UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

ARTIFICIAL NEURAL NETWORKS APPLIED TO IMPROVE LOW-COST AIR QUALITY MONITORING PRECISION

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Sergio Malagón Fernández

In partial fulfilment

of the requirements for the degree in

TELECOMMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING

Advisor: Josep Paradells Aspas

Barcelona, July 2018

Abstract

It is a fact that air pollution is a major environmental health problem that affects everyone, especially in urban areas. Furthermore, the cost of high-end air pollution monitoring sensors is considerably high, so public administrations are unable to afford to place an elevated number of measuring stations, leading to the loss of information that could be very helpful.

Over the last few years, a large number of low-cost sensors have been released, but its use is often problematic, due to their selectivity and precision problems.

A calibration process is needed in order to solve an issue with many parameters with no clear relationship among them, which is a field of application of Machine Learning.

The objectives of this project are first, integrating three low-cost air quality sensors into a Raspberry Pi and then, training an Artificial Neural Network model that improves precision in the readings made by the sensors.

Resum

Està demostrat que la contaminació de l'aire és un gran problema per a la salut a nivell mundial, especialment en zones urbanes. A més, el cost dels sensors de contaminació de gama alta és considerablement alt, motiu pel qual els organismes públics no es poden permetre emplaçar una gran quantitat d'estacions de mesura, perdent informació que podria resultar molt útil.

Al llarg dels últims anys, han sorgit molts sensors de contaminació de baix cost, però el seu ús és sovint complicat, ja que tenen problemes de selectivitat i precisió.

Els objectius d'aquest projecte són primer de tot integrar tres sensors de contaminació de baix cost en una Raspberry Pi i sobre tot, entrenar un model basat en una xarxa neuronal artificial que millori la precisió de les lectures realitzades pels sensors.

Resumen

Es un hecho que la contaminación del aire es un gran problema para la salud a nivel mundial, especialmente en zonas urbanas. Además, el coste de los sensores de contaminación de gama alta es considerablemente alto, por lo que los organismos públicos no pueden permitirse emplazar un gran número de estaciones de medida, perdiendo información que podría ser muy útil.

A lo largo de los últimos años, han surgido muchos sensores de contaminación de bajo coste, pero su uso suele ser complicado, ya que tienen problemas de selectividad y precisión.

Los objetivos de este proyecto son primero integrar tres sensores de contaminación de bajo coste en una Raspberry Pi y sobre todo, entrenar un modelo basado en una red neuronal artificial que mejore la precisión de las lecturas realizadas por los sensores.

Acknowledgements

I would like to start by thanking my advisor for this project, Josep Paradells, for giving me the opportunity to carry out this project and his guidance along the process.

I would also like to thank Marisa Catalán for her help at CSIC.

Last but not least, I want to thank my family, friends and you, Natalia, for all the unconditional support throughout all these years, keeping my head clear at all times.

Revision history and approval record

Revision	Date	Purpose
0	03/06/2018	Document creation
1	07/06/2018	Document revision
2	13/06/2018	Document revision
3	25/06/2018	Document revision
4	01/07/2018	Final document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Sergio Malagón	smalagon96@gmail.com
Josep Paradells	teljpa@entel.upc.edu

Written by:		Reviewed and approved by:	
Date	01/07/2018	Date	01/07/2018
Name	Sergio Malagón	Name	Josep Paradells
Position	Project Author	Position	Project Supervisor

Table of contents

Abstract 1
Resum 2
Resumen 3
Acknowledgements 4
Revision history and approval record 5
Table of contents 6
List of Figures 8
List of Tables: 9
1. Introduction 10
1.1. Overview 10
1.2. Objectives 10
1.3. Work Plan 10
1.3.1. Incidents and delays 11
2. State of the art 12
2.1. Air quality monitoring 12
2.2. Machine Learning 13
2.2.1. Supervised learning 13
2.2.2. Unsupervised learning 13
2.2.3. Artificial Neural Networks 14
2.2.3.1. Human neurons 14
2.2.3.2. Perceptron 14
2.2.3.3. Multi-layer neural networks 15
2.2.3.4. Activation functions 15
2.2.3.5. Backpropagation 16
3. Methodology 19
3.1. Problem Statement 19
3.2. Sensing platform 19
3.3. Data collection 20
3.4. Provided data 21
3.5. Model building 21
3.5.1. Framework 22
3.5.2. Data pre-processing 22

3.5.3. Model architecture and training 23
3.5.3.1. K-Fold Cross-Validation 24
4. Results 25
4.1. CO targeted model 25
4.1.1. Feature importance 25
4.1.2. Network architecture and results 25
4.2. NO_2 targeted model 26
4.2.1. Feature importance 26
4.2.2. Network architecture and results 27
4.3. Final discussion 27
5. Budget 29
6. Environmental Impact 30
7. Conclusions and future development 31
Bibliography 32
Glossary 33

List of Figures

Figure 1: Project Gantt diagram 11

Figure 2: Representation of a single human neuron 14

Figure 3: Structure of a perceptron 14

Figure 4: Structure of a common Artificial Neural Network 15

Figure 5: Common activation functions 16

Figure 6: For simplicity, we can portray the loss function as a 3D function. Through each iteration of the gradient descent algorithm, we pursue the coordinates at the minimum ..17

Figure 7: Schematic representation of the connections between the sensors and the Raspberry Pi 20

Figure 8: Structure of the SQLite database 21

Figure 9: Temporal evolution of the CO and NO_2 concentrations along the measuring campaign. In order to obtain a clearer plot, only the weekly mean of the concentrations is shown. 22

Figure 10: Average CO and NO_2 concentrations for each day of the week 23

Figure 11: Representation of 10-Fold Cross-Validation 24

Figure 12: CO targeted model feature importance. Features labeled as (ref) correspond to the high-end reference sensors, while those labeled by (lc) correspond to low-cost sensors 25

Figure 13: Plot of the CO targeted model training and validation errors during a single fold of the cross-validation process 26

Figure 14: CO targeted model feature importance. Features labeled as (ref) correspond to the high-end reference sensors, while those labeled by (lc) correspond to low-cost sensors 26

Figure 15: Plot of the NO_2 targeted model training and validation errors during a single fold of the cross-validation process 27

Figure 16: Diagram of the Artificial Neural Networks that form the CO targeted model (left) and the NO_2 targeted model (right) 28

List of Tables:

Table 1: WHO Guidelines for particulate matter, ozone, nitrogen dioxide and sulphur dioxide. 12

Table 2: Error (MSE) of the CO targeted model for different network architectures 26

Table 3: Error (MSE) of the NO_2 targeted model for different network architectures 27

Table 4: Final performance metrics for both the CO targeted and the NO_2 targeted model, expressed in MSE and RMSE 28

Table 5: Budget breakdown 29

1. Introduction

1.1. Overview

It is a fact that air pollution is a major environmental health problem that affects everyone, especially in urban areas. Numerous studies, as [1], have determined that outdoor air pollution is carcinogenic to humans, as well as it increases notably the risk of suffering from cardiovascular and respiratory diseases.

The first step to address the problem is to measure it. However, the cost of current high-end air pollution monitoring sensors is considerably high. Consequently, public administrations are unable to afford to place an elevated number of measuring stations, leading to the loss of information that could be very helpful.

The obvious solution is to use low-cost sensors, but its selectivity and precision are problematic. A calibration process is needed in order to solve an issue with many parameters with no clear relationship among them, which is a field of application of Machine Learning

1.2. Objectives

The aim of this project is to integrate three air quality sensors into a Raspberry Pi and implement a Machine Learning algorithm that improves precision in the readings made by the sensors.

In order to train the model, the Raspberry Pi and the sensors were placed alongside a high-end air quality measuring station at CSIC (Centro Superior de Investigaciones Científicas) for 7 days. The measures taken there should have played an essential role in the development of the algorithm but, due to an unexpected issue that was not possible and the model had to be build based on a public dataset provided by the ENEA (National Agency for New Technologies, Energy and Sustainable Economic Development). For more information on that issue, please refer to Section 1.3.1.

1.3. Work Plan

This section presents the structure of the project, divided into work packages, along with a Gantt diagram with the temporary planning. In addition, incidents that occurred during the realization of the project are explained.

The project has been divided from the beginning into six work packages, which are the following:

- WP1: Introduction to Machine Learning
- WP2: Pollution and sensor research
- WP3: Sensing platform building
- WP4: Data acquisition
- WP5: Calibration with an Artificial Neural Network model

- WP6: Documentation

TASKS	START	END	DURATION	FEBRUARY				MARCH				APRIL				MAY				JUNE			
				W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
1. INTRODUCTION TO MACHINE LEARNING	03/02/2018	16/02/2018	13																				
1.1 Basic Machine Learning Concepts	03/02/2018	07/02/2018	4																				
1.2 Machine Learning with Python	07/02/2018	19/02/2018	12																				
2. POLLUTION AND SENSOR RESEARCH	19/02/2018	06/03/2018	15																				
2.1 Air pollution and health	19/02/2018	26/02/2018	7																				
2.2 Air quality sensors	26/02/2018	07/03/2018	9																				
3. SENSING PLATFORM BUILDING	07/03/2018	15/05/2018	69																				
3.1 Raspberry Pi Setup	07/03/2018	18/04/2018	42																				
3.2 Sensor configuration	18/04/2018	15/05/2018	27																				
4. DATA ACQUISITION	15/05/2018	20/06/2018	36																				
4.1 Preparation	15/05/2018	23/05/2018	8																				
4.2 Joint data acquisition with CSIC	13/06/2018	20/06/2018	7																				
5. CALIBRATION WITH ANN MODEL	01/06/2018	19/06/2018	18																				
6. DOCUMENTATION	19/06/2018	02/07/2018	13																				

Figure 1: Project Gantt diagram

1.3.1. Incidents and delays

As the project progressed, some incidences were found, which delayed the work package completion and ultimately led to the non-completion of the original plan.

First of all, the biggest incident was that the data acquisition done at CSIC did not work as expected. The sensing platform was placed alongside the high-end measuring station for seven days, but it only retrieved ten hours of data. The cause of that issue was investigated, but the conclusion was that it was caused by a programming error, due to the fact that until that moment, the acquisition script had not been running for so much time.

Fortunately, there was an alternative plan in order to be able to fulfill the objective of the project. The learning model had to be built based on a dataset provided by the ENEA (National Agency for New Technologies, Energy and Sustainable Economic Development) that contains the hourly responses of a gas multisensor device deployed in an Italian city for one year, recorded along with gas concentrations references from a certified analyzer. Adapting to the new plan was not extremely difficult because, during the data acquisition process, the first trials on applying learning models had been computed on that dataset.

Moreover, delays in relation with the original work plan were caused by two main factors:

- Lack of experience with the Raspberry Pi, which led to a slight delay in the setup and sensor integration.
- Schedule incompatibilities with the project supervisor, which led to slight delays in some stages of the project.

2. State of the art

The first section consists of a brief overview of the most common pollutants found in our environment, as well as a reference to numerous studies on low-cost air pollution monitoring.

The second section is the main section of this chapter and at first is focused on introducing the concept of Machine Learning.

A more thorough overview on Artificial Neural Networks is then provided, allowing that by the end of this chapter, the reader will have the necessary background for this project.

2.1. Air quality monitoring

As previously stated, it is a fact that air pollution is a major concern due to its significant impacts on human health and global environment. A huge number of hazardous compounds have been identified in our environment, but six of the most common pollutants are carbon monoxide (CO), nitrogen dioxide (NO_2), ground-level ozone (O_3), sulphur dioxide (SO_2), particulate matter (PM) and lead (Pb). A detail on their effects on human health can be found on [2].

In order to reduce effects on the population, governments and organizations have put limits on these pollutants. The table below shows the acceptable thresholds for these compounds, as stated by the WHO (World Health Organization) [3]

Table 1: WHO Guidelines for particulate matter, ozone, nitrogen dioxide and sulphur dioxide.

Pollutant	Thresholds
Carbon Monoxide (CO)	100 mg/m^3 (15 min) 15 mg/m^3 (1 h) 10 mg/m^3 (8 h) 7 mg/m^3 (24 h)
Nitrogen Dioxide (NO_2)	200 µg/m^3 (1 h) 40 µg/m^3 (1 year)
Ozone (O_3)	100 µg/m^3 (8 h)
Sulphur Dioxide (SO_2)	500 µg/m^3 (10 min) 20 µg/m^3 (24 h)
Particulate Matter ($PM_{2.5}$)	25 µg/m^3 (24 h) 10 µg/m^3 (1 year)
Particulate Matter (PM_{10})	50 µg/m^3 (24 h) 20 µg/m^3 (1 year)
Lead (Pb)	0.5 µg/m^3 (1 year)

With the objective of retrieving as much information as possible keeping a low cost, numerous studies have been carried out about the architectures of Wireless Sensor Networks for air quality monitoring and the effective calibration of the low-cost sensors, such as [4]–[9].

2.2. Machine Learning

Machine learning is one of the mainstays of modern-day information technology and is all around us, although we may not even realise we are using it.

Arthur Samuel, who coined the term machine learning in 1959, defined it as: *"the field of study that gives computers the ability to learn without being explicitly programmed."* [10]

It basically relies on algorithms allowing computers to find patterns in big chunks of data, just as a human might be able to do, although at a more extensive scale.

In general, any application of Machine Learning can be assigned to one of two broad classifications: supervised learning and unsupervised learning.

2.2.1. Supervised learning

In supervised learning, the dataset that is fed to the algorithm is formed by input and output values so, throughout the training process, the algorithm can establish a kind of relationship between the input and the output. That relationship will be used to predict the output when given a new dataset where only the input is known.

There are two different types of supervised learning problems: regression and classification.

In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories. [11]

An example of a regression problem would be trying to estimate the price of a house given its characteristics, such as size, number of bedrooms and neighbourhood where it is located.

An example of a classification problem would be trying to determine if given an image of an animal, trying to predict its species.

2.2.2. Unsupervised learning

In unsupervised learning, there is no knowledge of the output values, so it is not possible to train an algorithm to predict a certain outcome. Unsupervised learning is instead used to discover the underlying structure of the data, making clustering[1] possible.

An example of unsupervised learning would be classifying the customers of a certain business into market segments.

[1] Clustering: Finding subsets of data that are similar

2.2.3. Artificial Neural Networks

2.2.3.1. Human neurons

Artificial neural networks were developed mimicking how neurons interact in our brain. So, in order to introduce them, let's start by looking how a single brain neuron looks like.

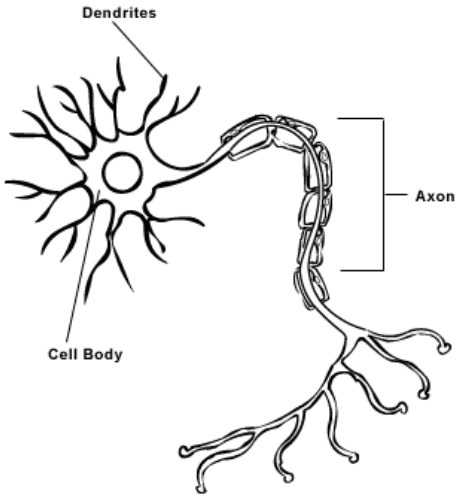

Figure 2: Representation of a single human neuron

A neuron has a cell body with a number of input wires, called dendrites, which receive inputs from other cells, and an output wire, called an axon, which is used to send signals to other neurons.

So, at a simplistic level, a neuron is a computational unit that gets a number of inputs, does some computation and sends an output, a little electric pulse, through its axon to other neurons in the brain. Those neurons, in turn, do further computation and send outputs to other neurons. Basically, this is the process by which all human thought happens.

2.2.3.2. Perceptron

The first step into Artificial Neural Networks was made by Frank Rosenblatt who, inspired by earlier work by Warren McCulloch and Walter Pitts, developed the perceptron in the late 1950's - early 1960's, also known as Single layer Neural Network. [12]

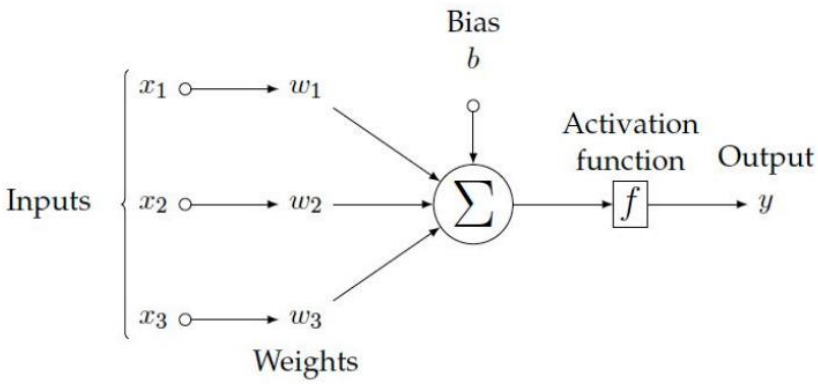

Figure 3: Structure of a perceptron

Figure 3 depicts the structure of a simple perceptron, or neuron. In this case, it has three inputs but in general, it could have any amount of them. In order to compute the output, Rosenblatt introduced the concept of 'weights', which are real numbers expressing the relevance to the output of the respective inputs. A bias unit is usually added to help to tune the desired outcome.

The output of a neuron is computed by a certain function, called the activation function (see section 2.2.3.4) In the case of Rosenblatt's perceptron, a step function is applied, leading to the following output:

$$f(\boldsymbol{x}, \boldsymbol{w}) = \begin{cases} 1, & if\ \boldsymbol{w} \cdot \boldsymbol{x} + b > 0 \\ 0, & otherwise \end{cases}$$

Equation 1: Step function used in Rosenblatt's perceptron

where $\boldsymbol{x}$ and $\boldsymbol{w}$ are vectors whose components are the inputs and weights of the perceptron and b is the bias unit.

Therefore, the output depends on a weighted sum of the inputs ($\boldsymbol{w} \cdot \boldsymbol{x} = \sum_j w_j x_j$), where the importance of each input is taken into account.

2.2.3.3. Multi-layer neural networks

Typically neurons are arranged into layers that all together form a network of neurons, just as in a human brain. Different layers may perform different kinds of transformations on their inputs.

The naming for the layers in an ANN is the following:

- Input layer: The layer that deals with the input values directly.
- Output layer: The layer that is responsible for outputting a value or a vector of values, relevant to the problem.
- Hidden layers: The layers that are neither an input nor an output layer and therefore are not exposed to the outside of the network. When there are many hidden layers, it is said that the network is a **deep neural network**.

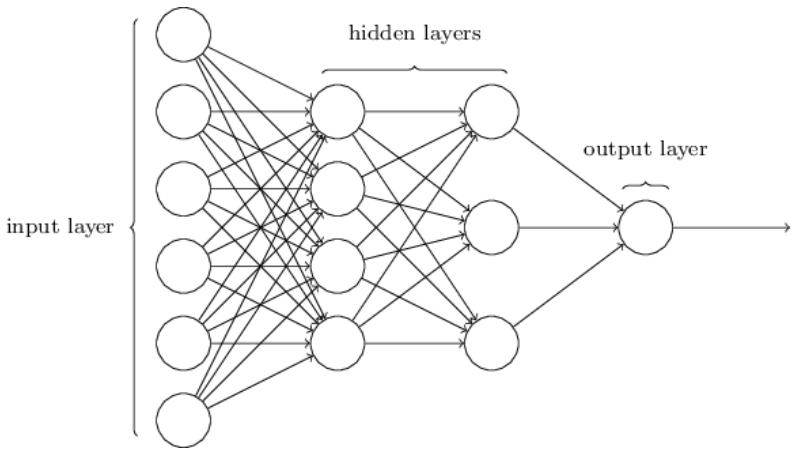

Figure 4: Structure of a common Artificial Neural Network

2.2.3.4. Activation functions

Firstly, when the perceptron was developed, the only activation function was the step function, as we have just seen.

Nowadays, a wide range of functions can be used. The main reason behind using activation functions is to introduce non-linearity in the model, leading to a smoother behaviour. The idea is that small changes in the weights cause small changes in the output and therefore, the learning process is easier.

Some of the common activation functions are shown below:

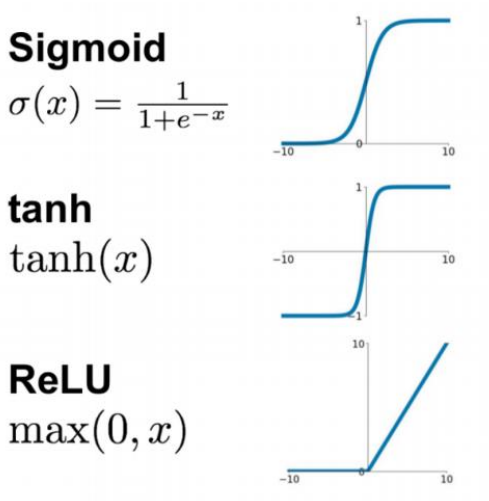

Figure 5: Common activation functions

It is important to keep in mind that, as an exception, when using neural networks applied to a regression problem, like we will do, the activation function in the output layer must be linear so the outcome is as expected.

2.2.3.5. Backpropagation

Now that we know the elements of a neural network, it's time to understand its most essential process, the learning process, which is called backpropagation.

The learning process takes the inputs and the desired outputs and updates the model's weights. Then, an output is generated recursively according to past training experience.

An iteration of the learning process can be decomposed into five blocks:

1. **Model initialization**: Neural networks can start from anywhere and the most common is to initialize the model randomly because no matter from where it starts, after the process the most accurate model will be achieved.
2. **Forward propagation**: After initializing the model, the natural step is to check its performance. Therefore, training input values are passed through the network and an output is obtained.
3. **Evaluating loss**: The actual output obtained at first is almost always wrong, because, at this stage, the network is unable to make any reasonable conclusion. Therefore, we need to compare it with the desired output we would like the network to learn.

   To do so, a loss or cost function is computed. The most common loss function is the Mean Squared Error (MSE):

$$C(\boldsymbol{w}, \boldsymbol{b}) = \frac{1}{N}\sum_{i=1}^{N} \|\hat{y}_i - y_i\|^2$$

Equation 2: Minimum Square Error cost function

where y_i is the desired output, $\hat{y}_i$ is the predicted output, $\boldsymbol{w}$ represents the weights in the network and **b**, the biases.

From now on, the objective will be to minimize that function tuning the values of weights and biases.

4. **Backpropagation**: The following step is to propagate the error backwards through the network, finding the contribution of each weight to the loss function. After backpropagation, we obtain the gradient of the loss function(∇C), influenced by all weights in the network.
5. **Weight update**: Once ∇C is computed, weights have to be updated in order to minimize error. The gradient ∇C indicates the direction of increase of the loss function, so each weight will be updated along the negative gradient direction.

That is done through the gradient descent algorithm:

$$w_j \leftarrow w_j - \alpha \nabla C$$

Where w_j is any weight and α is the learning rate.

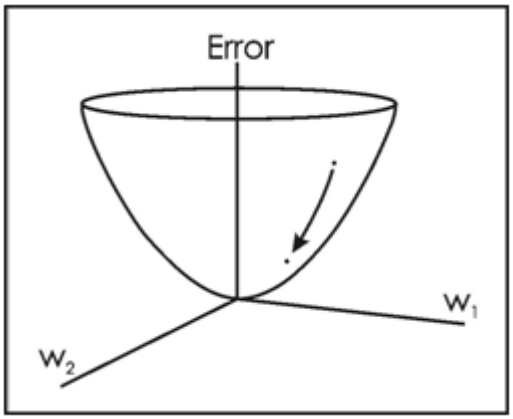

Figure 6: For simplicity, we can portray the loss function as a 3D function. Through each iteration of the gradient descent algorithm, we pursue the coordinates at the minimum

The learning rate is a parameter that can be chosen by the creator of the model. A high learning rate means that bigger steps are taken in the weight updates and thus, it may take less time for the model to converge to an optimal set of weights. However, an excessively high learning rate will result in too large steps and the algorithm may fail to converge.

However, in practice, improved optimization algorithms are used. Adam [13] is one of the most recent optimization algorithms, which has two main features:

- Each weight has its own individual learning rate.

- Learning rates are not constant. Instead, they keep adapting with each iteration based on the average of recent magnitudes of the gradients of the weight that represent how quick it is changing.

3. Methodology

3.1. Problem Statement

The main objective of this project is to build a model formed by an Artificial Neural Network that improves precision in the readings made by low-cost air quality sensors. In order to do so, a low-cost sensing platform has been built on top of a Raspberry Pi. However, due to the issue stated in Section 1.3.1, it has been impossible to retrieve valuable data and the model has been built using a public dataset containing readings from low-cost and high-end reference sensors.

3.2. Sensing platform

The low-cost air quality sensing platform built for this project is made of the following elements:

- **Raspberry Pi 3**

  The Raspberry Pi is a single-board computer developed in the United Kingdom by the Raspberry Pi Foundation in order to promote the teaching of computer science in schools. Due to its low cost, it has become very popular, even for uses outside its original target market, such as robotics.

  The official operating system is called Raspbian, an adapted version of Debian. Even though, other operating systems are supported.

  In this project, we are working with a Raspberry Pi 3 Model B, which is the latest model as of today.

- **SDS-011 PM sensor**

  The SDS-011 is a particulate matter sensor developed by Inovafit, a spin-off company from the University of Jinan. It uses the principle of laser scattering[2] to detect particle concentration in the air between 0.3 and 10 µm with a relative error of 10 %, according to the datasheet [14].

  It is connected to the Raspberry Pi through a USB port and it outputs two different categories, PM_{10} and $PM_{2.5}$.

- **Grove Multichannel Gas sensor**

  The Grove Multichannel Gas sensor is developed by Seeed Studio and has a built-in MICS-6814 analogic sensor.

  It is connected to the Raspberry Pi through the I^2C interface and in this project, it will be used to measure:

  – CO concentrations, with a detection range between 1 and 1000 ppm[3] and a sensitivity factor[4] between 1.2 and 50 [15].

[2] Laser scattering is a technique used to determine the size of small particles in suspension. The presence of particles is transformed into electrical signals.
[3] PPM: parts per million
[4] The sensitivity factor is defined as the change in the output of the sensor per unit change in the parameter being measured.

– NO_2 concentrations, with a detection range between 0.05 and 10 ppm and a sensitivity factor of 2.

- **DHT-22 sensor**

The DHT-22 sensor is developed by rDuinoStar and outputs values for temperature and relative humidity with a ±2 % error, which increases to ±5 % at the extreme limits.

It is connected to the Raspberry Pi through a GPIO pin.

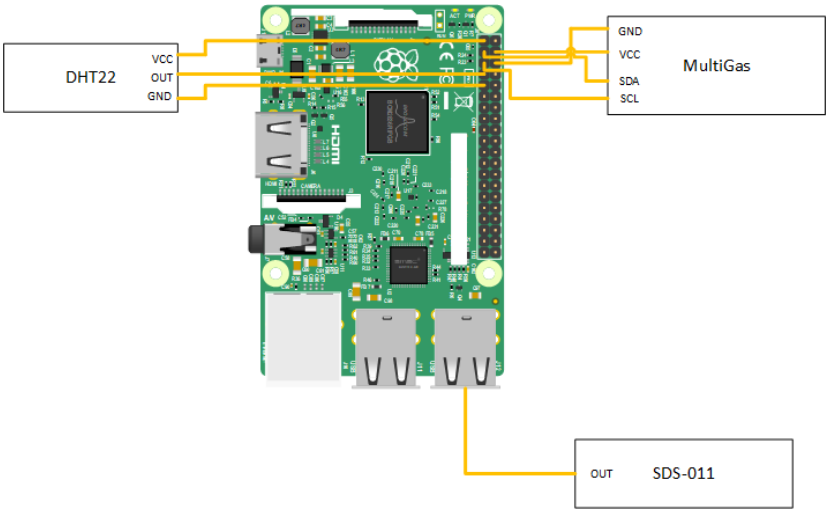

Figure 7: Schematic representation of the connections between the sensors and the Raspberry Pi

3.3. Data collection

The Raspberry Pi is programmed with Python to retrieve data from each of the three sensors every 30 seconds. Data is actually retrieved every 80 seconds, due to the fact that the execution of the script in charge of the PM sensor reading causes an extra delay of 60 seconds.

When the station is turned on, an internal timer is started that, knowing the starting date and time, allows to relate each measurement to a date and time. This is done due to the fact that it is impossible to directly retrieve a timestamp during the measuring campaign because we have not been able to get access to an Internet connection at CSIC facilities.

In order to retrieve values from each sensor, the following external libraries have been used:

- Adafruit Python DHT Sensor Library, written by Tony DiCola [16].
- Grove Multichannel Gas Sensor Library, written in C by Paul van Haastrecht [17]. A Python wrapper has been written in order to interact with the sensor more easily.
- Nova PM Sensor Library, written in Python by Martin Luetzelberger [18].

Once they have been retrieved, the measurements are registered in an SQLite database that is queried at the end of the measuring campaign.

readings	
id	INTEGER
timestamp	NUMERIC
temperature	NUMERIC
humidity	NUMERIC
pm25	NUMERIC
pm10	NUMERIC
CO	NUMERIC
NO2	NUMERIC

Figure 8: Structure of the SQLite database

3.4. Provided data

Due to the issues stated in Section 1.3.1, the model has been built based on a dataset provided by the ENEA (National Agency for New Technologies, Energy and Sustainable Economic Development) [7], hosted on the UCI Machine Learning Repository [19].

The dataset contains 9358 instances of hourly averaged responses from a device with 5 low-cost air quality sensors. The device was located in a significantly polluted area within an Italian city. Data was recorded from March 2004 to February 2005 (one year) along with a certified analyzer that provided ground truth values.

Missing values are tagged with the '-200' value.

The provided attributes are the following:

- 0: Date (DD/MM/YYYY)
- 1: Time (HH.MM.SS)
- 2: True hourly averaged CO concentration in mg/m^3 (reference analyzer)
- 3: Hourly averaged CO targeted sensor response
- 4: True hourly averaged Non-Methane Hydrocarbons (NMHC) concentration in µg/m^3 (reference analyzer)
- 5: True hourly averaged Benzene concentration in µg/m^3 (reference analyzer)
- 6: Hourly averaged NMHC targeted sensor response
- 7: True hourly averaged NOx concentration in ppb (reference analyzer)
- 8: Hourly averaged NOx targeted sensor response
- 9: True hourly averaged NO2 concentration in µg/m^3 (reference analyzer)
- 10: Hourly averaged NO2 targeted sensor response
- 11: Hourly averaged O3 targeted sensor response
- 12: Temperature in °C
- 13: Relative Humidity (%)
- 14: Absolute Humidity

3.5. Model building

The objective of this project is to build a model capable of predicting accurately CO and NO_2 concentrations.

In order to improve the precision of the predictions, two separate models have been trained: one for each compound. Their input features will be the readings provided by the low-cost and reference sensors (excluding the reference sensor of the compound to be predicted), as well as extra features whose addition is considered appropriate.

3.5.1. Framework

In order to manipulate data, build the model and evaluate the results, different software has been used. A summary can be found below:

- Pandas [20] and Numpy [21] have been used for data manipulation.
- Keras [22] has been used to build the model. It is a high-level framework that runs on top of Tensorflow, one of the most used libraries for machine learning, developed by Google.
- Scikit-learn [23] has been used for the training of the model, data pre-processing and metrics evaluation.
- Matplotlib [24] has been used for data visualization.

3.5.2. Data pre-processing

First of all, data is obtained from a CSV file with the structure stated in Section 3.4.

After inspecting the file, two empty columns were found (columns 15 and 16), as well as a column with more than 90 % missing values (column 4, related to the true hourly NMHC concentration). Therefore, those three columns were excluded from the dataset.

Moreover, data was found with missing values, which were marked as '-200'. As a consequence, all readings with missing values were also excluded from the dataset.

The next step was to visualize the evolution of the two compounds upon which the model was going to be built.

Figure 9 shows the evolution of the CO and NO_2 concentration, as measured by the reference analyzer. This suggests a seasonality, where pollution values are higher on certain months, especially the ones that correspond to winter.

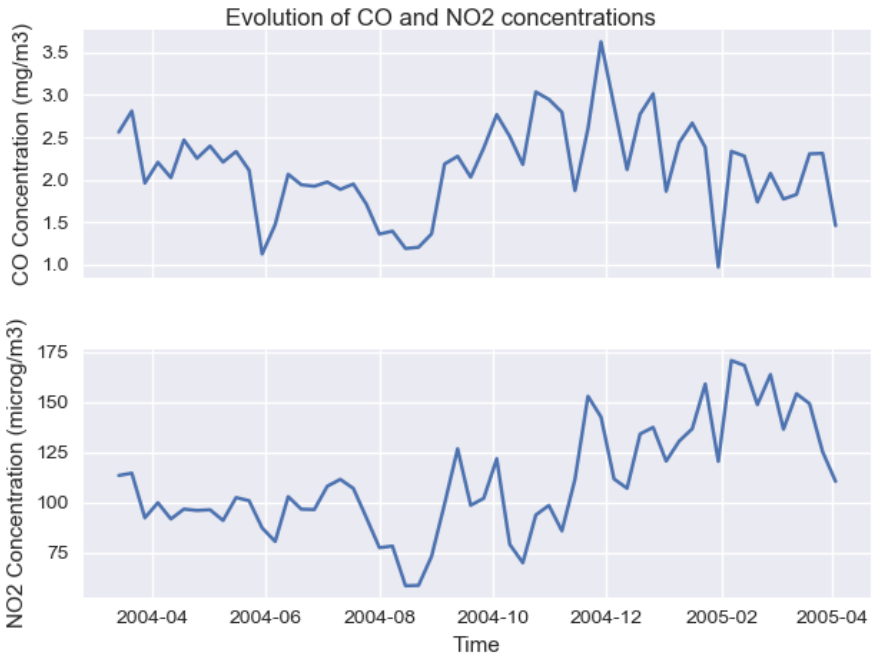

Figure 9: Temporal evolution of the CO and NO_2 concentrations along the measuring campaign. In order to obtain a clearer plot, only the weekly mean of the concentrations is shown.

To take that into account, an extra variable was added, assigning to each measurement an integer number relative to the month when it was taken.

Furthermore, figure 10 shows the average CO and NO_2 concentrations for each day of the week and it can be seen clearly that concentrations decrease on weekends.

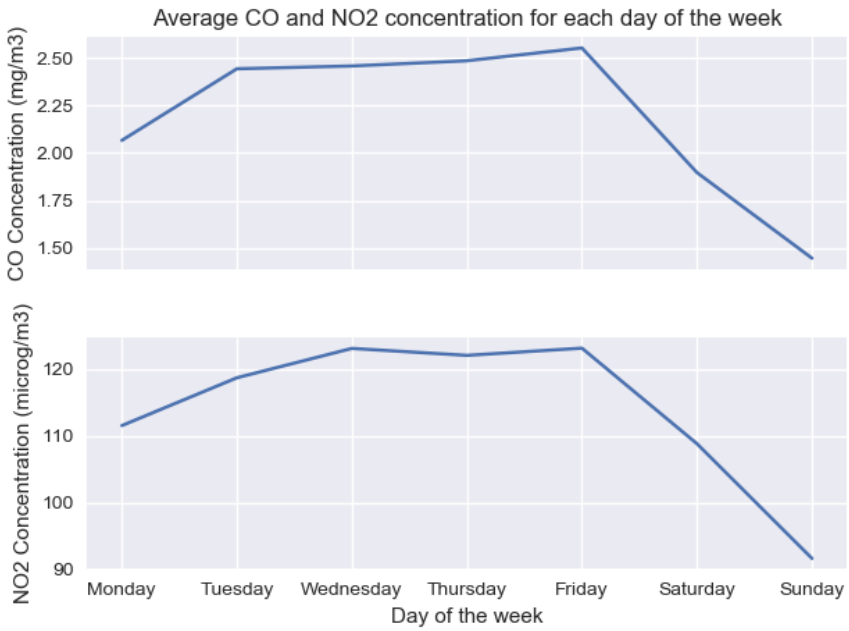

Figure 10: Average CO and NO_2 concentrations for each day of the week

To take that into account, another extra variable was added, assigning to each measurement an integer number relative to the day of the week when it was taken.

Finally, in order to be able to obtain reasonable results, the MinMaxScaler algorithm (in Scikit-learn library) was applied, which converts all values in each feature into the range [0,1].

$$x_{i\_rescaled} = \frac{x_{i\_original} - \min(x)}{\max(x) - \min(x)}$$

3.5.3. Model architecture and training

As previously stated, two different models have been trained in order to improve the precision of low-cost CO and NO_2 measurements, with the objective to minimize the Minimum Square Error (MSE) of the predictions made by the model.

For each case, a neural network has been trained with the following hyperparameters fixed from the start:

- Adam optimizer. As explained in Section 2.2.3.5, Adam is one of the most recent optimization algorithms and it generalises well to most machine learning problems, without any need to tune its parameters.

- Batch size[5]. The batch size is set to 32, due to the fact that larger batches degrade the quality of the model, reducing its ability to generalize [25].
- Activation function in the hidden layers: ReLU. As there are not any negative values in the dataset, the Rectified Linear Unit function has provided the best results.

There are other hyperparameters and configurations that were tested in order to find the best possible model. These are:

- Number of layers and the number of neurons per layer
- Number of epochs[6].

3.5.3.1. K-Fold Cross-Validation

K-Fold cross-validation has been used to validate the model performance. It is a popular method because it generally results in a less biased estimate of the model skill than other methods, such as a simple train/test split.

The general procedure is the following:

1. The dataset is shuffled randomly and split into k groups
2. For each unique group/fold:
    1. The group is taken as a test dataset
    2. The remaining groups are taken as a training dataset.
    3. The model is fit on the training set and evaluated on the test set.
    4. The evaluation score of the fold is retained
3. The skill of the model is summarized computing the mean of the obtained scores

It is important to take into account that, after the whole procedure is completed, each data sample has been given the opportunity to be used in the test set one time and used to train the model k-1 times.

In this project, 10-Fold Cross-Validation has been used, as this value has been shown empirically to minimize bias and variance in test error rates. Therefore, in each fold, 10% of the dataset has been used as training data and 90% as test data.

Figure 11: Representation of 10-Fold Cross-Validation

[5] The dataset is divided into batches in order to be able to pass it through the network. The batch size is the number of training examples contained in a single batch.
[6] The number of epochs determines how many times the entire dataset is passed (forward and backwards) through the dataset in the training phase.

4. Results

In this section, results from both the CO and NO_2 targeted models are shown. Some of the different configurations are discussed along with the different result interpretations.

Performance is measured by computing the Mean Square Error (MSE) because it is a metric that especially penalizes large errors, which are particularly undesirable. The Root Mean Square Error (RMSE) is computed at the end of the section in order to provide a clearer idea of the error magnitude.

The importance of each feature with respect to the reference targets is also shown, as computed through the Extra Trees algorithm[7]. This allows us to visualize the features (readings from certain sensors or extra variables) that make a bigger contribution to the final model.

4.1. CO targeted model

4.1.1. Feature importance

Figure 12 shows the importance of each feature in relation to the CO reference concentration.

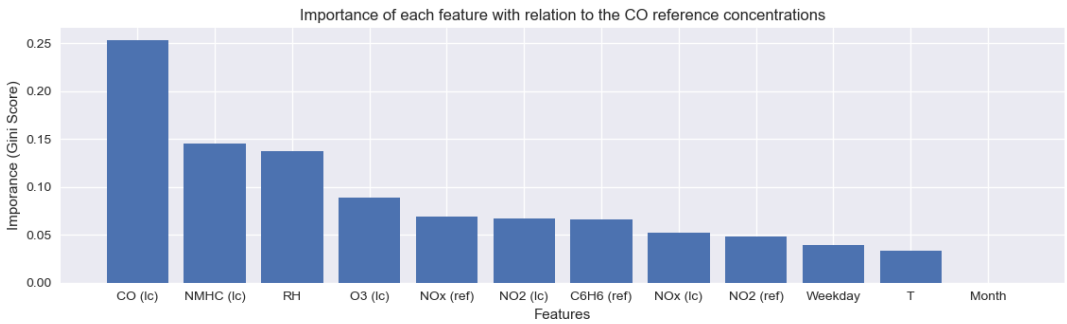

Figure 12: CO targeted model feature importance. Features labeled as (ref) correspond to the high-end reference sensors, while those labeled by (lc) correspond to low-cost sensors

As expected, the feature corresponding to the low-cost CO sensor is the most important of them all by far. Data retrieved by the NMHC and O_3 sensors is also relevant, which exposes the existing correlation among readings from different sensors. Relative humidity is also a relevant feature, showing that there is also a relation between the presence of CO and ambient factors.

4.1.2. Network architecture and results

In order to find the model with the lowest possible error, different architectures have been tested, tuning both the width (number of neurons per layer) and depth (total number of layers) of the neural network. The results can be seen in the table below:

[7] The Extra Trees algorithm is a variant of the Random Forest algorithm, that computes a metric called the Gini Importance that allows us to visualize the importance of each feature in relation to the output of the model

Table 2: Error (MSE) of the CO targeted model for different network architectures

# of neurons	1 hidden layer	2 hidden layers	3 hidden layers
5	0.001503	**0.001447**	0.002672
10	0.001511	0.001518	0.00152
15	0.001591	0.001629	0.001612
20	0.001672	0.001644	0.001666

The conclusion is that the architecture that minimizes MSE is that composed of two hidden layers with five neurons each.

Trying to improve performance, the training and validation errors were plotted for one fold in the cross-validation process.

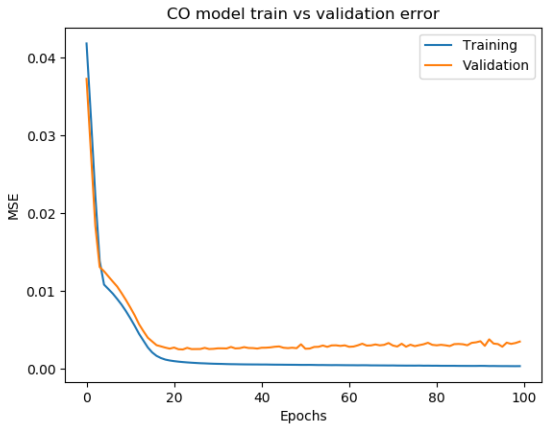

Figure 13: Plot of the CO targeted model training and validation errors during a single fold of the cross-validation process

We can see that there is a slight increase in validation error from epoch number 30 onwards. Therefore, in order to reduce error and training time, the number of epochs is set to 30. The MSE value is then reduced to **0.001360**.

4.2. NO_2 targeted model

4.2.1. Feature importance

Figure 14 shows the importance of each feature in relation to the NO_2 reference concentration.

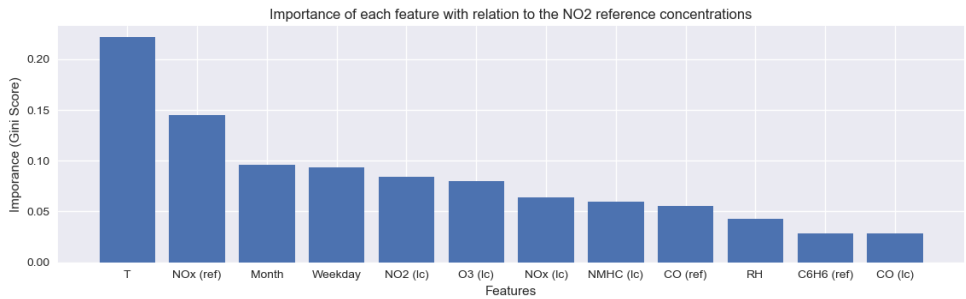

Figure 14: CO targeted model feature importance. Features labeled as (ref) correspond to the high-end reference sensors, while those labeled by (lc) correspond to low-cost sensors

In this case, the feature corresponding to the NO_2 low-cost sensor is only the fifth most important, which hints that it has a weak performance. Even though, the existence of a correlation between this compound, air temperature and the extra variables corresponding to the month and weekday of every measure helps the model obtain decent results. Correlation between NO_2 and another species whose sensors show good performance also help.

4.2.2. Network architecture and results

As we have done with the CO targeted model, different architectures have been tested, tuning both the width and depth of the neural network. The results can be seen in the table below:

Table 3: Error (MSE) of the NO_2 targeted model for different network architectures

# of neurons	1 hidden layer	2 hidden layers	3 hidden layers	4 hidden layers
5	0.005050	0.004952	0.005007	0.011981
10	0.004737	0.004543	0.004423	0.004880
15	0.004657	0.004363	0.004265	0.004541
20	0.004388	**0.003891**	0.003995	0.004145

The conclusion is that the architecture that minimizes MSE is that composed by two hidden layers with twenty neurons each.

Training and validation errors were also plotted for this model (see figure 15). In this scenario, validation error remains fairly constant over the last epochs. Then, the number of epochs is set to 80. The MSE value is then reduced to **0.003886**, which is practically equal to what we had obtained with 100 epochs, but training time is reduced.

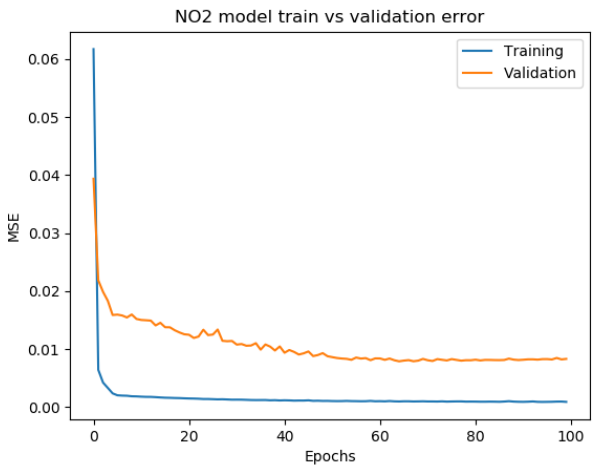

Figure 15: Plot of the NO_2 targeted model training and validation errors during a single fold of the cross-validation process

4.3. Final discussion

After having found the models that provide the best results for both cases, it would be interesting to compare their results along with their architecture.

Table 4: Final performance metrics for both the CO targeted and the NO_2 targeted model, expressed in MSE and RMSE

Model	Error (MSE)	Error (RMSE)
CO targeted model	0.001360	0.036878 mg/m^3
NO_2 targeted model	0.003886	0.062338 µg/m^3

The CO targeted model has ended up with a lower error than its NO_2 analogue. The most feasible explanation for that is the weak performance of the NO_2 sensor when compared to the CO sensor, as hinted in Section 4.2.1. However, it is not a huge drawback due to the existence of correctly performing sensors of species correlated with NO_2 and the inclusion of relevant extra variables, which make the error remain reasonable.

The same circumstances trigger the difference in architecture between both models, as can be seen in figure 16. While both have two hidden layers, the one targeted for CO has five neurons per layer and the one targeted for NO_2 has twenty. This is caused by the increase in capacity needed to compensate for the weak performance of the NO_2 sensor with other features.

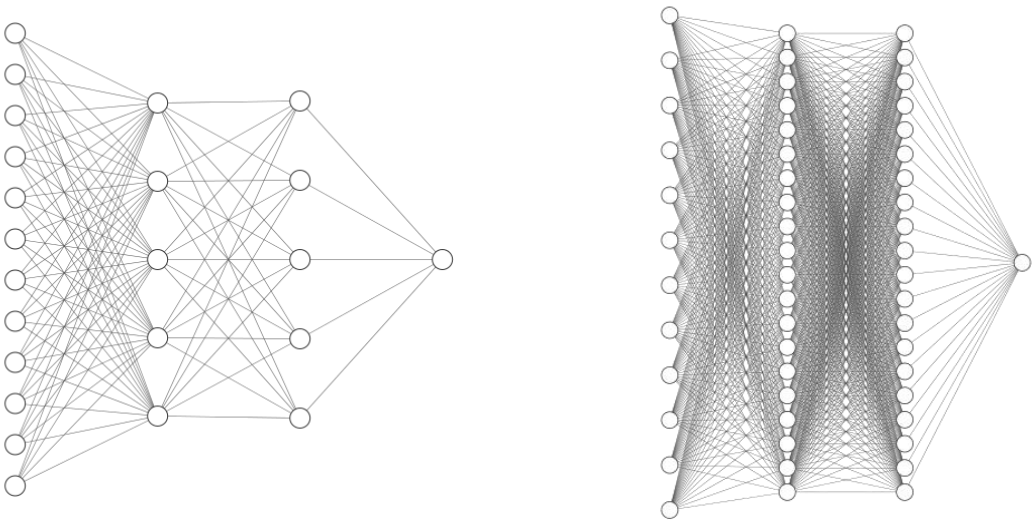

Figure 16: Diagram of the Artificial Neural Networks that form the CO targeted model (left) and the NO_2 targeted model (right)

5. Budget

The aim of the project was to build a low-cost air quality monitoring station and improve its precision through a Machine Learning algorithm.

Therefore, we only have to take into account the cost of the prototype (Raspberry Pi + Essential accessories + Sensors) and the approximate salary of a Junior Engineer along the five months that the project has lasted.

There are no software costs due to the fact that open-source software was used at all times and the dataset was compact enough to avoid the need for external servers.

With all this in mind, a table with the project's budget breakdown can be found below:

Table 5: Budget breakdown

Item	Cost
Raspberry Pi	39.99 €
MicroUSB Cable	6.59 €
32 GB MicroSD Card	12.12 €
SDS-011 Sensor	20.81 €
Grove Multichannel Gas Sensor	34.53 €
DHT-22 Sensor	6.99 €
Junior Engineer	400 hours * 15 €/hour = 6000 €
Final budget	6121.03 €

6. Environmental Impact

The result of this project definitely has a remarkable positive environmental impact. Having the possibility of obtaining accurate air quality measurements through low-cost sensors provides a broad range of possibilities.

Currently, public administrations are unable to afford placing an elevated number of measuring stations because the cost of high-end sensors is considerably high. That leads to the loss of information that could be very helpful in order to find ways to address the issue of air pollution.

Knowing that the retrieved data is reliable, a high number of low-cost sensors could be placed, for instance, on public shared bikes or buses. That would offer much more relevant information about pollution within a certain area and pave the way for future work in that field.

7. Conclusions and future development

The objectives of this project were integrating three low-cost air quality sensors into a Raspberry Pi and training an Artificial Neural Network model that improved precision in the readings made by the sensors.

To my mind, these objectives have been accomplished, but not as originally planned. The Raspberry Pi-based air quality monitoring station was built, but it failed to retrieve seven consecutive days of data when placed alongside a high-end measuring station. The cause of the failure was a programming issue.

Fortunately, there was an alternative plan in order to be able to fulfill the second objective of the project. The learning model was then built on a public dataset with hourly responses of a low-cost gas multisensor device deployed for one year along with a certified high-end monitoring station.

So, all in all, it has been proved that the imprecision of low-cost air quality sensors can be corrected through the application of models based on Artificial Neural Networks. The application of those models to the Raspberry Pi-based station is left for a future work that solves the existing issues. A future work could also explore the application of LSTM (Long Short-Term Memory) neural networks, which in theory work well with data based on time series.

As a personal remark, this project has allowed me to learn the basics of machine learning and neural networks, as well as its application to a real problem with Python, which I consider that will be a decisive addition to my formation as a Telecommunication Engineer.

Bibliography

[1] L. / Geneva, “IARC: Outdoor air pollution a leading environmental cause of cancer deaths,” 2013.

[2] O. US EPA, “Criteria Air Pollutants.”

[3] “WHO Air quality guidelines for particulate matter, ozone, nitrogen dioxide and sulfur dioxide.”

[4] J. J. Li, B. Faltings, O. Saukh, D. Hasenfratz, and J. Beutel, “Sensing the Air We Breathe -- the OpenSense Zurich Dataset.”

[5] O. A. Postolache, J. M. D. Pereira, and P. M. B. S. Girao, “Smart Sensors Network for Air Quality Monitoring Applications,” *IEEE Trans. Instrum. Meas.*, vol. 58, no. 9, pp. 3253–3262, Sep. 2009.

[6] T. Alhmiedat, “A Survey on Environmental Monitoring Systems using Wireless Sensor Networks,” *J. Networks*, vol. 10, 2016.

[7] S. De Vito, M. Piga, L. Martinotto, and G. Di Francia, “CO, NO2 and NOx urban pollution monitoring with on-field calibrated electronic nose by automatic bayesian regularization,” *Sensors Actuators B Chem.*, vol. 143, no. 1, pp. 182–191, Dec. 2009.

[8] 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,” in *Sensors and Actuators, B: Chemical*, 2015, vol. 215, pp. 249–257.

[9] and G. X. Gaganjot Kaur Kang, Jerry Zeyu Gao, Sen Chiao, Shengqiang Lu, “Air Quality Prediction: Big Data and Machine Learning Approaches.”

[10] A. L. Samuel, “Some studies in Machine Learning using the game of checkers.”

[11] A. Ng, “Machine Learning | Coursera.” [Online]. Available: https://www.coursera.org/learn/machine-learning/.

[12] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. 1962.

[13] D. P. Kingma and J. L. Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION.”

[14] “The SDS011 Air Quality Sensor experiment.” [Online]. Available: http://aqicn.org/sensor/sds011/es/. [Accessed: 21-Feb-2018].

[15] Seeed Studio, “Grove - Multichannel Gas Sensor.” [Online]. Available: http://wiki.seeedstudio.com/Grove-Multichannel_Gas_Sensor/. [Accessed: 01-Jul-2018].

[16] T. DiCola, “Adafruit Python DHT,” 2015. [Online]. Available: https://github.com/adafruit/Adafruit_Python_DHT.

[17] P. Van Haastrecht, “C program for the Raspberry PI and information to interact with a Grove MultiChannel gas sensor with a MiCS-6814 gas sensor,” 2017. [Online]. Available: https://github.com/paulvha/multichannel-gas.

[18] M. Luetzelberger, “Monitoring air quality with a Nova PM2.5/PM10 Sensor and Python | Yet another Raspberry Blog … | Page 2.” [Online]. Available: https://raspberryblog.de/?p=2184&page=2. [Accessed: 18-Jun-2018].

[19] D. Dheeru and E. Karra Taniskidou, “{UCI} Machine Learning Repository.” 2017.

[20] W. McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 51–56.

[21] T. E. Oliphant, “Guide to NumPy,” 2006.

[22] F. Chollet and others, “Keras.” 2015.

[23] R. Garreta and G. Moncecchi, “Learning scikit-learn: Machine Learning in Python,” *Physiol. Res.*, vol. 12, pp. 461–468, 2013.

[24] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.

[25] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P. Tak, and P. Tang, “ON LARGE-BATCH TRAINING FOR DEEP LEARNING: GENERALIZATION GAP AND SHARP MINIMA.”

Glossary

Acronym	Name
ANN	Artificial Neural Network
CSIC	Centro Superior de Investigaciones Científicas
MSE	Mean Square Error
RMSE	Root Mean Square Error
ReLU	Rectified Linear Unit
I^2C	Inter-integrated Circuit
GPIO	General Purpose Input-Output
PM	Particulate Matter
NMHC	Non-Methane HydroCarbons
CO	Carbon monoxide
NO_2	Nitrogen dioxide
NO_x	Nitrogen oxides (in general)
O_3	Ozone
LSTM	Long Short-Term Memory