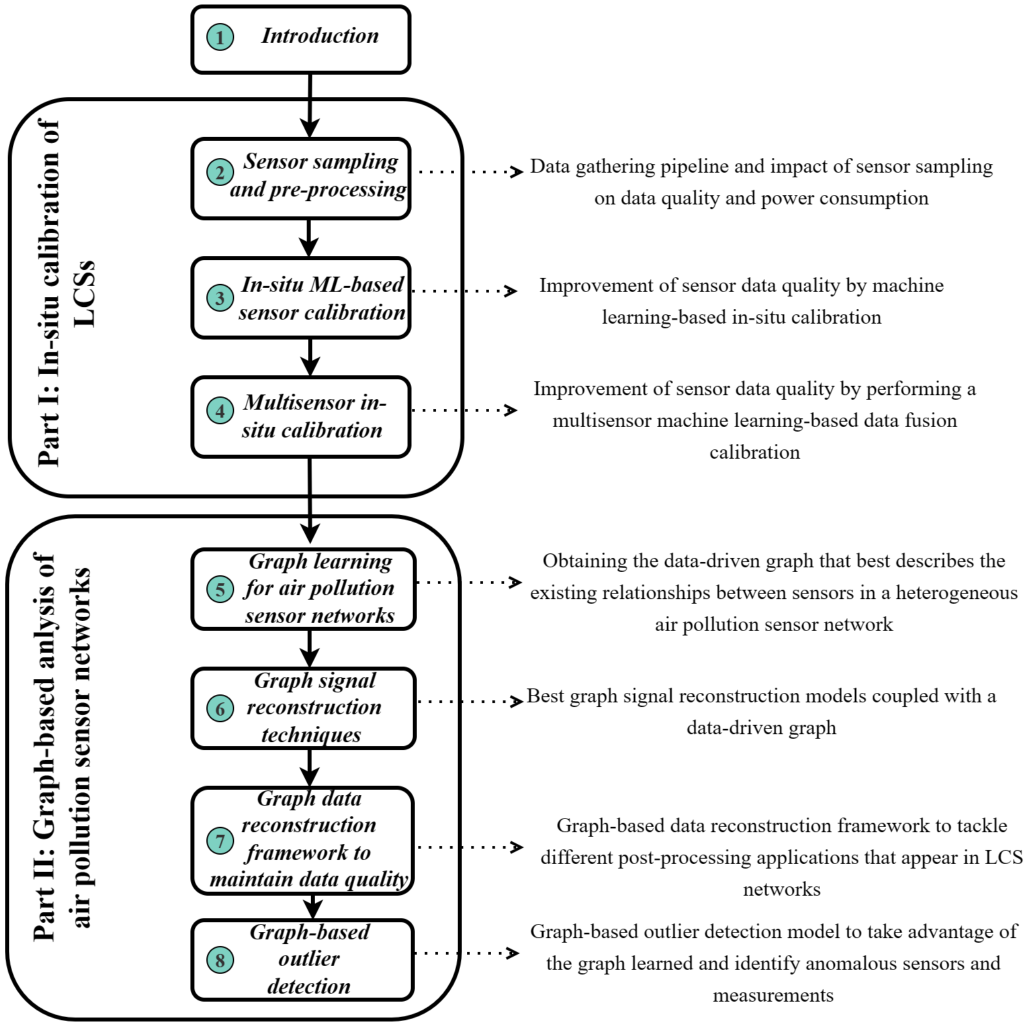
- ***Graph learning techniques for air pollution sensor networks***: in this first chapter, we have studied which are the best techniques to learn a graph that represents the measurements of a network of air pollution sensors. For this purpose, we have compared *graph inferring* techniques based on *graphical lasso*, *geographical distance*, and *graph signal smoothness* notion. To compare them, we have analyzed the efficiency of data reconstruction, where the graph acts as a feature selector selecting the neighborhoods of each sensor. The results show that the data-driven approach based on the smoothness of the signals is the most suitable technique since it allows obtaining a Laplacian matrix that describes the existing relationships between sensors. In addition, we have shown how clustering can be used to mitigate scalability problems.
- ***Graph signal reconstruction techniques for air pollution sensor networks***: once it had been seen that the best way is to learn the graph from the data, different *graph signal reconstruction* techniques from different families have been studied; semi-supervised learning, signal processing, and kernel methods. The experiments performed mimic the real case where any node in the network can present data gaps, even different sensors at the same time. The results have proven that kernel-based graph signal reconstruction and Laplacian interpolation to be superior, with Laplacian interpolation being the simplest (requiring fewer hyperparameters) and with good performance for sparse graphs.
- ***Graph-based data reconstruction framework to maintain air pollution sensor network data quality***: in this chapter, we have dealt with different post-processing applications that arise in this type of sensor network, where nodes may fail, may be under maintenance, may be relocated, and may present data gaps. Therefore, using the smoothness-based graph and signal reconstruction by Laplacian interpolation, we have presented a *graph-based data reconstruction framework* that allows for dealing with a large set of applications to maintain data quality. This framework presents great flexibility, since any node can be imputed in real time, and also because the nature of the graph signal processing tools allow a distributed implementation. The results have shown that in some applications such as missing value imputation, the proposed framework can obtain a slightly lower performance than specific imputation algorithms, but it is more versatile and achieves better performance in other tasks, such as virtual sensing and data fusion. In this way, this graph-based data reconstruction framework allows for maintaining the quality of the network data facing different problems that may arise in an accurate, interpretable, and efficient way.
- ***Graph-based outlier detection for air pollution sensor networks***: once the data

quality problems had been tackled by data reconstruction, there was still the detection of anomalous measurements. In the same context, we have proposed an outlier detection algorithm, the *Volterra graph-based outlier detection*, that uses the measurements from the sensors distributed in a region by means of a graph and a Volterra-like graph signal reconstruction model. The implemented algorithm has shown great detection and localization capabilities of the sensor/measurement causing the outlier. It has also proven its ability to detect malfunctioning sensors, e.g. drifting sensors. Hence, this algorithm is very useful for this type of network where detecting the sensor that produces the outlier measurement allows for carrying out the proper actions; data imputation, recalibration, replacement, or maintenance.

In short, in this second part of the thesis, we have seen how graphs are a good tool to represent sensor networks for air pollution monitoring. In addition, the topology described by the graph provides information about the relationships between sensors and allows using data from other sensors distributed in a region in a simple and efficient way. Hence, we have discussed and experimented how a graph-based approach can be useful to maintain and monitor the quality of the data provided by a sensor network, which may have low-cost sensors and high-precision nodes, in order to increase the reliability of the network measurements. Furthermore, graph signal processing is a fast-growing field so this approach will benefit from future methodologies and techniques.

In conclusion, in the first part, we have improved the low-cost sensor data quality individually, using machine learning techniques in calibration. Whilst, in the second part we have combined graph signal processing techniques with machine learning to maintain and monitor the quality of the estimates produced by a sensor network.

Generally speaking, there are two branches for future work. First, the study of more advanced calibration techniques that can adapt to the particularities of *in-situ* calibration of low-cost sensors such as the data available for calibration, for instance using transfer learning or pre-trained models. The study of adaptive techniques to deal with the problem of concept drift and relocation could have a large impact on the operational costs associated with the deployment of a low-cost sensor network. Secondly, it could be said that, to the best of our knowledge, this thesis has opened the possibility of applying graph signal processing to the specific case of heterogeneous air pollution sensor networks and the different tasks required. Thus, future work can elaborate more specific graph-based algorithms to reconstruct signals, graph learning models specific to the mobile sensor network case, or even the application of specific graph neural network methodologies over the whole framework described during the second part of the thesis. Besides, it would also be interesting to study other applications that can be approached using this graph-based framework, e.g., sensor clustering, sensor placement, or other air pollution-specific tasks. All in all, we believe that this graph-based framework can be seen as a sensor network monitoring tool, which will be further exploited and investigated to be included in data quality-ensuring pipelines for different kinds of applications.



1 Introduction
Part I: In-situ calibration of LCSs
2 Sensor sampling and pre-processing
Data gathering pipeline and impact of sensor sampling on data quality and power consumption
3 In-situ ML-based sensor calibration
Improvement of sensor data quality by machine learning-based in-situ calibration
4 Multisensor in-situ calibration
Improvement of sensor data quality by performing a multisensor machine learning-based data fusion calibration
Part II: Graph-based anlysis of air pollution sensor networks
5 Graph learning for air pollution sensor networks
Obtaining the data-driven graph that best describes the existing relationships between sensors in a heterogeneous air pollution sensor network
6 Graph signal reconstruction techniques
Best graph signal reconstruction models coupled with a data-driven graph
7 Graph data reconstruction framework to maintain data quality
Graph-based data reconstruction framework to tackle different post-processing applications that appear in LCS networks
8 Graph-based outlier detection
Graph-based outlier detection model to take advantage of the graph learned and identify anomalous sensors and measurements

**Figure 9.1:** General overview of the different thesis chapters and the main challenges addressed.

# BIBLIOGRAPHY

[1] M. Kampa and E. Castanas, “Human health effects of air pollution”, *Environmental pollution*, vol. 151, no. 2, pp. 362–367, 2008.

[2] R. M. Babadjouni *et al.*, “Clinical effects of air pollution on the central nervous system; a review”, *Journal of Clinical Neuroscience*, vol. 43, pp. 16–24, 2017.

[3] W. H. Organization *et al.*, *WHO global air quality guidelines: particulate matter (PM2. 5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide*. World Health Organization, 2021.

[4] D. E. Williams, “Low cost sensor networks: How do we know the data are reliable?”, *ACS sensors*, vol. 4, no. 10, pp. 2558–2565, 2019.

[5] P. Kumar *et al.*, “The rise of low-cost sensing for managing air pollution in cities”, *Environment international*, vol. 75, pp. 199–205, 2015.

[6] N. H. Motlagh *et al.*, “Toward massive scale air quality monitoring”, *IEEE Communications Magazine*, vol. 58, no. 2, pp. 54–59, 2020.

[7] A. C. Rai *et al.*, “End-user perspective of low-cost sensors for outdoor air pollution monitoring”, *Science of The Total Environment*, vol. 607, pp. 691–705, 2017.

[8] A. Ripoll *et al.*, “Testing the performance of sensors for ozone pollution monitoring in a citizen science approach”, *Science of the Total Environment*, vol. 651, pp. 1166–1179, 2019.

[9] J. M. Barcelo-Ordinas, P. Ferrer-Cid, J. Garcia-Vidal, M. Viana, and A. Ripoll, “H2020 project captor dataset: Raw data collected by low-cost mox ozone sensors in a real air pollution monitoring network”, *Data in Brief*, vol. 36, p. 107 127, 2021.

[10] J. M. Barcelo-Ordinas, M. Doudou, J. Garcia-Vidal, and N. Badache, “Self-calibration methods for uncontrolled environments in sensor networks: A reference survey”, *Ad Hoc Networks*, vol. 88, pp. 142–159, 2019.

[11] B. Maag, Z. Zhou, and L. Thiele, “A survey on sensor calibration in air pollution monitoring deployments”, *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4857–4870, 2018.

[12] F. Concas *et al.*, “Low-cost outdoor air quality monitoring and sensor calibration: A survey and critical analysis”, *ACM Transactions on Sensor Networks (TOSN)*, vol. 17, no. 2, pp. 1–44, 2021.

[13] L. Spinelle, M. Aleixandre, and M. Gerboles, “Protocol of evaluation and calibration of low-cost gas sensors for the monitoring of air pollution”, *Publication Office of the European Union, Luxembourg*, 2013.

[14] L. Spinelle, M. Gerboles, M. G. Villani, M. Aleixandre, and F. Bonavitacola, “Field calibration of a cluster of low-cost commercially available sensors for air quality monitoring. part b: NO, CO and CO2”, *Sensors and Actuators B: Chemical*, vol. 238, pp. 706–715, 2017.

[15] N. Zimmerman *et al.*, “A machine learning calibration model using random forests to improve sensor performance for lower-cost air quality monitoring.”, *Atmospheric Measurement Techniques*, vol. 11, no. 1, 2018.

[16] A. Bigi, M. Mueller, S. K. Grange, G. Ghermandi, and C. Hueglin, “Performance of no, $no_2$ low cost sensors and three calibration approaches within a real world application”, *Atmospheric Measurement Techniques*, vol. 11, no. 6, pp. 3717–3735, 2018.

[17] J. M. Barcelo-Ordinas, J. Garcia-Vidal, M. Doudou, S. Rodrigo-Muñoz, and A. Cerezo-Llavero, “Calibrating low-cost air quality sensors using multiple arrays of sensors”, in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2018, pp. 1–6.

[18] J. M. Barcelo-Ordinas, P. Ferrer-Cid, J. Garcia-Vidal, A. Ripoll, and M. Viana, “Distributed multi-scale calibration of low-cost ozone sensors in wireless sensor networks”, *Sensors*, vol. 19, no. 11, 2019, ISSN: 1424-8220. DOI: 10.3390/s19112503.

[19] P. Schneider, N. Castell, M. Vogt, F. R. Dauge, W. A. Lahoz, and A. Bartonova, “Mapping urban air quality in near real-time using observations from low-cost sensors and model information”, *Environment international*, vol. 106, pp. 234–247, 2017.

[20] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains”, *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[21] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, “Graph signal processing: Overview, challenges, and applications”, *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[22] A. Sandryhaila and J. M. Moura, “Discrete signal processing on graphs: Frequency analysis”, *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[23] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, “Connecting the dots: Identifying network structure via graph signal processing”, *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[24] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, “Learning graphs from data: A signal representation perspective”, *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.

[25] V. Chandola, A. Banerjee, and V. Kumar, “Outlier detection: A survey”, *ACM Computing Surveys*, vol. 14, p. 15, 2007.

[26] P. Ferrer-Cid, J. Garcia-Calvete, A. Main-Nadal, Z. Ye, J. M. Barcelo-Ordinas, and J. Garcia-Vidal, “Sampling trade-offs in duty-cycled systems for air quality low-cost sensors”, *Sensors*, vol. 22, no. 10, p. 3964, 2022.

[27] L. Spinelle, M. Gerboles, M. G. Villani, M. Aleixandre, and F. Bonavitacola, “Field calibration of a cluster of low-cost available sensors for air quality monitoring. part a: Ozone and nitrogen dioxide”, *Sensors and Actuators B: Chemical*, vol. 215, pp. 249–257, 2015.

[28] M. Si, Y. Xiong, S. Du, and K. Du, "Evaluation and calibration of a low-cost particle sensor in ambient conditions using machine-learning methods", *Atmospheric Measurement Techniques*, vol. 13, no. 4, pp. 1693–1707, 2020.

[29] P. Han *et al.*, "Calibrations of low-cost air pollution monitoring sensors for co, no2, o3, and so2", *Sensors*, vol. 21, no. 1, p. 256, 2021.

[30] B. Mijling, Q. Jiang, D. d. Jonge, and S. Bocconi, "Field calibration of electrochemical no2 sensors in a citizen science context", *Atmospheric Measurement Techniques*, vol. 11, no. 3, pp. 1297–1312, 2018.

[31] G. D. Astudillo, L. E. Garza-Castañon, and L. I. M. Avila, "Design and evaluation of a reliable low-cost atmospheric pollution station in urban environment", *IEEE Access*, vol. 8, pp. 51 129–51 144, 2020.

[32] S. Ali, T. Glass, B. Parr, J. Potgieter, and F. Alam, "Low cost sensor with iot lorawan connectivity and machine learning-based calibration for air pollution monitoring", *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2020.

[33] T. Becnel *et al.*, "A distributed low-cost pollution monitoring platform", *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 738–10 748, 2019.

[34] P. Chiavassa, F. Gandino, and E. Giusto, "An investigation on duty-cycle for particulate matter monitoring with light-scattering sensors", in *2021 6th International Conference on Smart and Sustainable Technologies (SpliTech)*, IEEE, 2021, pp. 1–6.

[35] M. A. Fekih, W. Bechkit, and H. Rivano, "On the data analysis of participatory air pollution monitoring using low-cost sensors", in *2021 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2021, pp. 1–7.

[36] M. R. Chowdhury, S. De, N. K. Shukla, and R. N. Biswas, "Energy-efficient air pollution monitoring with optimum duty-cycling on a sensor hub", in *2018 Twenty Fourth National Conference on Communications (NCC)*, IEEE, 2018, pp. 1–6.

[37] G. R. Espinosa, B. Montrucchio, E. Giusto, and M. Rebaudengo, "Low-cost pm sensor behaviour based on duty-cycle analysis", in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2021, pp. 1–8.

[38] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: A survey", *IEEE wireless communications*, vol. 14, no. 2, pp. 70–87, 2007.

[39] A. Rebeiro-Hargrave *et al.*, "City wide participatory sensing of air quality", *Frontiers in Environmental Science*, p. 587,

[40] R. Sahu *et al.*, "Robust statistical calibration and characterization of portable low-cost air quality monitoring sensors to quantify real-time o3 and no2 concentrations in diverse environments", *Atmospheric Measurement Techniques*, vol. 14, no. 1, pp. 37–52, 2021.

[41] P. Nowack, L. Konstantinovskiy, H. Gardiner, and J. Cant, “Machine learning calibration of low-cost no2 and pm10 sensors: Non-linear algorithms and their impact on site transferability”, *Atmospheric Measurement Techniques*, vol. 14, no. 8, pp. 5637–5655, 2021.

[42] S. De Vito *et al.*, “Calibrating chemical multisensory devices for real world applications: An in-depth comparison of quantitative machine learning approaches”, *Sensors and Actuators B: Chemical*, vol. 255, pp. 1191–1210, 2018.

[43] M. I. Mead *et al.*, “The use of electrochemical sensors for monitoring urban air quality in low-cost, high-density networks”, *Atmospheric Environment*, vol. 70, pp. 186–203, 2013.

[44] S. Marathe, A. Nambi, M. Swaminathan, and R. Sutaria, “Currentsense: A novel approach for fault and drift detection in environmental iot sensors”, in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 93–105.

[45] L. Sun, D. Westerdahl, and Z. Ning, “Development and evaluation of a novel and cost-effective approach for low-cost no2 sensor drift correction”, *Sensors*, vol. 17, no. 8, p. 1916, 2017.

[46] *Support circuits (ppb): Isb individual sensor board datasheet*, [Online] `https://www.alphasense.com/products/support-circuits-air/`, [Accessed: 20 November 2021].

[47] *Reference station data website of the regional government of catalonia, spain*, [Online] `https://mediambient.gencat.cat/es/05_ambits_dactuacio/atmosfera/qualitat_de_laire/vols-saber-que-respires/descarrega-de-dades/descarrega-dades-automatiques/index.html`.

[48] *Zenodo’s captor data website*, [Online] `https://zenodo.org/record/5770589`.

[49] E. G. Snyder *et al.*, “The changing paradigm of air pollution monitoring”, *Environmental science & technology*, vol. 47, no. 20, pp. 11 369–11 377, 2013.

[50] P. Ferrer-Cid, J. M. Barcelo-Ordinas, J. Garcia-Vidal, A. Ripoll, and M. Viana, “A comparative study of calibration methods for low-cost ozone sensors in iot platforms”, *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9563–9571, Dec. 2019, ISSN: 2372-2541.

[51] Z. Idrees and L. Zheng, “Low cost air pollution monitoring systems: A review of protocols and enabling technologies”, *Journal of Industrial Information Integration*, vol. 17, p. 100 123, 2020.

[52] S. Rajasegarar *et al.*, “High-resolution monitoring of atmospheric pollutants using a system of low-cost sensors”, *IEEE transactions on geoscience and remote sensing*, vol. 52, no. 7, pp. 3823–3832, 2013.

[53] N. Castell *et al.*, “Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates?”, *Environment international*, vol. 99, pp. 293–302, 2017.

[54] M. Penza, D. Suriano, M. G. Villani, L. Spinelle, and M. Gerboles, “Towards air quality indices in smart cities by calibrated low-cost sensors applied to networks”, in *SENSORS, 2014 IEEE*, IEEE, 2014, pp. 2012–2017.

[55] L. Spinelle, M. Gerboles, M. G. Villani, M. Aleixandre, and F. Bonavitacola, “Calibration of a cluster of low-cost sensors for the measurement of air pollution in ambient air”, in *SENSORS, 2014 IEEE*, IEEE, 2014, pp. 21–24.

[56] J. M. Cordero, R. Borge, and A. Narros, “Using statistical methods to carry out in field calibrations of low cost air quality sensors”, *Sensors and Actuators B: Chemical*, vol. 267, pp. 245–254, 2018.

[57] D. Hasenfratz, O. Saukh, and L. Thiele, “On-the-fly calibration of low-cost gas sensors”, in *European Conference on Wireless Sensor Networks*, Springer, 2012, pp. 228–244.

[58] O. Saukh, D. Hasenfratz, and L. Thiele, “Reducing multi-hop calibration errors in large-scale mobile sensor networks”, in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, ser. IPSN ’15, New York, NY, USA: ACM, 2015, pp. 274–285.

[59] P. Wei *et al.*, “Impact analysis of temperature and humidity conditions on electrochemical sensor response in ambient air quality monitoring”, *Sensors*, vol. 18, no. 2, p. 59, 2018.

[60] M. Mueller, J. Meyer, and C. Hueglin, “Design of an ozone and nitrogen dioxide sensor unit and its long-term operation within a sensor network in the city of zurich”, *Atmospheric Measurement Techniques*, vol. 10, no. 10, pp. 3783–3799, 2017.

[61] J. Friedman, T. Hastie, R. Tibshirani, *et al.*, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[62] T. Sayahi, A. Butterfield, and K. Kelly, “Long-term field evaluation of the plantower pms low-cost particulate matter sensors”, *Environmental pollution*, vol. 245, pp. 932–940, 2019.

[63] P. Ferrer-Cid, J. M. Barcelo-Ordinas, J. Garcia-Vidal, A. Ripoll, and M. Viana, “Multisensor data fusion calibration in iot air pollution platforms”, *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3124–3132, 2020.

[64] S. Madgwick, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays”, *Report x-io and University of Bristol (UK)*, vol. 25, pp. 113–118, 2010.

[65] J. Wu, Y. Feng, and P. Sun, “Sensor fusion for recognition of activities of daily living”, *Sensors*, vol. 18, no. 11, p. 4029, 2018.

[66] R. Tan, G. Xing, Z. Yuan, X. Liu, and J. Yao, “System-level calibration for data fusion in wireless sensor networks”, *ACM Trans. Sen. Netw.*, vol. 9, no. 3, 28:1–28:27, Jun. 2013.

[67] F. Castanedo, “A review of data fusion techniques”, *The Scientific World Journal*, vol. 2013, 2013.

[68] C. Malings *et al.*, "Development of a general calibration model and long-term performance evaluation of low-cost sensors for air pollutant gas monitoring", *Atmospheric Measurement Techniques*, vol. 12, no. 2, pp. 903–920, 2019.

[69] M. A. Zaidan *et al.*, "Intelligent calibration and virtual sensing for integrated low-cost air quality sensors", *IEEE Sensors Journal*, vol. 20, no. 22, pp. 13 638–13 652, 2020.

[70] J. K. Hackett and M. Shah, "Multi-sensor fusion: A perspective", in *Proceedings., IEEE International Conference on Robotics and Automation*, IEEE, 1990, pp. 1324–1330.

[71] R. C. Luo, C.-C. Yih, and K. L. Su, "Multisensor fusion and integration: Approaches, applications, and future research directions", *IEEE Sensors journal*, vol. 2, no. 2, pp. 107–119, 2002.

[72] W. Elmenreich and R. Leidenfrost, "Fusion of heterogeneous sensors data", in *2008 International Workshop on Intelligent Solutions in Embedded Systems*, IEEE, 2008, pp. 1–10.

[73] C.-Y. Chong, S. Mori, W. H. Barker, and K.-C. Chang, "Architectures and algorithms for track association and fusion", *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 1, pp. 5–13, 2000.

[74] P. Avery, "Combining measurements with correlated errors", *CLEO Note CBX*, pp. 95–55, 1996.

[75] A. S. Gwelo *et al.*, "Principal components to overcome multicollinearity problem", *Oradea Journal of Business and Economics*, vol. 4, no. 1, pp. 79–91, 2019.

[76] L. Morawska *et al.*, "Applications of low-cost sensing technologies for air quality monitoring and exposure assessment: How far have they gone?", *Environment international*, vol. 116, pp. 286–299, 2018.

[77] H. Chojer, P. Branco, F. Martins, M. Alvim-Ferraz, and S. Sousa, "Development of low-cost indoor air quality monitoring devices: Recent advancements", *Science of the Total Environment*, vol. 727, p. 138 385, 2020.

[78] E. Esposito, S. De Vito, M. Salvato, G. Fattoruso, and G. Di Francia, "Computational intelligence for smart air quality monitors calibration", in *International Conference on Computational Science and Its Applications*, Springer, 2017, pp. 443–454.

[79] Y. Liu, K. Zhou, and Y. Lei, "Using bayesian inference framework towards identifying gas species and concentration from high temperature resistive sensor array data", *Journal of Sensors*, vol. 2015, 2015.

[80] S. Salcedo-Sanz, J. Portilla-Figueras, E. Ortiz-Garcia, A. Perez-Bellido, R. Garcia-Herrera, and J. Elorrieta, "Spatial regression analysis of nox and o 3 concentrations in madrid urban area using radial basis function networks", *Chemometrics and Intelligent Laboratory Systems*, vol. 99, no. 1, pp. 79–90, 2009.

[81] P. Ferrer-Cid, J. M. Barcelo-Ordinas, and J. Garcia-Vidal, "Graph learning techniques using structured data for iot air pollution monitoring platforms", *IEEE Internet of Things Journal*, pp. 1–1, 2021. DOI: 10.1109/JIOT.2021.3067717.

[82] P. Ferrer-Cid, J. M. Barcelo-Ordinas, and J. Garcia-Vidal, “Graph signal reconstruction techniques for iot air pollution monitoring platforms”, *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 25 350–25 362, 2022. DOI: 10.1109/JIOT.2022.3196154.

[83] A. Lewis, W. R. Peltier, and E. von Schneidemesser, “Low-cost sensors for the measurement of atmospheric composition: Overview of topic and future applications”, 2018.

[84] X. Niu, X. Wang, J. Gao, and X. Wang, “Has third-party monitoring improved environmental data quality? an analysis of air pollution data in china”, *Journal of environmental management*, vol. 253, p. 109 698, 2020.

[85] F. Karagulian *et al.*, “Review of the performance of low-cost sensors for air quality monitoring”, *Atmosphere*, vol. 10, no. 9, p. 506, 2019.

[86] S. Mahajan and P. Kumar, “Evaluation of low-cost sensors for quantitative personal exposure monitoring”, *Sustainable Cities and Society*, vol. 57, p. 102 076, 2020.

[87] N. U. Okafor, Y. Alghorani, and D. T. Delaney, “Improving data quality of low-cost iot sensors in environmental monitoring networks using data fusion and machine learning approach”, *ICT Express*, vol. 6, no. 3, pp. 220–228, 2020.

[88] G. R. McKercher and J. K. Vanos, “Low-cost mobile air pollution monitoring in urban environments: A pilot study in lubbock, texas”, *Environmental technology*, vol. 39, no. 12, pp. 1505–1514, 2018.

[89] V. van Zoest, F. B. Osei, G. Hoek, and A. Stein, “Spatio-temporal regression kriging for modelling urban no2 concentrations”, *International journal of geographical information science*, vol. 34, no. 5, pp. 851–865, 2020.

[90] F. E. Ahangar, F. R. Freedman, and A. Venkatram, “Using low-cost air quality sensor networks to improve the spatial and temporal resolution of concentration maps”, *International Journal of Environmental Research and Public Health*, vol. 16, no. 7, p. 1252, 2019.

[91] B. Crawford *et al.*, “Mapping pollution exposure and chemistry during an extreme air quality event (the 2018 kı–lauea eruption) using a low-cost sensor network”, *Proceedings of the National Academy of Sciences*, vol. 118, no. 27, 2021.

[92] H.-J. Chu, M. Z. Ali, and Y.-C. He, “Spatial calibration and pm2. 5 mapping of low-cost air quality sensors”, *Scientific reports*, vol. 10, no. 1, pp. 1–11, 2020.

[93] I. Sówka, M. Badura, M. Pawnuk, P. Szymański, and P. Batog, “The use of the gis tools in the analysis of air quality on the selected university campus in poland”, *Archives of Environmental Protection*, vol. 46, no. 1, 2020.

[94] A. Nori-Sarma *et al.*, “Low-cost no2 monitoring and predictions of urban exposure using universal kriging and land-use regression modelling in mysore, india”, *Atmospheric Environment*, vol. 226, p. 117 395, 2020.

[95] L. Weissert *et al.*, “Low-cost sensors and microscale land use regression: Data fusion to resolve air quality variations with high spatial and temporal resolution”, *Atmospheric environment*, vol. 213, pp. 285–295, 2019.

[96] E. S. Coker, A. K. Amegah, E. Mwebaze, J. Ssematimba, and E. Bainomugisha, "A land use regression model using machine learning and locally developed low cost particulate matter sensors in uganda", *Environmental Research*, vol. 199, p. 111 352, 2021.

[97] J. Hofman *et al.*, "Spatiotemporal air quality inference of low-cost sensor data: Evidence from multiple sensor testbeds", *Environmental Modelling & Software*, p. 105 306, 2022.

[98] G. S. Hagler, R. Williams, V. Papapostolou, and A. Polidori, *Air quality sensors and data adjustment algorithms: When is it no longer a measurement?*, 2018.

[99] R. E. Connolly *et al.*, "Long-term evaluation of a low-cost air sensor network for monitoring indoor and outdoor air quality at the community scale", *Science of The Total Environment*, vol. 807, p. 150 797, 2022.

[100] I. Jabłoński, "Graph signal processing in applications to sensor networks, smart grids, and smart cities", *IEEE Sensors Journal*, vol. 17, no. 23, pp. 7659–7666, 2017.

[101] I. Hough, R. Sarafian, A. Shtein, B. Zhou, J. Lepeule, and I. Kloog, "Gaussian markov random fields improve ensemble predictions of daily 1 km pm2. 5 and pm10 across france", *Atmospheric Environment*, vol. 264, p. 118 693, 2021.

[102] H.-R. Song, M. Fuentes, and S. Ghosh, "A comparative study of gaussian geostatistical models and gaussian markov random field models", *Journal of Multivariate analysis*, vol. 99, no. 8, pp. 1681–1697, 2008.

[103] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure", *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[104] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations", *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.

[105] V. Kalofolias, "How to learn a graph from smooth signals", in *Artificial Intelligence and Statistics*, 2016, pp. 920–929.

[106] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under laplacian and structural constraints", *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 825–841, 2017.

[107] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso", *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[108] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs", in *International Conference on Computational Learning Theory*, Springer, 2004, pp. 624–638.

[109] G. B. Ribeiro and J. B. Lima, "Graph signal processing in a nutshell", *Journal of Communication and Information Systems*, vol. 33, no. 1, 2018.

[110] X. Dong, D. Thanou, L. Toni, M. Bronstein, and P. Frossard, "Graph signal processing for machine learning: A review and new perspectives", *IEEE Signal processing magazine*, vol. 37, no. 6, pp. 117–127, 2020.

[111] N. Meinshausen, P. Bühlmann, *et al.*, "High-dimensional graphs and variable selection with the lasso", *The annals of statistics*, vol. 34, no. 3, pp. 1436–1462, 2006.

[112] A. P. Dempster, "Covariance selection", *Biometrics*, pp. 157–175, 1972.

[113] S. Chen *et al.*, "Signal inpainting on graphs via total variation minimization", in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 8267–8271.

[114] N. A. Pantazis and D. D. Vergados, "A survey on power control issues in wireless sensor networks.", *IEEE Communications Surveys and Tutorials*, vol. 9, no. 1-4, pp. 86–107, 2007.

[115] O. Heinävaara, J. Leppä-Aho, J. Corander, and A. Honkela, "On the inconsistency of $l$ 1-penalised sparse precision matrix estimation", *BMC bioinformatics*, vol. 17, no. 16, p. 448, 2016.

[116] A. Lazaridis, "A note regarding the condition number: The case of spurious and latent multicollinearity", *Quality & Quantity*, vol. 41, no. 1, pp. 123–135, 2007.

[117] B. Van Stein, H. Wang, W. Kowalczyk, M. Emmerich, and T. Bäck, "Cluster-based kriging approximation algorithms for complexity reduction", *Applied Intelligence*, vol. 50, no. 3, pp. 778–791, 2020.

[118] P. Govender and V. Sivakumar, "Application of k-means and hierarchical clustering techniques for analysis of air pollution: A review (1980–2019)", *Atmospheric Pollution Research*, vol. 11, no. 1, pp. 40–56, 2020.

[119] L. Stanković, E. Sejdić, S. Stanković, M. Daković, and I. Orović, "A tutorial on sparse signal reconstruction and its applications in signal processing", *Circuits, Systems, and Signal Processing*, vol. 38, no. 3, pp. 1206–1263, 2019.

[120] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-based reconstruction of graph signals", *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 764–778, 2016.

[121] P. L. Fung *et al.*, "Input-adaptive proxy for black carbon as a virtual sensor", *Sensors*, vol. 20, no. 1, p. 182, 2020.

[122] M. A. Zaidan *et al.*, "Mutual information input selector and probabilistic machine learning utilisation for air pollution proxies", *Applied sciences*, vol. 9, no. 20, p. 4475, 2019.

[123] S. De Vito, E. Esposito, N. Castell, P. Schneider, and A. Bartonova, "On the robustness of field calibration for smart air quality monitors", *Sensors and Actuators B: Chemical*, vol. 310, p. 127 869, 2020.

[124] M. Matusowsky, D. T. Ramotsoela, and A. M. Abu-Mahfouz, "Data imputation in wireless sensor networks using a machine learning-based virtual sensor", *Journal of Sensor and Actuator Networks*, vol. 9, no. 2, p. 25, 2020.

[125] L. Liu, S. M. Kuo, and M. Zhou, "Virtual sensing techniques and their applications", in *2009 International Conference on Networking, Sensing and Control*, IEEE, 2009, pp. 31–36.

[126] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro, “Kernelized probabilistic matrix factorization: Exploiting graphs and side information”, in *Proceedings of the 2012 SIAM international Conference on Data mining*, SIAM, 2012, pp. 403–414.

[127] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering”, *Advances in neural information processing systems*, vol. 29, pp. 3844–3852, 2016.

[128] Y. Wu, D. Zhuang, A. Labbe, and L. Sun, “Inductive graph neural networks for spatiotemporal kriging”, *arXiv preprint arXiv:2006.07527*, 2020.

[129] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu, “Time-varying graph signal reconstruction”, *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 870–883, 2017.

[130] S. Wang, Y. Li, J. Zhang, Q. Meng, L. Meng, and F. Gao, “Pm2. 5-gnn: A domain knowledge enhanced graph neural network for pm2. 5 forecasting”, in *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, 2020, pp. 163–166.

[131] T. H. Do *et al.*, “Graph-deep-learning-based inference of fine-grained air quality from mobile iot sensors”, *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8943–8955, 2020.

[132] V.-D. Le, “Spatiotemporal graph convolutional recurrent neural network model for citywide air pollution forecasting”, *TechRxiv*, 2021.

[133] D. Hagan *et al.*, “Calibration and assessment of electrochemical air quality sensors by colocation with regulatory-grade instruments”, *Atmosph. Measurement Tech.*, vol. 11, no. 1, pp. 315–328, 2018.

[134] P. Ferrer-Cid, J. M. Barcelo-Ordinas, and J. Garcia-Vidal, “Data reconstruction applications for iot air pollution sensor networks using graph signal processing”, *Journal of Network and Computer Applications*, p. 103 434, 2022.

[135] M. E. Quinteros *et al.*, “Use of data imputation tools to reconstruct incomplete air quality datasets: A case-study in temuco, chile”, *Atmospheric environment*, vol. 200, pp. 40–49, 2019.

[136] X. Liu, X. Wang, L. Zou, J. Xia, and W. Pang, “Spatial imputation for air pollutants data sets via low rank matrix completion algorithm”, *Environment international*, vol. 139, p. 105 713, 2020.

[137] N. Okafor and D. Delaney, “Missing data imputation on iot sensor networks: Implications for on-site sensor calibration”, 2021.

[138] A. Mondal, M. Das, A. Chatterjee, and P. Venkateswaran, “Recovery of missing sensor data by reconstructing time-varying graph signals”, *arXiv:2203.00418*, 2022.

[139] A. Mnih and R. R. Salakhutdinov, “Probabilistic matrix factorization”, *Advances in neural information processing systems*, vol. 20, pp. 1257–1264, 2007.

[140] M.-Z. Zhang, L.-M. Wang, and S.-M. Xiong, “Using machine learning methods to provision virtual sensors in sensor-cloud”, *Sensors*, vol. 20, no. 7, p. 1836, 2020.

[141] G. Aiello, V. Chetta, M. Del Coco, E. Giangreco, S. Pino, and D. Storelli, “A virtual augmentation for air quality measurement sensor networks in smart cities”, in *2019 IEEE International Symposium on Measurements & Networking (M&N)*, IEEE, 2019, pp. 1–6.

[142] S. Guo, L. He, Y. Gu, B. Jiang, and T. He, “Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links”, *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2787–2802, 2013.

[143] I. Heimann *et al.*, “Source attribution of air pollution by spatial scale separation using high spatial density networks of low cost air quality sensors”, *Atmospheric Environment*, vol. 113, pp. 10–19, 2015.

[144] T.-B. Ottosen and P. Kumar, “Outlier detection and gap filling methodologies for low-cost air quality measurements”, *Environmental Science: Processes & Impacts*, vol. 21, no. 4, pp. 701–713, 2019.

[145] P. Ferrer-Cid, J. M. Barcelo-Ordinas, and J. Garcia-Vidal, “Volterra graph-based outlier detection for air pollution sensor networks”, *IEEE Transactions on Network Science and Engineering*, 2022.

[146] N. Zimmerman, “Tutorial: Guidelines for implementing low-cost sensor networks for aerosol monitoring”, *Journal of Aerosol Science*, vol. 159, p. 105 872, 2022.

[147] S. Munir, M. Mayfield, D. Coca, S. A. Jubb, and O. Osammor, “Analysing the performance of low-cost air quality sensors, their drivers, relative benefits and calibration in cities—a case study in sheffield”, *Environmental monitoring and assessment*, vol. 191, no. 2, p. 94, 2019.

[148] G. Miskell, J. A. Salmond, and D. E. Williams, “Solution to the problem of calibration of low-cost air quality measurement sensors in networks”, *ACS sensors*, vol. 3, no. 4, pp. 832–843, 2018.

[149] J. Li, A. Hauryliuk, C. Malings, S. R. Eilenberg, R. Subramanian, and A. A. Presto, “Characterizing the aging of alphasense no2 sensors in long-term field deployments”, *ACS sensors*, vol. 6, no. 8, pp. 2952–2959, 2021.

[150] L. Weissert *et al.*, “Low-cost sensor networks and land-use regression: Interpolating nitrogen dioxide concentration at high temporal and spatial resolution in southern california”, *Atmospheric Environment*, vol. 223, p. 117 287, 2020.

[151] D. Chen, C.-T. Lu, Y. Kou, and F. Chen, “On detecting spatial outliers”, *Geoinformatica*, vol. 12, no. 4, pp. 455–475, 2008.

[152] M.-F. Harkat, G. Mourot, and J. Ragot, “Sensor failure detection of air quality monitoring network”, *IFAC Proceedings Volumes*, vol. 33, no. 11, pp. 529–534, 2000.

[153] M. F. Harkat, G. Mourot, and J. Ragot, “Sensor fault detection and isolation of an air quality monitoring network using non linear principal component analysis”, in *16th IFAC World Congress*, Citeseer, 2005, pp. 4–8.

[154] T. Yu, X. Wang, and A. Shami, “Recursive principal component analysis-based data outlier detection and sensor data aggregation in iot systems”, *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2207–2216, 2017.

[155] V. Van Zoest, A. Stein, and G. Hoek, “Outlier detection in urban air quality sensor networks”, *Water, Air, & Soil Pollution*, vol. 229, no. 4, pp. 1–13, 2018.

[156] Z. Wang, J. Feng, Q. Fu, S. Gao, X. Chen, and J. Cheng, “Quality control of online monitoring data of air pollutants using artificial neural networks”, *Air Quality, Atmosphere & Health*, vol. 12, no. 10, pp. 1189–1196, 2019.

[157] D. Gong *et al.*, “Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection”, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1705–1714.

[158] S. Pidhorskyi, R. Almohsen, D. A. Adjeroh, and G. Doretto, “Generative probabilistic novelty detection with adversarial autoencoders”, *arXiv:1807.02588*, 2018.

[159] H. Wu *et al.*, “Probabilistic automatic outlier detection for surface air quality measurements from the china national environmental monitoring network”, *Advances in Atmospheric Sciences*, vol. 35, no. 12, pp. 1522–1532, 2018.

[160] Z. Xiao, H. Fang, and X. Wang, “Nonlinear polynomial graph filter for anomalous iot sensor detection and localization”, *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4839–4848, 2020.

[161] H. E. Egilmez and A. Ortega, “Spectral anomaly detection using graph-based filtering for wireless sensor networks”, in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 1085–1089.

[162] C.-T. Lu, D. Chen, and Y. Kou, “Algorithms for spatial outlier detection”, in *Third IEEE International Conference on Data Mining*, IEEE, 2003, pp. 597–600.

[163] Y. Kou, C.-T. Lu, and D. Chen, “Spatial weighted outlier detection”, in *Proceedings of the 2006 SIAM international conference on data mining*, SIAM, 2006, pp. 614–618.

[164] S. Shekhar, C.-T. Lu, and P. Zhang, “Detecting graph-based spatial outliers: Algorithms and applications (a summary of results)”, in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 371–376.

[165] G. Lewenfus, W. Alves Martins, S. Chatzinotas, and B. Ottersten, “On the use of vertex-frequency analysis for anomaly detection in graph signals”, *Anais do XXXVII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2019)*, pp. 1–5, 2019.

[166] K. Gopalakrishnan, M. Z. Li, and H. Balakrishnan, “Identification of outliers in graph signals”, in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 4769–4776.

[167] A. Gaddam, T. Wilkin, M. Angelova, and J. Gaddam, “Detecting sensor faults, anomalies and outliers in the internet of things: A survey on the challenges and solutions”, *Electronics*, vol. 9, no. 3, p. 511, 2020.

[168] Z. Zhang, A. Mehmood, L. Shu, Z. Huo, Y. Zhang, and M. Mukherjee, “A survey on fault diagnosis in wireless sensor networks”, *IEEE Access*, vol. 6, pp. 11 349–11 364, 2018.

[169] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: Identifying density-based local outliers”, in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[170] L. Xie, D. Pi, X. Zhang, J. Chen, Y. Luo, and W. Yu, “Graph neural network approach for anomaly detection”, *Measurement*, vol. 180, p. 109 546, 2021.

[171] G.-J. Qi and J. Luo, “Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[172] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era”, in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[173] Z. Xiao, H. Fang, and X. Wang, “Anomalous iot sensor data detection: An efficient approach enabled by nonlinear frequency-domain graph analysis”, *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3812–3821, 2020.

[174] Z. Xiao, H. Fang, and X. Wang, “Distributed nonlinear polynomial graph filter and its output graph spectrum: Filter analysis and design”, *IEEE Transactions on Signal Processing*, vol. 69, pp. 1–15, 2021.

[175] G. Leus, M. Yang, M. Coutino, and E. Isufi, “Topological volterra filters”, in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 5385–5399.

[176] M. O. Franz and B. Schölkopf, “A unifying view of wiener and volterra theory and polynomial kernel regression”, *Neural computation*, vol. 18, no. 12, pp. 3097–3118, 2006.

[177] C. O’Reilly, A. Gluhak, M. A. Imran, and S. Rajasegarar, “Anomaly detection in wireless sensor networks in a non-stationary environment”, *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1413–1432, 2014.