A Deep Learning Framework for Air Pollution Forecasting and Interpolation

BY

MARCO MIGLIONICO
B.S., Politecnico di Milano, Milan, Italy, 2017

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2019

Chicago, Illinois

Defense Committee:

Ouri Wolfson, Chair and Advisor
Jane Lin, Civil and Materials Engineering
Matteo Matteucci, Politecnico di Milano

ACKNOWLEDGMENTS

My sincere gratitude goes first of all to Prof. Ouri Wolfson and Prof. Jane Lin who encouraged me and helped me during this work we made, dedicating to me lot of their precious time. Thanks to them I started my journey into Deep Learning, a field that one year ago was completely new for me, but that now has become one of my biggest passion. Also, I want to thank Prof. Matteo Matteucci that accepted to be the Politecnico advisor for this thesis.

I want to thank my mother, my father and my sister that everyday, despite the distance and the time difference, have always found the time to send me a message, to call me (maybe too many times, right mom?) and to encourage me to give my best during the easy and difficult moments. If I am here right now, it is only thanks to them and I will be grateful for that for the rest of my life.

I want to thank my best friends Paride and Simone, because no matter how far away we are, our friendship is still the same or maybe even stronger. Thank you for being not just simple friends, but also my main source of inspiration,discussion and improvement. We share the same ambitions and the same dreams and I am sure we will reach them together.

Finally, I want to express my gratitude to the people that made my Chicago experience unforgettable. Special mention to my favourite girls Miranda, Manuela, Isabel and Georgina that have always found the time to listen to my problems, to understand me and to make me feel loved in every single day of this experience.

MM

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

FIGURE **PAGE**

LIST OF ABBREVIATIONS

UIC	University of Illinois at Chicago
NO_2	Nitrogen dioxide
$PM_{2.5}$	Fine particulate matter 2.5
PM_{10}	Fine particulate matter 10
CMAQ	Community Multiscale AirQuality
WRF	Weather Research and Forecasting
MLR	Multiple Linear Regression
ARMA	Autoregressive Moving Average
ARMA	Autoregressive Moving Average
SVR	Support Vector Regression
ANN	Artificial Neural Network
BPNN	Back Propagation Neural Network
MLP	Multi-Layer Perceptron
RBF NN	Radial Basis Function Neural Network
NFNN	Neuro-Fuzzy Neural Network
GRNN	General Regression Neural Network
RNN	Recurrent Neural Network

LIST OF ABBREVIATIONS (continued)

LSTM	Long Short Term Memory Neural Network
LSTME	Long Short Term Memory Neural Network Extended
BDLSTM	Bidirectional Long Short Term Memory Neural Network
CEC	Constant Error Carrousel
DRNN	Deep Recurrent Neural Network
GDBT	Gradient Boosting Decision Tree
DFNN	Deep Feed Forward Neural Network
TDNN	Time Delay Neural Network
STDL	Spatio Temporal Deep Learning Model
ARMA	Autoregressive Moving Average
SBU-LSTM	Undirectional and Bidirectional Long Short Term Memory Neural Network
SB-LSTM	Stacked Bidirectional Long Short Term Memory Neural Network
CNN	Convolutional Neural Network
IDW	Inverse Distance Weighting
LUR	Land Use Regression

LIST OF ABBREVIATIONS (continued)

EPA	Environmental Protection Agency
AQS	Air Quality System
AQI	Air Quality Index
RMSE	Root Mean Square Error
BPTT	Backpropagation through time
RF	Random Forest

SUMMARY

Air pollution has been identified as the world's largest single environmental health risks by the World Health Organization. Protecting humans from the damage caused by air pollution, is one of the major issues for the global community. To accomplish this task, real time air-quality information is necessary, such as the concentration of $PM_{2.5}$, PM10 and NO2.5.

In this Thesis we will address this problem by creating a new framework capable of predicting and interpolating the $PM_{2.5}$ concentration.

For this purpose we will create a deep learning model, that combine Recurrent and Feed Forward Neural Network. In particular, we will use an LSTM in with a Bidirectional Training that process the input sequence in both chronological and anti-chronological order using two separate hidden layers. In this way, the network can take advantage of the 2 direction to better exploit the context and learn new patterns.

We will then use the predicted value at each ground monitoring station as input for another Neural Network model trained using a technique called "Self-Training" for semi-supervised learning and we will create interactive maps of the City of Chicago on a 1km X 1km square grids with an hourly frequency.

Finally, we will compare our model with different baselines and we will propose some possible extension of the proposed methodology.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Air pollution causes millions of deaths every year and has been identified as the world's largest single environmental health risks by the World Health Organization. Protecting humans from the damage caused by air pollution, is one of the major issues for the global community. To accomplish this task, real time air-quality information is necessary, such as the concentration of NO_2, $PM_{2.5}$ and PM_{10}.

Air quality monitoring stations can measure the concentrations of pollutants at their locations. The retrieved data highlight that air quality spatial variation is nonlinear and it is caused by several factors such as traffic conditions, meteorology and surface land use. Nevertheless, the extent of spatial coverage by these stations is limited, due to the high cost to build, operate, and maintain.

To overcome this problem, the main objective will be to forecast air quality concentrations throughout a city with high precision, using the air quality data measured by existing ground monitoring stations and several data sources that are available or will be made available in the future in a city. This research will concentrate on fine particles with a diameter that is in general less than 2.5 micrometers or smaller ($PM_{2.5}$), since recent studies have proved that this pollutant lead to more than 800,000 premature deaths per year. For this reason, $PM_{2.5}$ has

been ranked as the 13th leading cause of mortality worldwide.[1]

This pollutant has been recognized as the main cause of a variety of health problems and have been associated with deaths from heart or lung disease. Several studies show that short-term exposures (more than 24 hours) and long-term exposures (one or several years) have a strong correlation to these effects. Some category of people that suffer of heart or lung diseases, but also older adults when exposed to particle pollution, are more likely to visit hospitals, emergency rooms, or in some serious cases, even die.[2]

1.2 Background

1.2.1 Existing air pollutants forecasting methods

In the past years, lot of researches have focused on finding new approach and improve the existent methods to predict air pollutant concentrations. As a rule, the techniques for forecasting air pollutant concentrations can be separated in two main categories [3]:

- **Deterministic methods:** these methods embrace statistical methods and meteorological principles to represent the dispersion, emanation, diffusion, transformation, and elimination processes of pollutants based on atmospheric physics and chemical reactions. These techniques can be considered model-based methods since their architecture are predefined based on certain theoretical assumptions, and it is possible to calculate through precise priori knowledge their parameters.

  Example of these methods are Community Multiscale AirQuality (CMAQ) model [4], Nested Air Quality Prediction Modeling System [5] and WRF Chem model [6].

- **Statistical methods:** these methods do not use complex theoretical techniques but employ statistical based techniques to forecast air quality.

  The most broadly used strategies include multiple linear regression(MLR) [7], the autoregressive moving average(ARMA) [8],the support vector regression (SVR) [9] and the artificial neural network (ANN) [10]. In particular, ANN methods have shown to be self-adaptive and robust for the time series prediction task, thanks to its ability to perform nonlinear mapping.

Recently, several efforts have been done in order to improve the prediction capability of ANN techniques[3]. Typical examples include back propagation neural network (BPNN) [11], multilayer perceptron (MLP) [12] , radial basis function neural network (RBF NN) [13], neuro-fuzzy neural network (NFNN) [14], general regression neural network (GRNN) [15], and recurrent neural network (RNN) [16].

1.2.2 Existing air pollutants interpolation methods

Regarding more specifically the interpolation part (air quality inference), different approaches have been tested during the years. It is possible to divide them into four main categories:

- **The standard interpolation techniques** (Inverse Distance Weighted Interpolation, Kriging, Spatial Averaging, Gaussian Interpolation and Land Use Regression)
- **The Dispersion Models** (Gaussian Plume, Lagrangian and Eulerian Dispersion model)

- **The Machine Learning Techniques for air quality inference**(Support Vector Regressor, Decision Tree, K-Nearest Neighbor, Nave Bayes and Random Forest)
- **The Deep Learning air quality inference techniques** (e.g. U-Air, ADAIN)

From now on we will use the term interpolation and air quality inference interchangeably to indicate the process of deriving the pollution concentration of unknown locations, using both standard and deep learning techniques.

1.2.3 Related works

Long Short Term Memory Neural Networks have been effectively employed in several studies involving time series forecasting, such as traffic flow prediction [17], wind power prediction [18], human trajectory prediction [19].

LSTM was proposed for the first time by Hochreiter and Schmidhuber [20].In their work they address the exploding gradient problem of the recurrent Neural Network by introducing a new method called Long Short-Term Memory(LSTM). In this gradient-based method a constant error flow within its constant error carrousel CEC is guaranteed by the internal architecture of each memory cell, provided that the error flow trying to leak out of memory cells is cut off by truncated backpropagation. The main idea of this kind of RNN is the presence of a memory cell which can maintain its state over time.

Recently, Sak et al. (2016) [21] applied LSTM for pollution risk forecasting, but they didn't predict real-value concentrations of air pollutants since they only analyzed the pollution risk ranking. Besides this method do not consider the spatial correlations between monitoring stations.

Fan et al. (2017) [22] proposed a spatiotemporal prediction method based on deep recurrent neural network (DRNN) and missing value processing algorithms. Three different missing values fixing algorithms were implemented using missing interval and missing tag to create a representation of the time series patterns. These methods were furtherly incorporated into deep neural network framework composed by different LSTM layers and fully connected layers. In their work to train and test their model they employed air quality and meteorological datasets form the Jingjinji area in China. The baseline models used against the DRNN were gradient boosting decision trees (GBDT) and Deep feed forward neural networks (DFNN). The results show that the proposed DRNN framework outperforms both the baseline in term of RMSE.

Li et al. (2017) [3]in their work, applied an extended version of LSTM called Extended LSTM (LSTME). It consisted in a model that intrinsically considers spatio-temporal correlations in order to make air pollution forecasting. The layers of the Long short-term memory (LSTM) network were employed to derive intrinsic useful features in an automatic way from historical and auxiliary data, like time stamp and meteorological information. The data were merged together to enhance the performance of the proposed methodology. Hourly $PM_{2.5}$ (fine particles with a diameter that is in general less than 2.5 micrometers or smaller) concentration data retrieved using twelve ground air quality monitoring stations in the city of Beijing from January 2014 to the end of May 2016 were employed to evaluate the performance of the proposed LSTME methodology. Several techniques were used to perform the experiments. These techniques include the time delay neural network model (TDNN), the spatiotemporal deep learning model (STDL), the traditional LSTM, the autoregressive moving average model (ARMA) and

the support vector regression model(SVR). The final results showed that the LSTM extended methodology outperformed the statistics-based techniques.

Reddy et al. (2017) [23] investigated the use of the LSTM recurrent neural network (RNN) as a framework for forecasting in the future, based on time series data of pollution and meteorological information in Beijing. Due to the sequence dependencies associated with large-scale and longer time series datasets, RNNs, and in particular LSTM models, demonstrated to be well-suited. The results demonstrated that the LSTM model produced equivalent accuracy when forecasting future timesteps compared to the baseline support vector regression for a single timestep.

Cui et al. (2018) [24] proposed a new model based on deep stacked unidirectional and bidirectional LSTM neural network(SBU-LSTM) model to forecast network-wide traffic speed, which considers forward and backward dependencies . The power of BI-LSTM is used in combination with historical data to capture the temporal dependencies and the variablity in the spatial features. The model was used to predict traffic speed on urban networks and freeway. The proposed SBU-LSTM, when compared with other baseline and state of the arts models shows better performance in term of predictions for the entire network.

Fan et al. (2017) [22] create a Deep Recurrent Neural Network (DRNN), based on LSTM, that predict future air pollution concentrations in the area of Jingjinji (China) and at the same time tackles the problem of missing values by implementing 3 different algorithms to fix the missing values. They compared the performance of the model with Deep Forward Neural Networks (MLP) and Gradient Boosting Decision Trees, showing that their model outperforms the base-

line. Besides, they applied inverse distance weight interpolation to create air pollution maps of the study area.

Huang et al. (2018) [25] developed a Deep Neural Network model that combines CNN and LSTM to predict the next hour concentration of $PM_{2.5}$ using historical pollution data and meteorological data like cumulated wind speed and cumulated hours of rain. They created a framework that use the power of CNN to extract feature importance and the ability of LSTM to solve long-term dependencies. In this work they use the historical data of the previous 24 hours to predict the next hour concentration and they compare their results with different baselines like SVM, Random Forest, Decision Tree, Multi-Layer Perceptron, CNN and LSTM.

Lin et al.(2018) [26] presented a new approach to forecast PM2.5 concentration. This new approach is based on the combination between CNN and RNN.

Combining these two approaches, they created the GC-DCRNN based on diffusion convolution. The main innovations of this model were the use of a graph based structure, that was built using the similarity between different locations. This approach was possible due to the high number of stations in different locations present in this work. The similarity measure used to build the graph was based on the geographical features of each location within different buffers. The experiment was set in Beijing and Los Angeles and the results were compared with other baselines including the standard Linear Regression(LR), the Vector Autoregression(VAR) and the Gradient Boosting Machines(GBM).

Regarding more specifically interpolation part we will start with the standard interpolation techniques:

Wong et al(2004) [27] selected 4 different interpolation methods to interpolate O3 and PM10,. All the methods we based on the same mathematical formulation and on weighted average. The methods applied were Spatial Averaging, Nearest Neighbor, Inverse Distance Weighting (IDW) and Universal Kriging. The results indicated that Kriging performed slightly better than the other 3 methods.

Mercer et al. (2011) [28] compared Universal Kriging and Land Use Regression(LUR) to interpolate the values of gaseous oxides of nitrogen (NOx) in the city of Los Angeles. The results show that Universal Kriging perform well or better than the analogous LUR models, evaluated in terms of CV R squared.

Keler et al. (2014) [29] used inverse distance weighting interpolation to generate surfaces of PM2.5 concentration . The interpolated results were based on the daily PM2.5 values. The goal of this study was the possibility of detecting high PM2.5 hot-spots concentrations,that might be dangerous for sensistive people.

Contreras et al. (2018) [30] after predicting the air pollutants values using a Land Use Regression model, interpolated the values using a new technique based on Inverse Distance Weighting that takes into account also the wind direction, and compare the generated maps with the one obtained from standard methods like IDW. The results show how the new methodology can capture the pollution dispersion in a more accurate way with respect to the IDW.

For the dispersion models:

Arystanbekova et al. (2004) [31] applied Gaussian Plume to estimate local pollution level on a grid map. In their work they read the inputs file form a GIS file and produce as output some maps of pollutions levels.

Mok et al. (2008) [32] proposed a modified version of the Gaussian Plume to simulate the dispersion of the air pollutants under non-homogeneous wind conditions. The setting of the experiment was the city of Lisbon and the results were evaluated with the measured concentration of sulphur dioxide showing good skills in modeling the dispersion and transport of air pollutants under complex wind conditions. The results show a significant improvement compared to the traditional Gaussian Plume.

Beelen et al. (2010) [33] compared the performance of Land Use Regression (LUR) and a dispersion model called URBIS in a Dutch urban area. URBIS used Gaussian Plume to calculate the pollution levels near industrial sources and another model called CAR (Calculation of Air pollution from Road traffic) to calculate the pollution level near roads. Some 1km x 1km grid maps was created to monitor and model the dispersion of air pollutants. The URBIS model performed better than LUR regression in term of R squared. On the calculation of the annual average the two models showed very similar performance.

Van der Swaluw et al. (2017) [34] presented high resolution maps on a 1km x 1km grids to model air quality in Netherlands. They applied the OPS model, a combination of Lagrangian Trajectory model for long range transport and Gaussian Plume to calculate the concentration maps. The model was compared with another dispersion model called Eulerian Model with the same resolution. The results show similar performance between the two dispersion models,

even if the OPS model shows faster computational time.

For the deep learning approaches instead:

Zheng et al. (2013) [35] proposed a new methodology (U-Air) based on a co-training framework to infer air quality information in a city using the data coming from a small number of monitoring stations and several datasets like meteorological data, traffic information and point of interest. The setting of the experiments was the city of Beijing and the results show that the proposed co-training semi-supervised model based on Artificial Neural Networks (ANN) and Conditional Random Field (CRF) outperform other techniques including Linear Interpolation, Gaussian Interpolation and Decision Tree.

Chen et al. (2016) [36] infer air quality concentration applying an ensemble semi-supervised technique. They generate multiple classifiers and they trained them using a co-training framework. The dataset used include road-network features, traffic related features, check-in features, POIs and other monitoring stations data. They first train multiple classifiers using bootstrapping from the original dataset and assign a confidence label to unlabeled data. The examples with the highest confidence are then added to the training data and the training process is repeated. This process is repeated until all the examples have been classified. This technique outperforms different methodology like Gaussian Interpolation, K- Nearest Neighbor, Nave Bayes and Random Forest.

Finally, Cheng et al. (2018) [37] proposed a generic neural network approach called ADAIN, based on the attention mechanism for urban air quality inference. In this work, they used both recurrent and feed forward neural network to capture deep feature interactions. They leverage

the data from the ground monitoring stations and other urban data sources including meteorological data, road networks and POIs. The model outperformed other standard interpolation and machine learning techniques like K-Nearest Neighbor, Co-Training, Gaussian Interpolation, Linear Interpolation and Support Vector Regressor.

1.2.4 Useful definitions

EPA [38]: "The United States Environmental Protection Agency (EPA or sometimes U.S. EPA) is an agency of the federal government of the United States which was created for the purpose of protecting human health and the environment. The EPA AirData website provides access to air quality data collected at outdoor monitors across the United States, Puerto Rico, and the U. S. Virgin Islands. The data comes primarily from the AQS (Air Quality System) database. AirData assists a wide range of people, from the concerned citizen who wants to know how many unhealthy air quality days there were in his county last year to air quality analysts in the regulatory, academic, and health research communities who need raw data".

Air Quality Index [38]: "The AQI is an index for reporting daily air quality. It tells how clean or polluted the air is, and what associated health effects might be a concern for people. The AQI focuses on health effects people may experience within a few hours or days after breathing polluted air. The function used to convert from air pollutant concentration to AQI varies by pollutants, and is different in different countries. Air quality index values are divided into ranges, and each range is assigned a descriptor and a color code". The United States Environmental Protection Agency issued a standard table, that is presented in Figure 1

Air Quality Index (AQI) Values	Levels of Health Concern	Colors
When the AQI is in this range:	*..air quality conditions are:*	*...as symbolized by this color:*
0 to 50	Good	Green
51 to 100	Moderate	Yellow
101 to 150	Unhealthy for Sensitive Groups	Orange
151 to 200	Unhealthy	Red
201 to 300	Very Unhealthy	Purple
301 to 500	Hazardous	Maroon

Figure 1: AQI Index Values, Levels and Colors

Feature [39]: "In machine learning and pattern recognition, a feature is an individual measurable property or characteristic of a phenomenon being observed. Choosing informative, discriminating and independent features is a crucial step for effective algorithms in pattern recognition, classification and regression. Features are usually numeric, but structural features such as strings and graphs are used in syntactic pattern recognition. The concept of feature is related to that of explanatory variable used in statistical techniques such as linear regression".

1.3 Research Objectives

In this study we will focus on a particular class of RNN, the Long Short-Time Memory Neural Networks to predict pollution concentrations in the city of Chicago. In fact, RNN can

handle arbitrary sequences of inputs and they also guarantee the capacity to learn temporal sequences. These characteristics seems to be particularly suited for handling the spatio-temporal evolution of air pollutants distribution [40].

In addition, this study will:

1. incorporate various data sources, including the outdoor air quality data retrieved from EPA(US Environmental Protection Agency) air monitoring stations from state, local and tribal monitoring agencies across the United States. The other data source incorporated will include historical meteorological data such as Wind direction, Wind Speed, Temperature, Humidity and Pressure.
2. compare the proposed neural network approach with other techniques including the Multi-Layer Perceptron, the General Regression Neural Network and the standard LSTM. The results will be compared in term of RMSE
3. apply inverse distance weighting interpolation to obtain interactive pollution concentration maps of the city of Chicago with a temporal resolution of one hour, on a grid composed of square cells of 1 km x 1 km.

1.3.1 Scientific contributions

In recent years deep learning methods have received lot of attention. Nevertheless, the potential of deep learning methods in forecasting the air pollutants concentrations has not been fully exploited yet for what concerns the predictive power of spatial-temporal data, the spatial scale of the prediction area and the depth of the model architecture.

Theoretically speaking, a complex NN can represent any linear or nonlinear function. The inherent nonlinear structure of NNs is useful for managing complex relations in problems of diverse disciplines. NN models, present some important characteristics like recognizing complex patterns or relationships in historical observations, learning from past data and forecasting future values using the relationships found [41].

In this study we will propose a deep stacked bidirectional LSTM (SB-LSTM) neural network architecture, which considers both forward and backward dependencies in time series data, to predict pollution concentration at the ground monitoring stations.

This model was firstly proposed in [24] for network wide traffic speed prediction . In our approach we will apply the stacked bidirectional and unidirectional LSTM to predict the air pollution concentration in the city of Chicago. The topology of the network will have some significant difference and will be adapted to our problem.

A special subclass of recurrent neural network (RNN) architecture is the LSTM. One of the main problem of RNN is that they suffer the vanishing gradient point in handling time series. By including memory cells in the hidden layer of RNN, LSTM cells can address this issue. Thanks to its ability to learn from the past data and determine the relationship and effects among time series, LSTM performs well in long time horizon prediction with respect to machine learning techniques [42].

Most of the newly proposed LSTM-based prediction models have relatively shallow structures with only one hidden layer to deal with time series data [40; 43]. Existing studies [44] have demonstrated that deep LSTM with more tahn one hidden layer can reach higher levels of

representations of sequence data.

The main reason for stacking multiple LSTM is to allow for greater model complexity. In fact, as in feedforward neural network stacking multiple layers is useful to create a hierarchical feature representation of the input data, the same principle can be applied for stacked LSTM. At every time step an LSTM, receives the recurrent input. If the input is already the result from an LSTM layer (or a feedforward layer) then the current LSTM can create a more complex feature representation of the current input. Normally, the dataset fed to an LSTM model is chronologically arranged, with the result that the information in the LSTMs is passed in a positive direction from the time step t -1 to the time step t along the chain-like structure. Thus, the LSTM structure only makes use of the forward dependencies [24]. It is highly possible that useful information is filtered out or not efficiently passed through the chain-like gated structure. Therefore, it may be informative to consider backward dependencies, which pass information in a negative direction, into consideration.

The idea of Bidirectional LSTM(BDLSTM) comes from bidirectional RNN [45], which use 2 different hidden layers to process the sequence of data in both directions(backward and forward). BDLSTMs connect the two hidden layers to the same output layer. Bidirectional networks have proved better performance than the unidirectional counterpart in several fields, like speech recognition [44].

Based on our knowledge and on the review of the literature, there are no studies about the air pollution analysis that make use of the backward dependency. For this reason, in our model we introduce also the Bidirectional LSTM (BDLSTM), a special type of LSTM with the ability to

deal with both forward and backward dependencies.

The impact of the neural network hyperparameters, including the number of neurons, layers, time lags (length of the time series input), the batch size and the dimension of weight matrices in LSTM/BDLSTM layers will be further analyzed in the following sections.

After predicting the air pollution concentration at each station, we will create for each hour an interactive visualization map of the air pollution with a spatial resolution of 1km x 1km, using an inverse distance weighting interpolation algorithm(IDW). The choice of this interpolation method is based on the comparison between Inverse Distance Interpolation and other interpolation techniques. Researchers discovered that the Inverse Distance Weighting index is the closest to the medium sampling points pollution index and to the medium pollution prediction index. [37]

These maps can be very useful to accurately identify high risk region based for sensible people and calculate different risk level areas in a city.

The contributions of this study can be summarized in the following way:

1. A deep neural network composed of bidirectional LSTM to capture the long-term spatio-temporal dependency in the concentration of air pollutants, and to forecast the air pollutant concentration over a very short time period (from 1 to 4 hours) is presented;
2. our framework make use of both forward dependencies and backward dependencies;
3. auxiliary data, like weather conditions, are integrated into our deep learning framework

4. Our framework will be capable to interpolate the pollution concentrations over the whole study region using an inverse distance weighted interpolation technique that will produce air pollution map with a grid cell resolution of 1km x 1 km.

CHAPTER 2

DATA

2.1 Data used in the study

As previously stated, different data sources have been employed in this study.
In particular, we can categorize them as follows:

2.1.1 $PM_{2.5}$ Data

The hourly ground measurements of $PM_{2.5}$ from 4 monitoring stations in the urban area of Chicago collected from September 2013 to September 2017 were retrieved from the US Environmental Protection Agencys (EPA) Air Quality System. The hourly $PM_{2.5}$ concentrations at each monitoring site were used and expressed as micrograms per cubic meter (i.e., $\mu g/m^3$). The dataset contained 35520 records for each station. Besides, several missing values were present. We decided to impute them using a linear interpolation technique.
The following plots shows the $PM2.5_{2.5}$ hourly concentration for each monitoring station during the whole period considered:

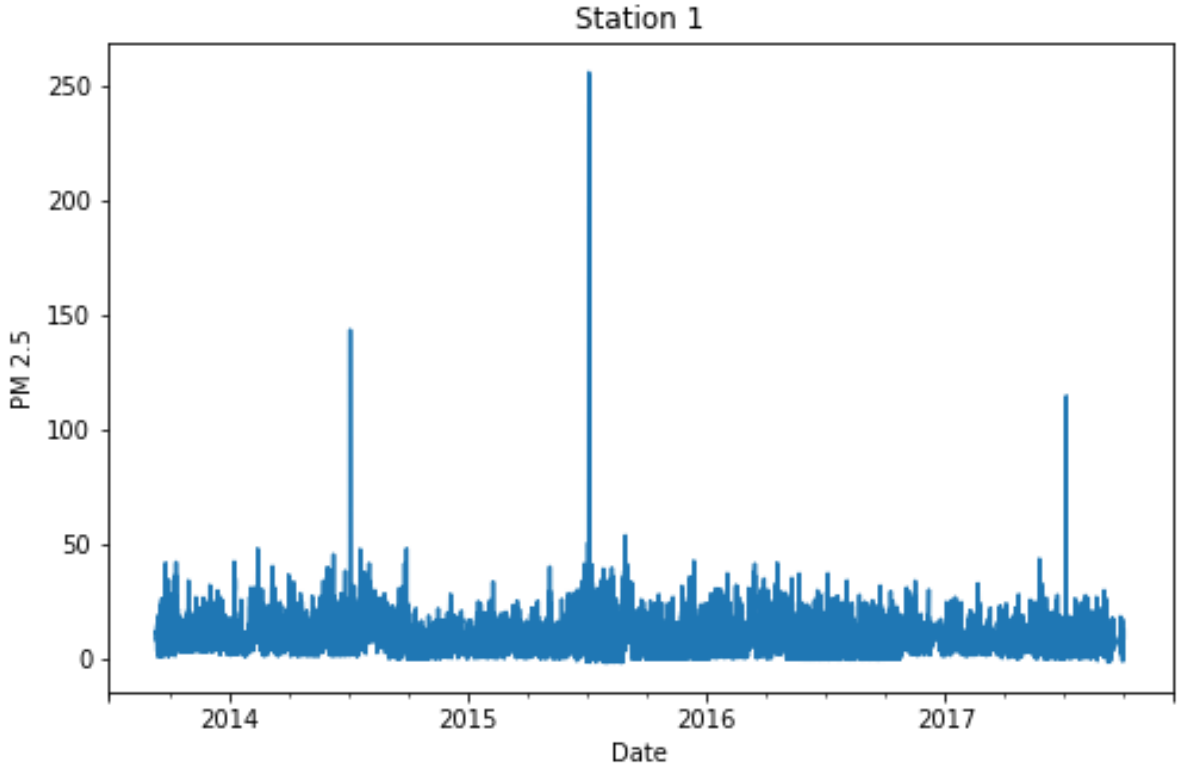

Figure 2: Station 1 Hourly Concentration

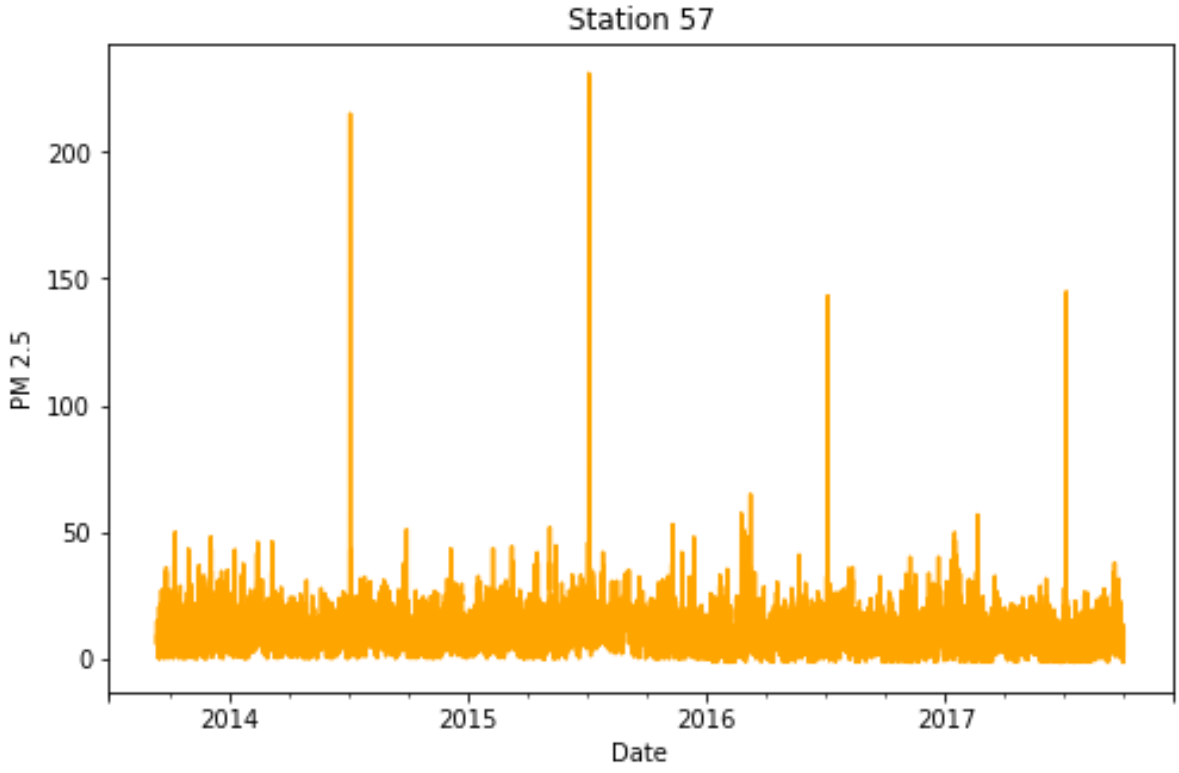

Figure 3: Station 57 Hourly Concentration

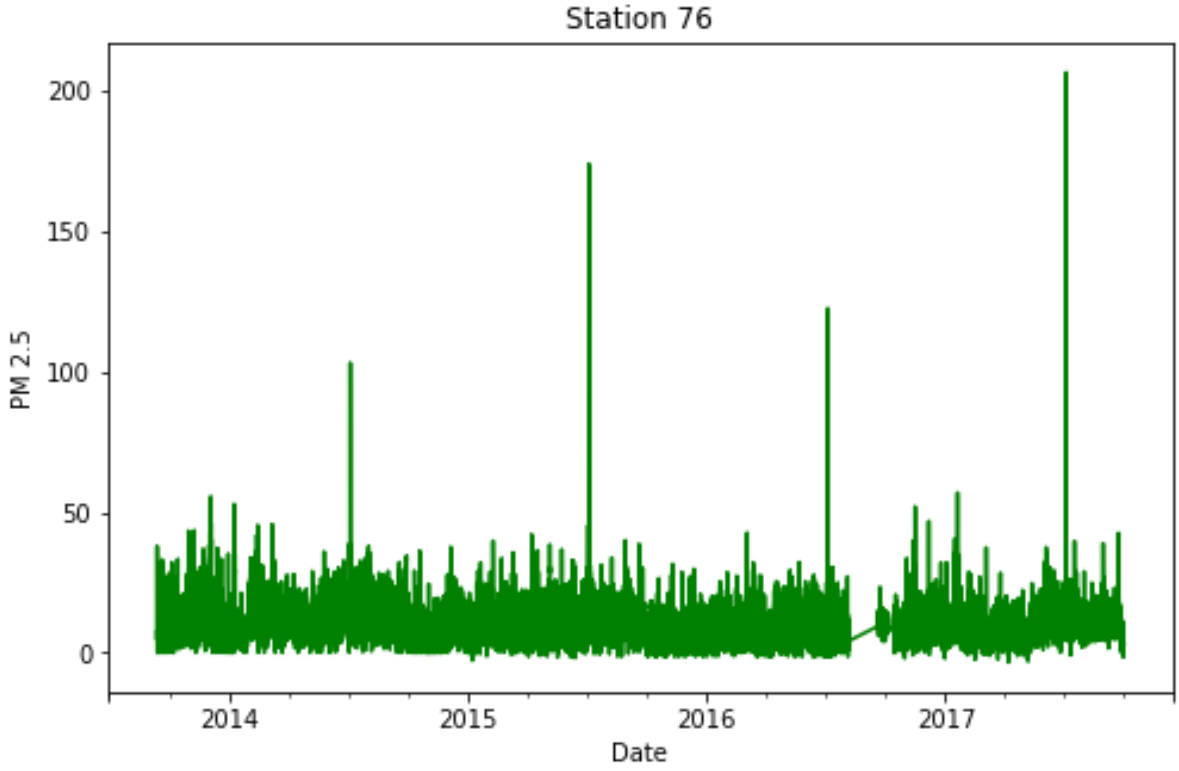

Figure 4: Station 76 Hourly Concentration

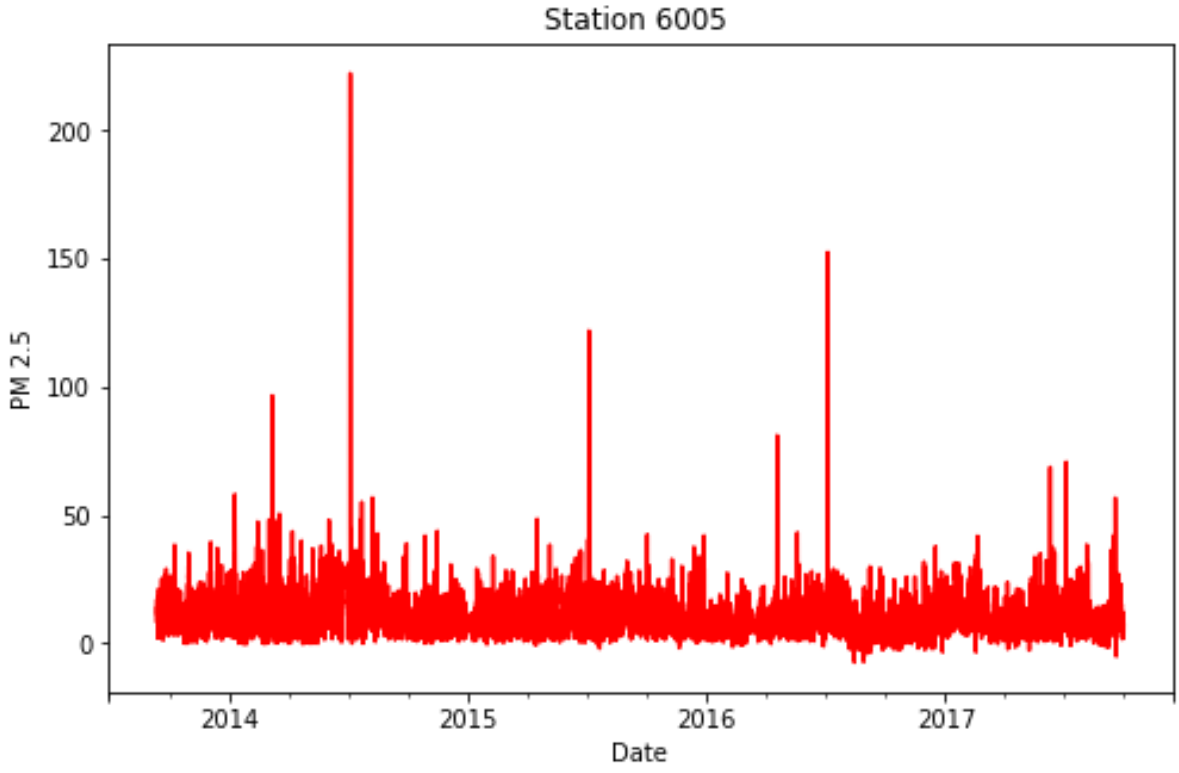

Figure 5: Station 6005 Hourly Concentration

The following figure show the Chicago area considered for the experiment with the 4 EPA monitoring stations:

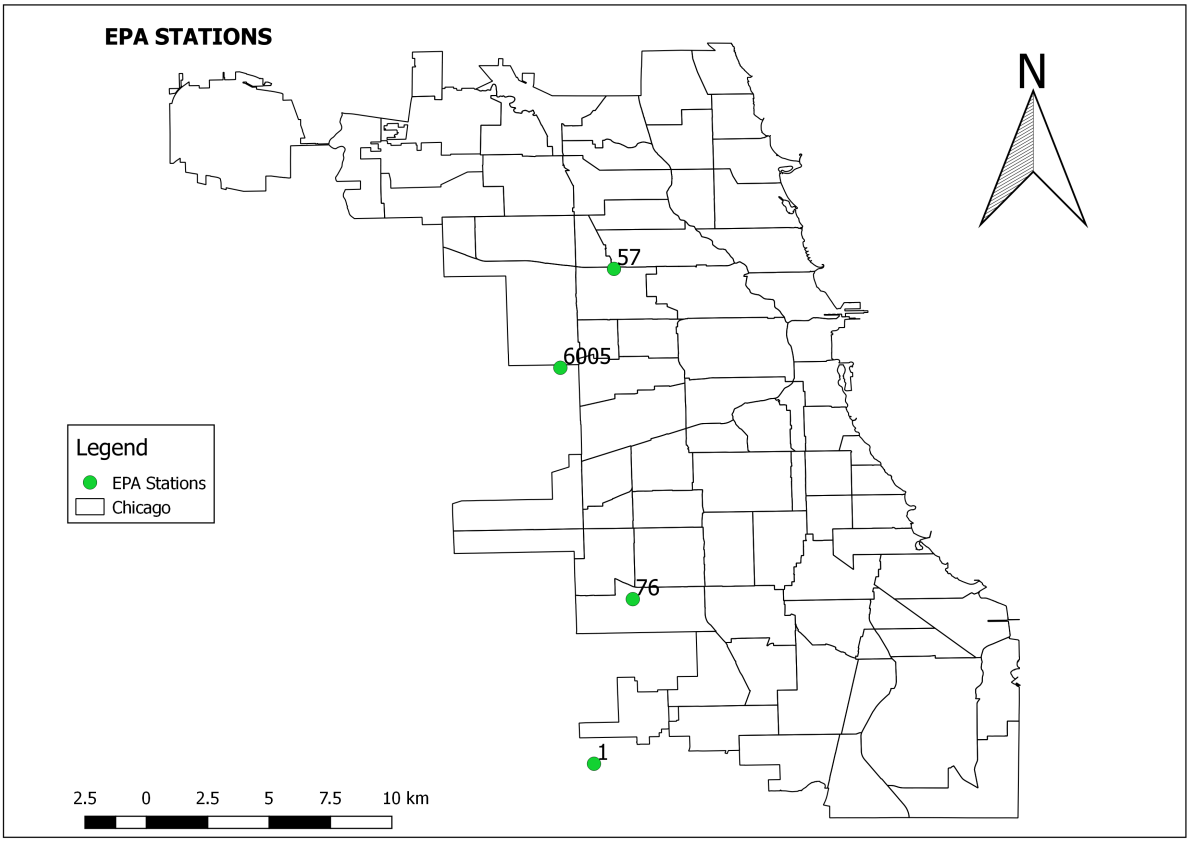

Figure 6: EPA Stations in Chicago

2.1.2 Meteorological Data

Meteorological data for the same period (September 2013 to September 2017) were obtained using Weather API on the OpenWeatherMap website. The dataset contained the hourly values of Temperature, Humidity, Pressure, Wind direction, Wind Speed and Weather Description. The data were preprocessed, in order to select the appropriate time period and to reduce the dataset only for the city of Chicago. The dataset contained some missing values that were imputed using a linear interpolation technique. One hot encoding was applied to the Weather Description feature to convert the categorical values into numerical ones. The following plot show the distribution of the different meteorological features across the time period considered, that consist of 35520 timestamps.

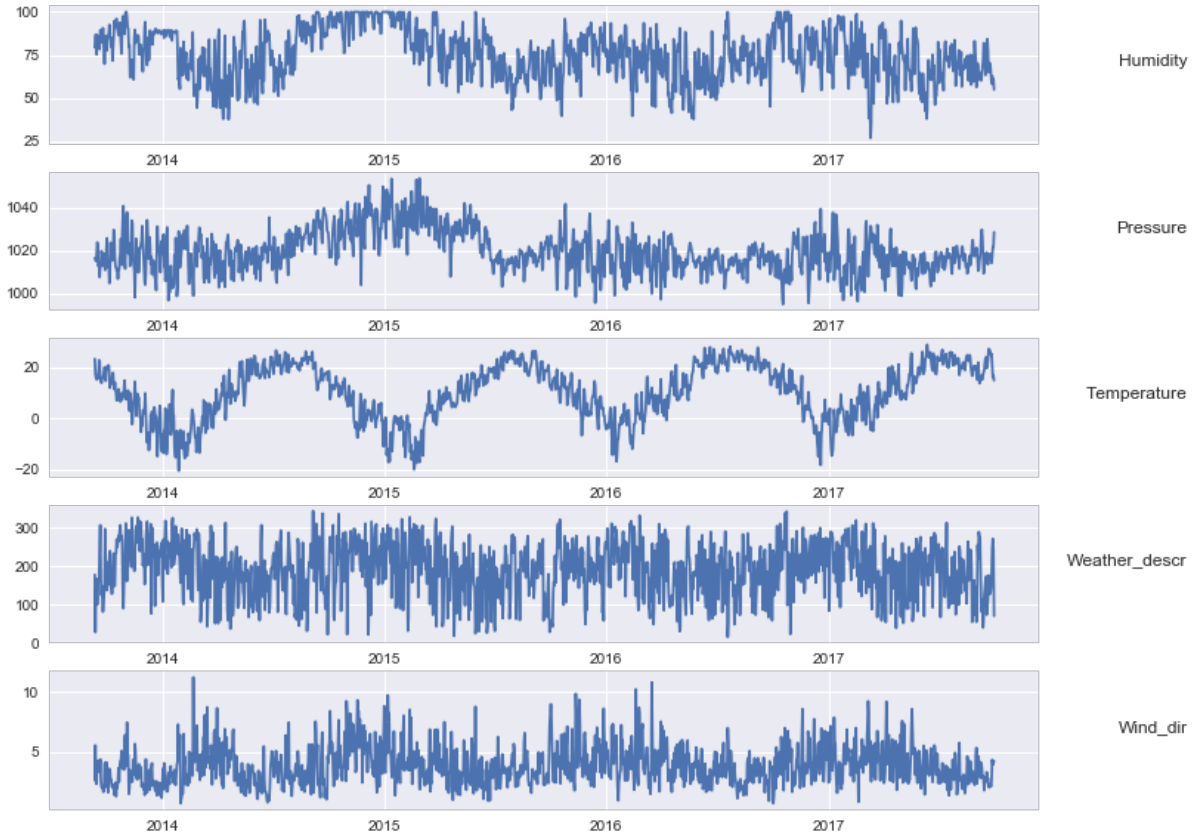

Figure 7: Meteorological Data years 2013-2017

The following table shows the statistics for each feature considered:

TABLE I: Features Statistics

Feature	**Unit**	**Range**	**Mean**	**#NA**
PM2.5	ugm^{-3}	[-7.4, 255.5]	9.86	6367
Temperature	$°C$	[-24.26, 34.56]	10.29	0
Humidity	%	[9, 100]	75.53	184
Pressure	hpa	[941, 1077	1019.58	0
Wind Direction	°	[0, 360]	191.79	0
Wind Speed	m/s	[0, 25]	3.86	0

Finally, we can see the correlation between the different features considered with the following heatmap (Figure 10:

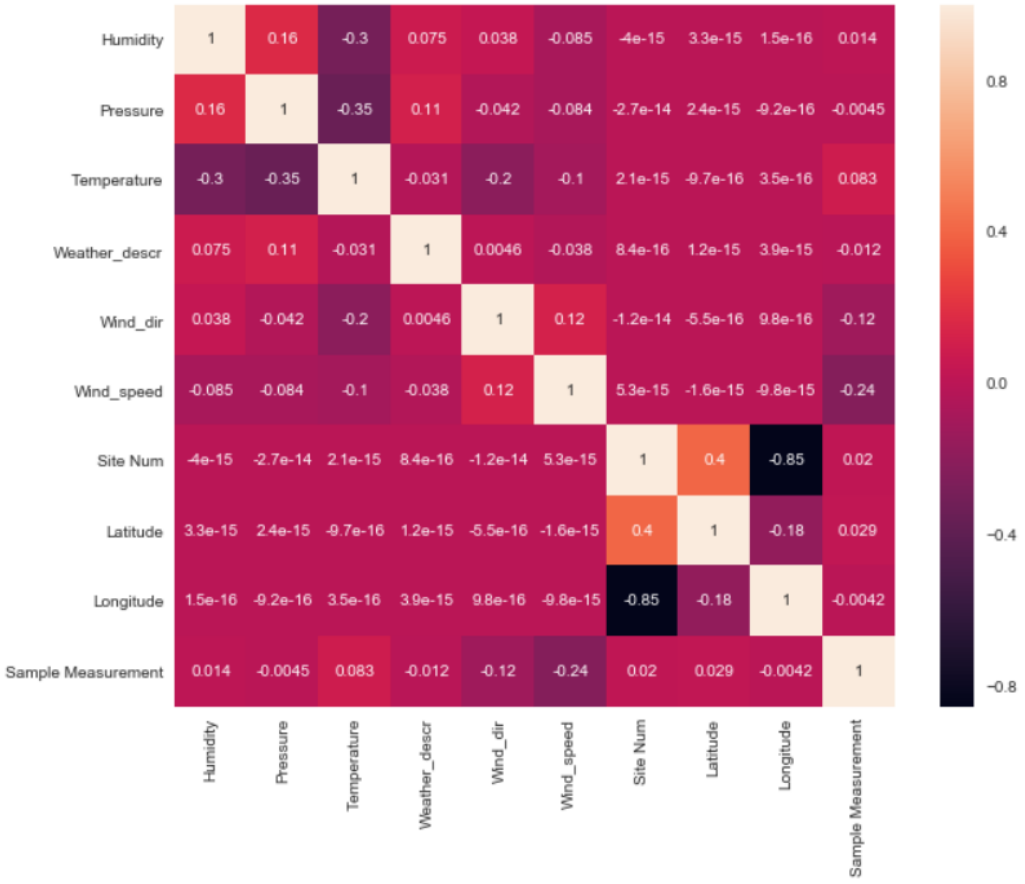

Figure 8: Correlation Between Features

From this last figure, we can notice that the variables that are more correlated to the Sample Measurement are Wind Speed and Wind direction. Their negative correlation makes lot of sense, since when the wind speed increases the value of the sample measurement decrease, this because the wind play an important role in the pollution dispersion. This is even more clear in the following jointplot:

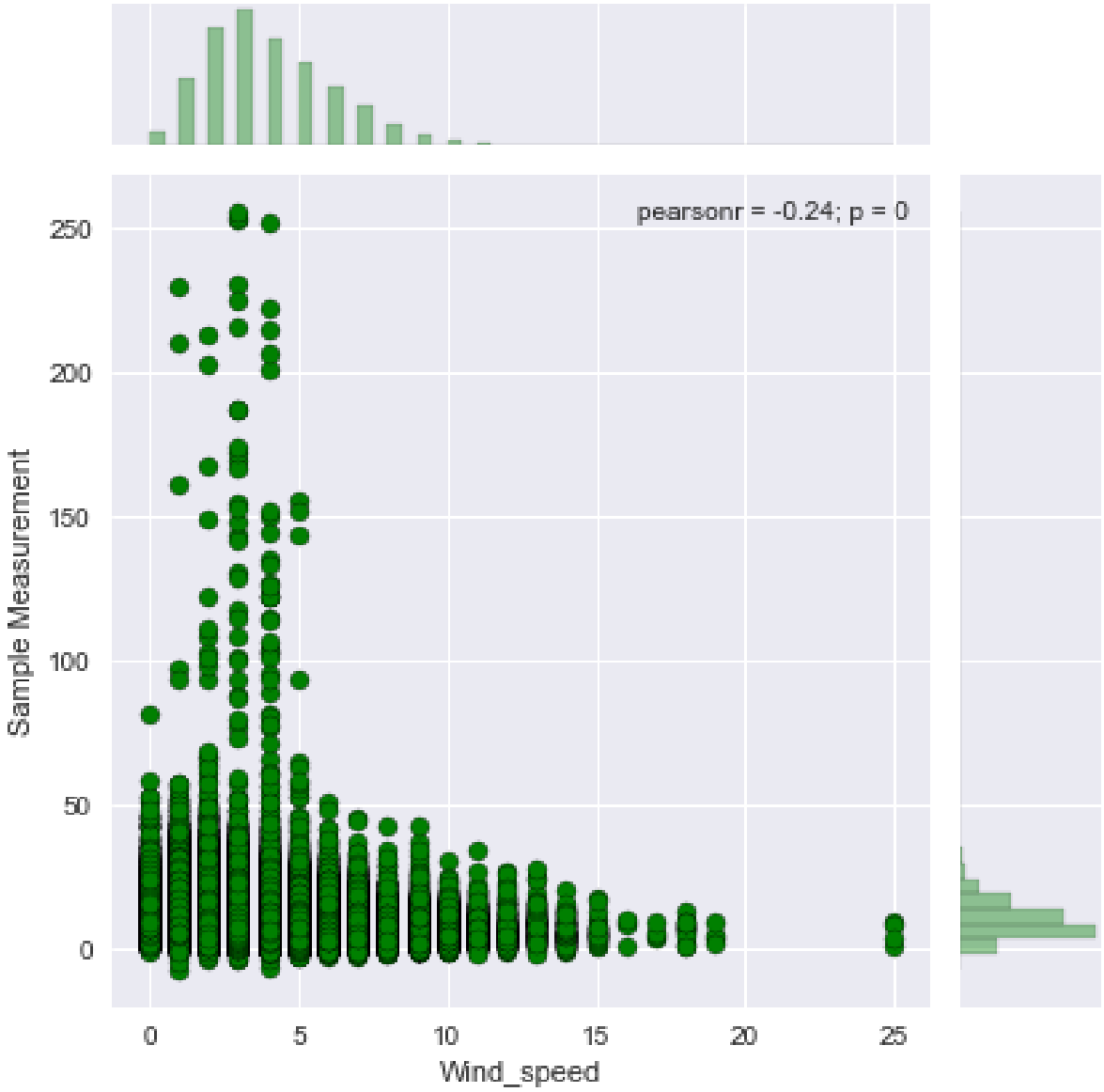

Figure 9: Correlation Between PM2.5 Measures and Wind Speed

CHAPTER 3

FORECASTING EXPERIMENTAL METHODOLOGY

3.1 Introduction

The main objective is to spatially and temporally interpolate and predict the pollutant concentrations at a fine scale and high precision. However, the methodology can be readily extended to other pollutants and other cities.

For the prediction part, the proposed Deep Stacked Bidirectional LSTM will be compared with other three approaches: the Multi-Layer Perceptron, the General Regression Neural Network and the standard LSTM.

The precision of the exposure models will be evaluated using the root mean squared error.

The Deep Stacked Bidirectional LSTM, will be the new model on which we will build our research.

3.2 Time series forecasting problem

Before going deep into the methodology applied in this study, it is important to define the class of problem that we will deal with and to give some background knowledge about it.

It is possible to define a Time Series as a collection of data points ordered in time. It is also common, to define a Time Series as a collection of points equally spaced in time.

A Time Series is composed of discrete-time data and it is frequently visualized through line

charts [46]. In this work the Time Series is represented by the PM 2.5 measures at each monitor station, where each measure is separated by its successor by a fixed time, in this case every hour.

Time series forecasting is a relevant field of machine learning that consists in the usage of previously observed values to predict future values, in contrast to time series analysis that models the time series to detect the main components like trends, seasonal pattern and relation to external factors [47] .

3.3 Long Short-Time Memory Neural Networks(LSTM)

LSTM neural networks belong to the category of Recurrent neural networks(RNN). RNN can store representations of recent input events, using their feedback connections, in form of activations.

This category of neural networks is called recurrent since it contains feedback loops(recurrent edges) and it uses the current input together with the previous states to predict the output at the current time. In Figure 10 it is possible to observe the structure of a LSTM unfolded in time:

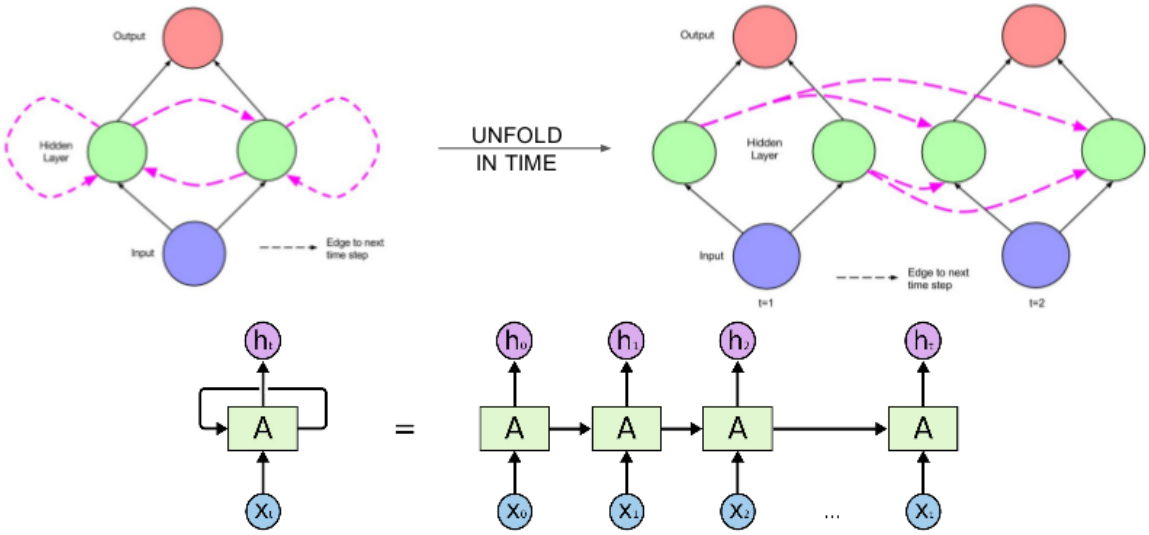

Figure 10: LSTM unfolded in time

Usually recurrent neural networks are trained using Back-propagation through time (BPTT). During Backpropagation, the error signals flowing backwards in time are multiplied and tend to either vanish or blow up with Real-Time Recurrent Learning or the traditional Back-Propagation Through Time, so they cant capture long-term dependencies.

The size of the weights influence in an exponential way the temporal evolution of the backpropagated error. Oscillating weights may lead to blowing up gradients, while in the case of vanishing gradients,a prohibitive amount of time is taken to learn how to bridge long time lags, or does not work at all. Hochreiter and Schmidhuber [20] managed to overcome this obstacle with the introduction of the Long Short Term Memory (LSTM) Neural Networks. This new model is

a type of Recurrent Neural Network architecture that allows learning long-term dependencies and at the same time addresses the vanishing/exploding gradient problem. It is capable to learn how to bridge time intervals without losing the short time lag capabilities, even in case of noisy or long input sequences,

The error flow in LSTM is kept almost constant, with only some small linear modification, thanks to an efficient implementation of the gradient descent algorithm, that allow the error to flow in the memory cell of the LSTM, and the use of specific gate that regulate the flow of information to keep or discard. This allow to have an error flow that is not exploding or vanishing.

The main idea of this kind of RNN is that it is possible to maintain the memory state over time thanks to the presence of a memory cell (interchangeably block). This memory state is characterized by the presence of gating units which can regulate the flow of information in memory and by the presence of the cell state vector, that is an explicit memory present in all the LSTM cells in which the information about previous states flow or are modified based on the value of the gating units [21] .

In Figure 11 a schematic view of the LSTM memory cell is presented:

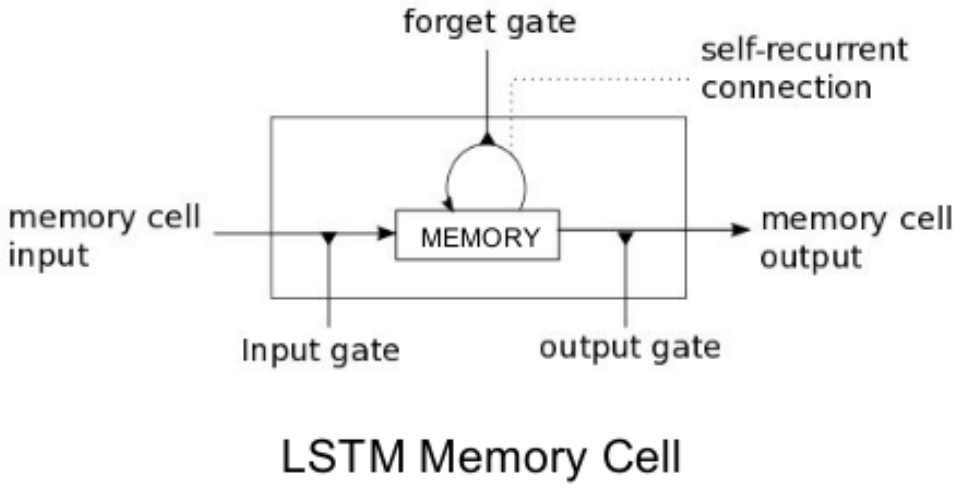

Figure 11: Schematic view of the LSTM memory cell

Long Short-Term Memory (LSTM) is based on the gradient descent method and can truncate the gradient where this does not do harm. The multiplicative gate units, work like a valve that can decide how many information should pass and how many of them should be discarded. The gatig units receive a value between 0 and 1. The value 0 means taht no information should pass while the value 1 indicates that the entire flow of information should pass. All the values in the middle between 0 and 1 indicate what is the quantity of information that should pass. During the learning process, they learn to close and open to modify the constant error flow of the cell state. The computation complexity of LSTM in term of space and time, is O(1) per time step and per weight [21].

The standard LSTM RNN architecture is composed by 3 layers, in order : the input, recurrent(LSTM) and output layer. The input layer is the first layer in the model and it is connected to the LSTM layer. The LSTM's cell output is directly linked to the cell input units and to the 3 gates (input, output and forget gates) through the recurrent connections.

For a LSTM network with one cell, it is possible to calculate the total number of parameters N, ignoring the bias with the following formula:

$$N = n_c * n_c * 4 + n_i * n_c * 4 + n_c * n_o + n_c * 3$$

where n_i represents the number of input units, n_c represents the number of memory cells and no is the number of output units [48] .

A more detailed view is presented in Figure 12.

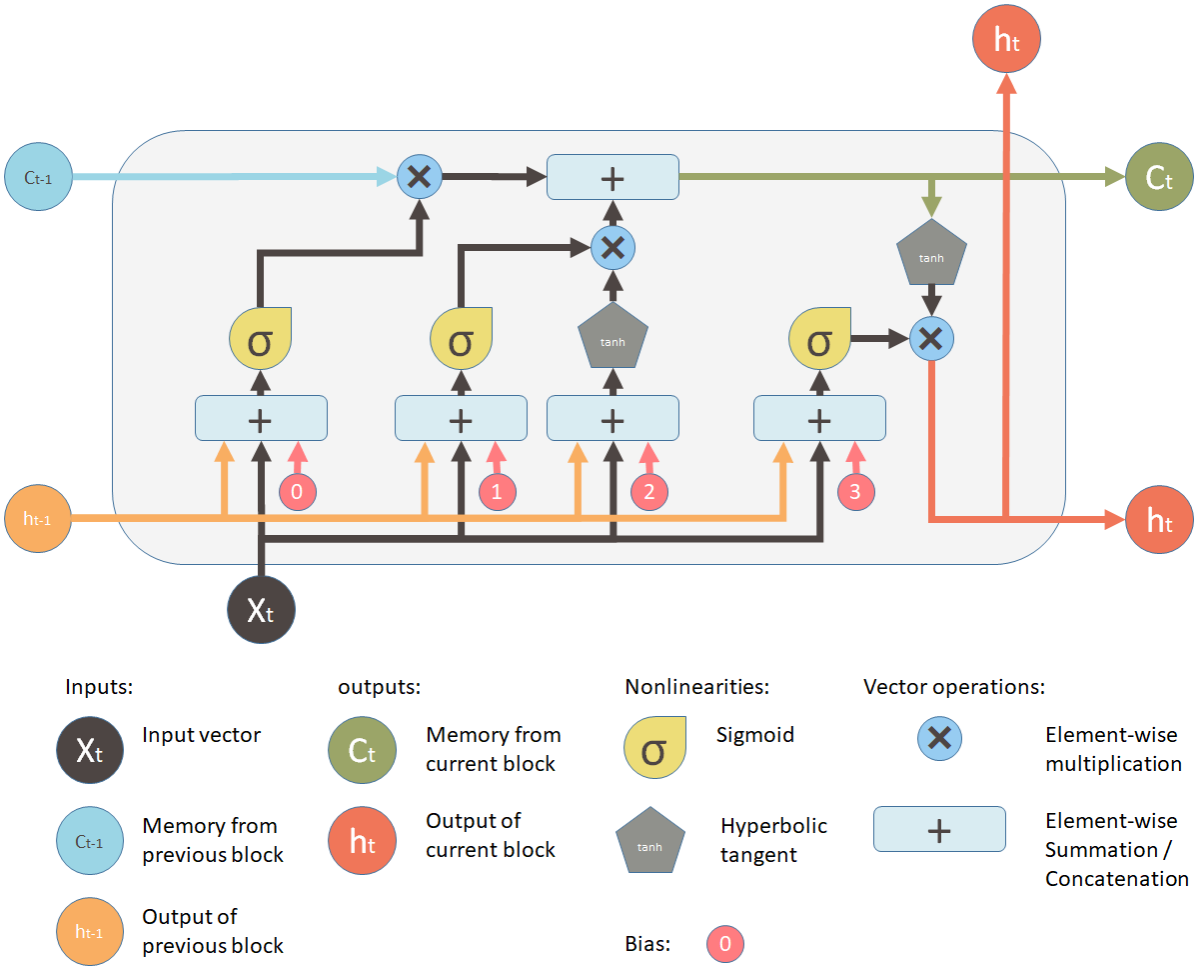

Figure 12: Detailed view of the LSTM memory cell

The memory of the LSTM is represented by the cell state vector. It experiences changes through the forget gate, used to forget the old memory, and through the input gate, used to add of new memory.

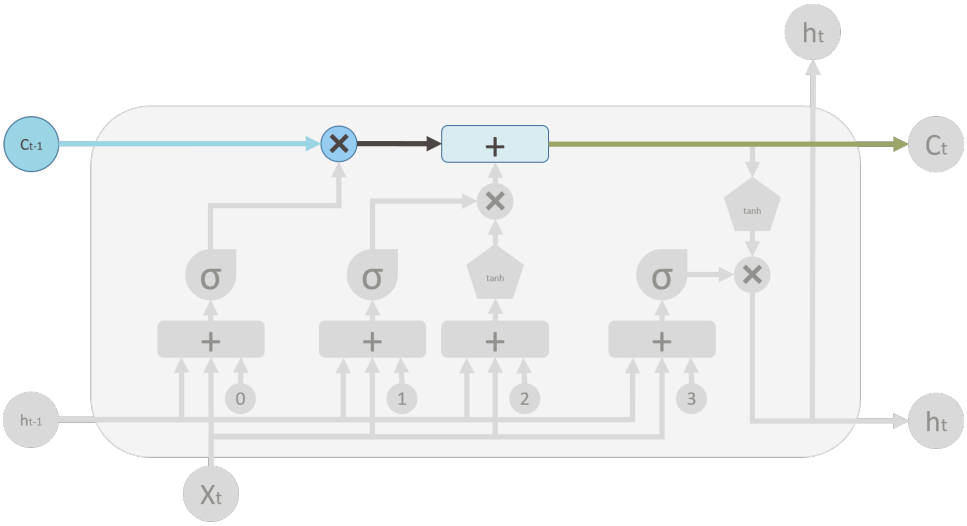

Figure 13: Cell State vector

Another component is the gate, a sigmoid layer followed by pointwise multiplication operator. The main function of the gates is to control the flow of information to/from the memory(highlighted above). They work like valves that can decide the amount of information that can pass through them. They are controlled by a concatenation of the the current input with the output from the previous time step and optionally the cell state vector.

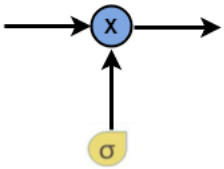

Figure 14: Gate

The main functionality of the forget gate is to decide which information to throw away from memory (delete old memory).

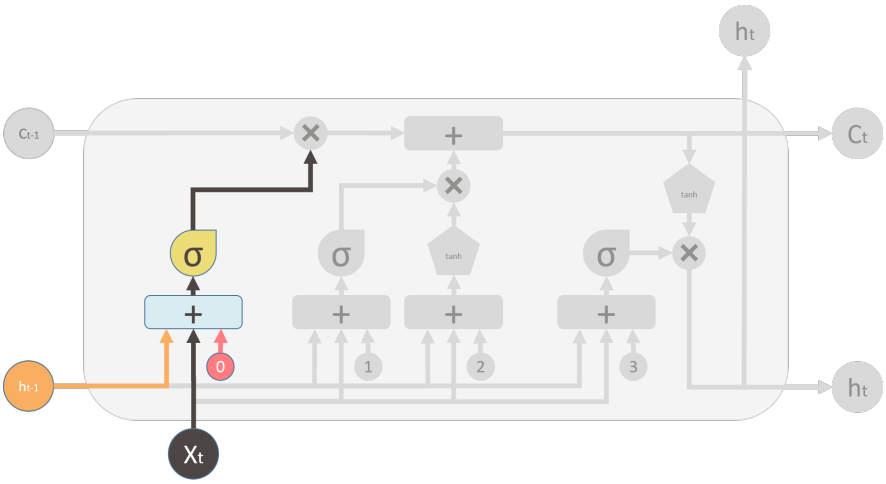

Figure 15: Forget Gate Input

This decision is made by the forget gate layer, a layer composed by a sigmoid activation function. The forget gate works as follows:
By taking the values of h_{t-1} and x_t, for each number in the cell state Ct1, it outputs a value in the range between 0 and 1 . A value of 0 represents completely get rid of this, while a value of 1 represents completely keep this [49]

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

The following step is to establish what data will be kept in the cell state. The process is divided in two parts. In the first part, a sigmoid layer called input gate layer make the decision on which values to update. In the second part, a list of new candidate values, Ct, is created by an hyperbolic tangent (tanh) layer. The final step, consists in combining the previous two parts to generate an update to the state. The input gate is used to decide what is the amount of information from the current input that can be added to cell state. [49; 50]

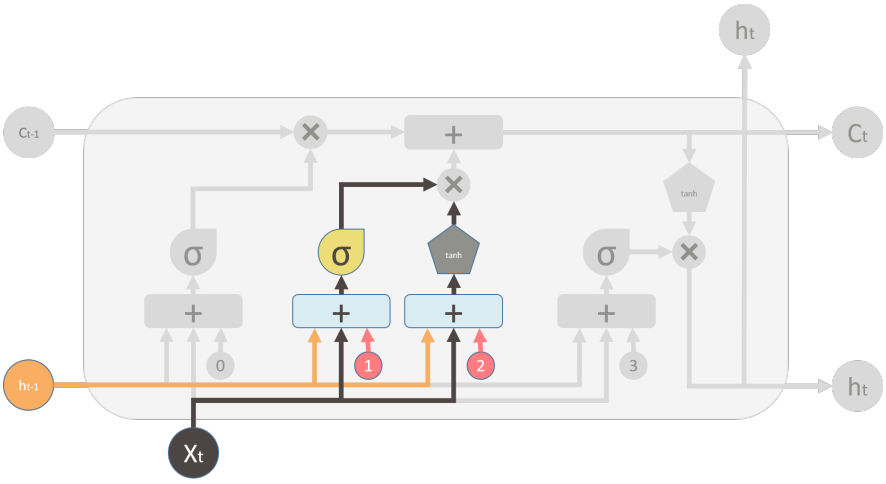

Figure 16: Input Gate

This multiplicative input gate protect the memory contents stored from perturbation by irrelevant inputs.

$$\tilde{C}_t = \tanh(W_c * [h_{t-1}, x_t] + b_C)$$

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

$C_{t\text{-}1}$, that represents the old cell state is updated into the new cell state C_t.

The forget process instead, is done by multiplying the old state by f_t, so that the network can forget the information that are no longer necessary . Then $i_t * \widetilde{C}_t$ is added.

This value is scaled by the amount that was decided in the previous step and represents the new candidate values that will be used to update the cell state.

Let's take as example the language model. This part corresponds to the moment in which the information about the old subjects gender are dropped and the new information are added. [49]

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t)$$

The two components, namely the new memory via the input gate and the old memory via the forget gate are aggregated in the cell state vector.

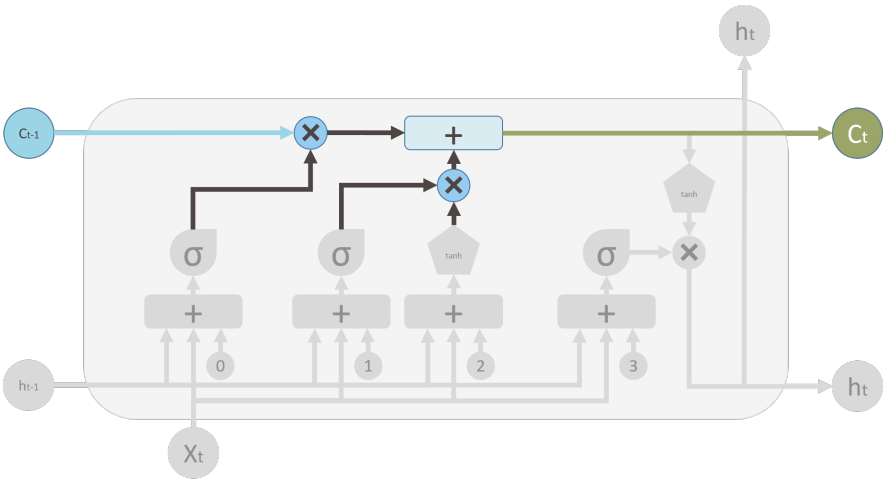

Figure 17: Memory update

The last step is to determine what will be the output of the network. The output is derived from the cell state, but with some modifications. A sigmoid layer is used to decide what parts of the cell state will be the output. Then, an hyperbolic tangent will be applied to the cell state,so that the values will be forced to be in the range [- 1, 1]. These values will be then multiplied with the sigmoid gate output, so that only the a specific part of it will represent the final output.

If we go back to the language model example we can see this process as follows: When the network will see a subject, in the case that what is coming next is a verb, it will output information about a verb. For instance, the output can be if the subject is plural or singular, so that it will be possible to know how to conjugate the verb's form in the case that a verb will follow next.

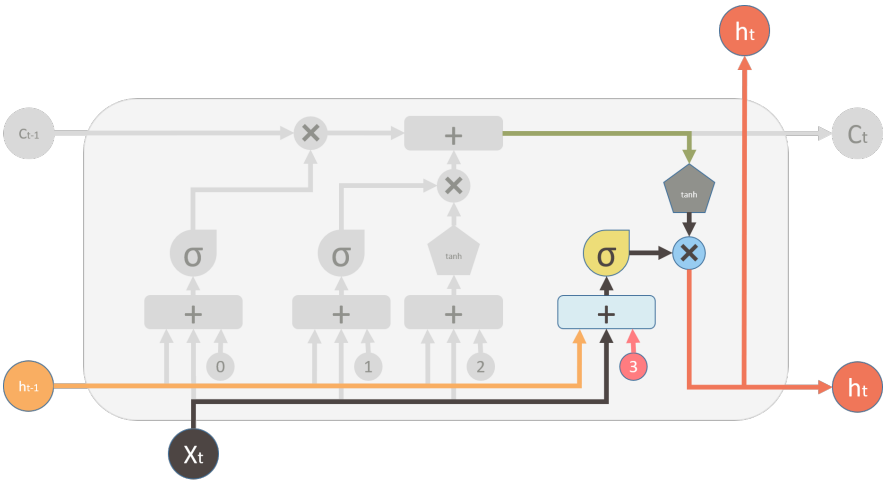

Figure 18: LSTM Output

$$h_t = o_t * tanh(C_t)$$

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$

3.4 Deep Bidirectional Long Short-Time Memory Neural Networks (BDLSTM)

The BDLSTM is the new model proposed in this study. We will proceed step by step, to understand the reasoning behind this kind of network. Since this BDLSTM is an extension of the classical LSTM, all the notions presented in the previous paragraph will apply to this network.

3.4.1 Deep LSTM

By vertically stacking multiple LSTM layers, it is possible to create Deep Long Short Term Memory Neural Networks (DLSTM), in which the input sequence of the next layer is represented by the output sequence of the previous layer. Given sufficient data and by increasing the number of parameters(more layers imply more parameters), Deep LSTM can significantly outperforms the standard LSTM with a single layer.

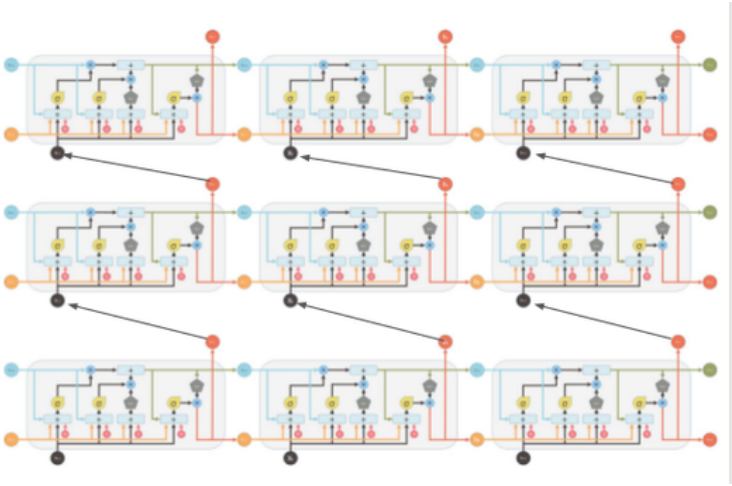

Figure 19: Stacked LSTM

Note that by considering the LSTM RNNs as feed forward neural network unrolled in time, it is already possible to recognize them as deep architectures. When the network is unrolled in time, the model parameters are shared between each layer.

It is possible to notice that even if multiple non linear layers are applied to the input, only a single nonlinear layer is used to process the features from a given time instant, before contributing to the output. Therefore, the depth of deep LSTM can be seen in a different way. When we feed the network with a new input at a specific time step, this input pass through several LSTM layers and it is backpropagated through time. Another advantage of Deep LSTM over the standard LSTM, is the fact that they can distribute the network parameters through

multiple layers, improving their usage for the network.

3.4.2 Bidirectional LSTM

The idea of Bidirectional LSTM comes from the possibility of processing the input sequence in chronological and anti-chronological order using two separate hidden layers. The result is then concatenated or averaged and fed forwarded into the same output layer. Bidirectional RNNs show that they can outperform their unidirectional counterpart in many fields, including speech recognition [51]. The main reason is that by processing the sequence in both way, they can take advantage of the 2 direction to better exploit the context and learn new patterns.
The standard RNNs are usually time or order dependent, they process the input sequence timestamps in chronological order. On the contrary the Bidirectional RNN consists of two RNNs, one to process the data in chronological order and the other one to process the data in an anti-chronological order and then merging the two representations. In this way, the BRNN obtain a richer representation of may capture patterns that may have been missed by the regular RNN.
A recent study by Cui et al. [24], applies a Bidirectional LSTM to forecast network wide traffic speed and the results obtained outperformed the traditional LSTM models. Keeping in mind the following considerations, we decided to undertake this direction and include the Bidirectional LSTM in our model.

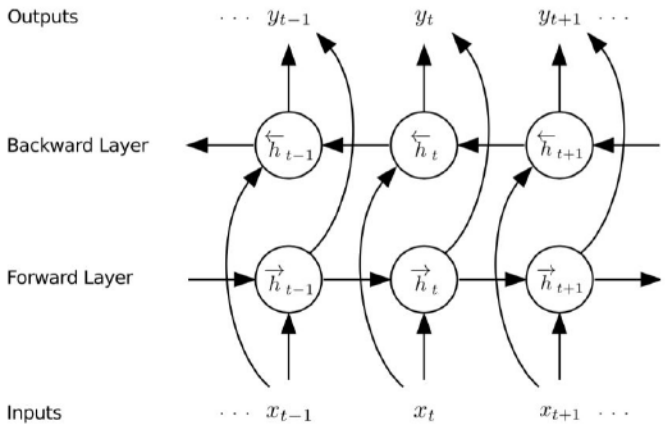

Figure 20: Bidirectional RNN

A BRNN architecture is presented in Figure 20. In general, the network calculates the 2 hidden sequences $\vec{h}$ and $\overleftarrow{h}$ (respectively forward and backward sequence). Then the forward layer from t = 1 to T together with backward layer from t = are iterated. At this point the output layer is updated and the output sequence y is computed:

$$\vec{h}_t = H(W_{x\vec{h}}x_t + W_{\vec{h}\vec{h}}\vec{h}_{t+1} + b_{\vec{h}})$$

$$\overleftarrow{h}_t = H(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}})$$

$$y_t = W_{\vec{h}y}\vec{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y$$

Combining BRNNs with LSTM gives bidirectional LSTM [52] The following picture shows an unfolded version of a bidirectional LSTM:

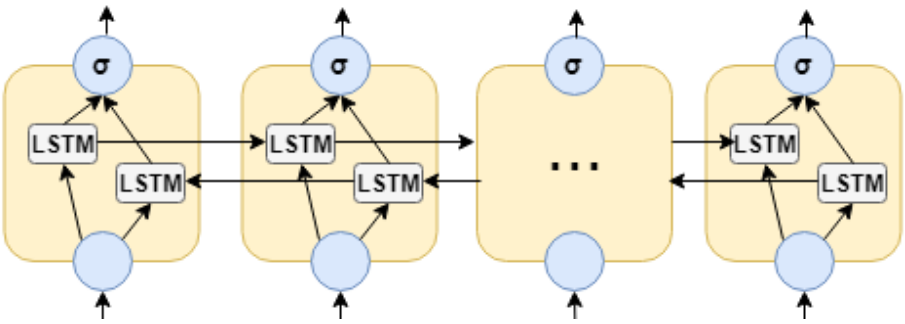

Figure 21: Unfolded Bidirectional LSTM

The bidirectional LSTM is based on the same concept of the BRNN of using forward and backward layers, in particular for both layers the standard LSTM update equations are used.

The BDLSTM layer generates an output vector, Y_T using this formula:

$$y_t = \sigma(\vec{h}_t, \overleftarrow{h}_t)$$

where σ function is used to combine the two output sequences. It can be a concatenating function, a summation function, an average function or a multiplication function. Similar to the LSTM layer, the final output of a BDLSTM layer can be represented by a vector, $Y_T = [y_{T-n,\ldots,y_{T-1}}]$, in which the last element, y_{T-1} is the predicted value for the next hour

[24].

In our work, we applied the Bidirectional LSTM in a slightly different way, with respect to their traditional usage. In particular, we followed the idea of Bidirectional Training proposed by Osogami et al (2017) [53].

The main difference between the Bidirectional Training and the traditional Bidirectional LSTM is that the last one uses both the sequence of the future and the sequence of the past during training. In our experiment instead, we still use the backward and forward model but only the past input sequence, without using the future sequence, since it is our target that we want to predict.

To summarize, we can say that our model uses backward and forward processing but only on past values, while the Bidirectional LSTM does the same but also on future values.

3.4.3 Deep Bidirectional LSTM

Existing studies [54] have demonstrated that deep LSTM architectures with more than one hidden layers can outperform the one with a single hidden layer by building an higher level of representation from the data.

The deep LSTM architectures are networks with several stacked LSTM hidden layers, in which the output of a LSTM hidden layer will be fed as the input into the subsequent LSTM hidden layer [24].

As we mentioned in the previous paragraph, BDLSTM can use both forward and backward dependencies.

Several combinations of hyperparameters (number of neurons, number of layers, batch size,

number of epochs) will be tested in the experiments. The optimal number of stacked BDLSTM layer will be determined by a grid search technique, to find the optimal value that can maximize the performance of our framework.

In theory, several studies show that the optimal value of stacked BDLSTM range from 2 to 5, so we will test only this combination of stacked layers.

At the top of the stacked BDLSTM we will add on or more fully connected layers that are appropriate for producing a higher-order feature representation [54].

Figure 22 represents the high-level structure of our deep neural network, since as mentioned before, the optimal number of layers and neurons will be determined after repeated experiments.

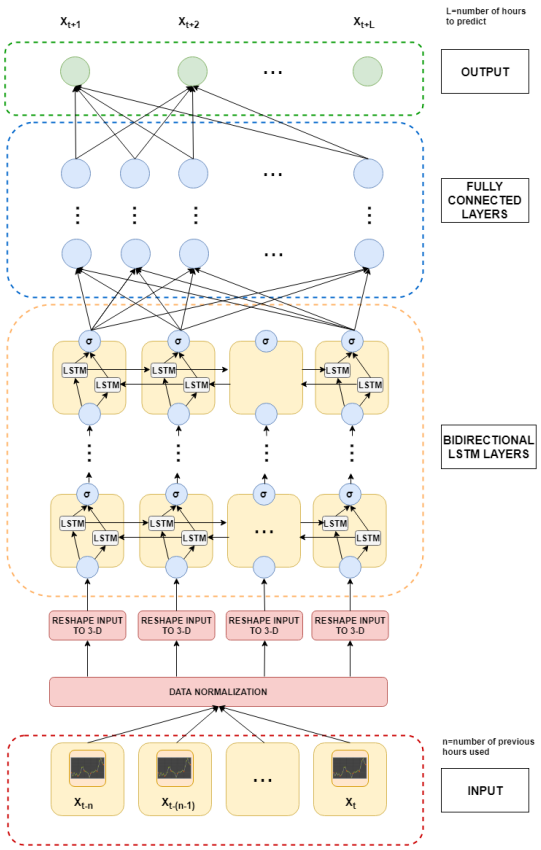

Figure 22: Proposed Model

The neural network architecture was implemented in Python 3.6 environment using Keras [55] (The Python Deep Learning Library) with Tensorflow [56] as backend.

3.4.4 Multiple Step Ahead Forecasting

The basic setup of Time Series Forecasting problems is to predict the next timestamp, that in our study is the next hour.

There some specific domains, in which is important to forecast more than one single step in the future. Examples are meteorological forecasting, financial forecasting and demand or supply forecasting.

In the domain of air pollutants, it is also very important to forecast more than one step in the future, to detect possible dangers for the subject at high risk but also to avoid people to be exposed to hazardous level of pollution. For this reason, we believe that it is important for our model to predict short term future concentration from 1 to 4 hours. In this way it will be possible to give specific alarms and safety guideline depending on the entity of the pollution event.

The approach chosen for the multiple step prediction is called Recursive Multi-stage prediction. In this approach we predict the future pollutants concentration in a step by step manner. We first start by predicting the Y value at the timestamp $t + 1$ using the previous n values $X_{t+1-n}, ..., X_{t-1}, X_t$

where n represents the number of time lags. After predicting this value, we feed it in our dataset, we aggregate it with the corresponding meteorological condition and we use it to make the new prediction for the timestamp $t+2$. We will continue with this procedure until the Y_{t+h} has been calculated, where h represents the number of timestamps we want to predict in the future [57].

The general framework is showed in the following picture:

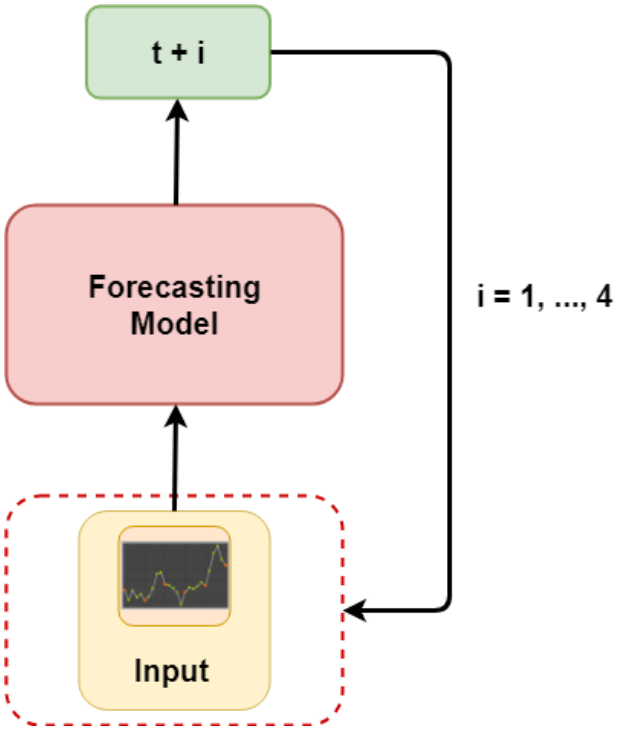

Figure 23: Recursive Multi-Stage Prediction

The advantages of this approach, include the possibility of using a single model for each station to make multiple step ahead predictions. Other methods like Independent Value Prediction would instead require to build a separate model for each future timestamp we want to predict and for each station. This will obviously be too expensive from a computational point of view, because it will imply the creation of 16 different models (4 model for the 4 hours prediction x 4 stations).

One of the main problems of the multi-step ahead forecasting is the so called Error Accumula-

tion that represent the propagation of each single prediction error in the following prediction. This problem is unavoidable in this kind of recursive approach, so in general the error will increase with the increase of the number of future timestamps we want to predict. For this reason, we chose 4 hours as number of future predictions, since the results shows that this value is a good tradeoff between accumulation error and accurate predictions.

CHAPTER 4

EXISTING METHODS FOR COMPARISON

The proposed methodology will be compared with different techniques. In particular in this work we focused on other Neural Networks techniques that are typically applied in time series forecasting problems.

4.1 Multi Layer Perceptron(MLP)

One of the most frequently adopted and most known type of neural network is the Multi Layer Perceptron. Most of the times, there is only a single direction in which the information are transmitted through the network : from input to output. Differently from what we can observe in Recurrent Neural Networks, in this case, the output of each neuron can't affect the neuron itself, but only the neurons of the successive layers, since there are no loops.

All the layers,excluding the first (input) and the last(output), aren't directly connected to the environment. Sometimes, there are some debates in the literature about the fact of considering the input layer (he first layer of the network) as a standalone layer in the network or not. The main reason is that the main functionality of this layer is to transmit the information to the successive layers, without executing any real processing operations over the input data. As we can observe in the next paragraph, it is possible to generate decision boundaries with the form of a semi-plane using only a simple perceptron with one input and a single layer.

It is possible to generate much more complex decision boundaries than a simple semi plane,

by adding additional layers. The effect of increasing the number of layers, is that each neuron behave like normal perceptron for the outputs of the neurons of the the previous layer. In this way the model is able to generate convex decision boundaries, generated by intersecting several semi planes generated by the other neurons. Talking about computing power, several studies demonstrate that stacking several layers with a linear activation function, don't lead to an increase of it, since applying a linear function over another linear function will result in a new linear function. The real power of multilayer perceptron will be fully exploited by using non-linear activation functions.

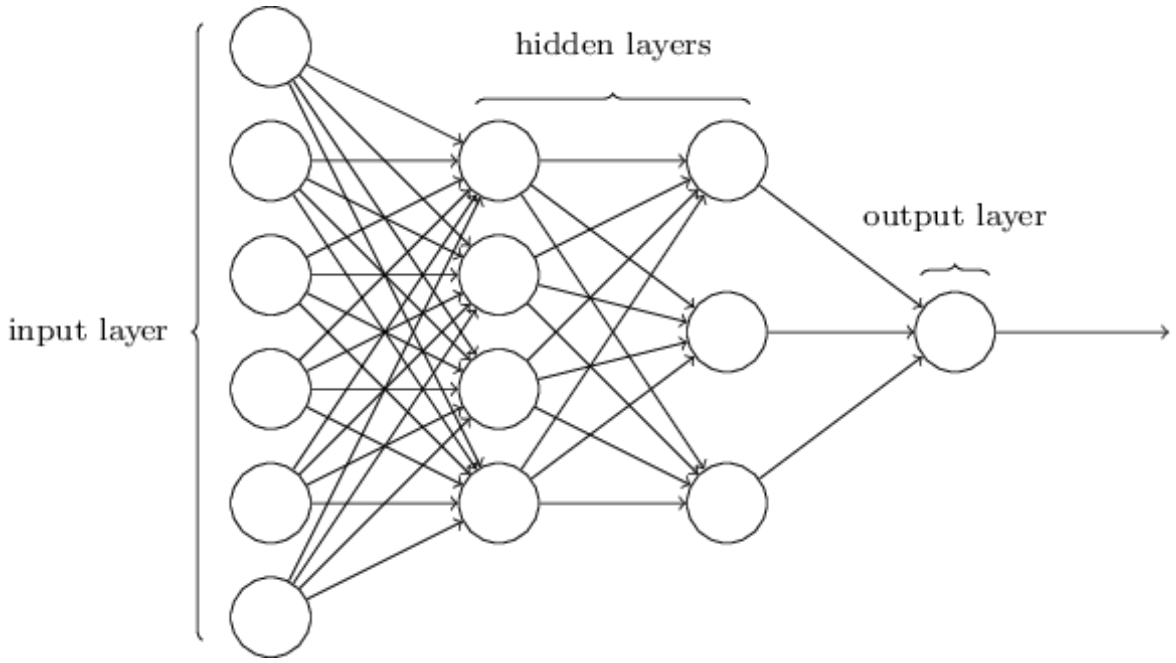

Figure 24: Multi Layer Perceptron

Each of its units forms a weighted sum of its inputs to which are added a constant. This amount is then passed through a nonlinear function which is often called the activation function [58]. Most units are interconnected in a manner "feed forward" as in this figure:

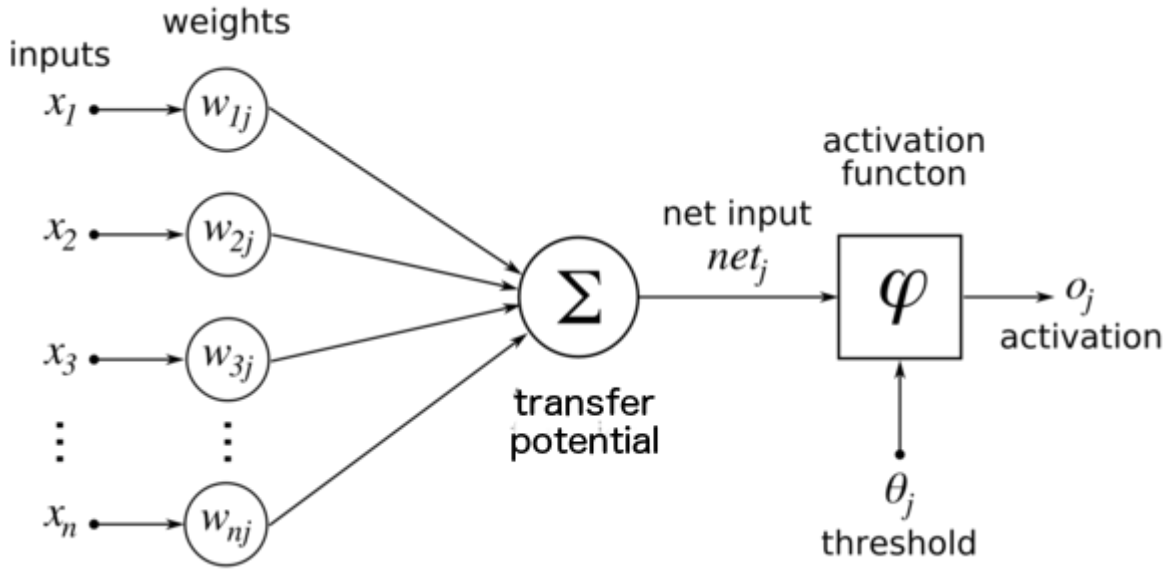

Figure 25: Feed Forward Example

In the first layer, the d-dimensional input x is multiplied by the matrix of weights w as we can see in the following formula:

$$b_j = \sum_{i=0}^{d} w_{ji}^{(1)} x_i \qquad j = 1, 2, ...n$$

The output of each neuron is then transformed, using an activation function, that typically is non-linear. The most used one are sigmoid, hyperbolic tangent and the rectified linear unit.

$$o_j = \varphi(b_j) = \frac{1}{1 + \exp(-b_j)}$$

In general,o_j represent the output of a hidden unit after applying an activation function. The hidden units have this name because differently from the input unit and the output unit they don't have a value specified by the problem or a target value that is used during the training process of the network.

In the same way, each layer following the first hidden layer, give an output that is obtained from the linear combination of the output of the previous layer and the new matrix of weights of the new layer:

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j \qquad k = 1, 2, ...K$$

This transformation, is dependent on the new weights w_{kj} . After the linear combination described before between weights and output of the previous layer, an activation function is applied.

It is possible to combine the previously described equation to obtain a new equation that is capable to describe propagation through the network of the input. It is also possible to describe how, from an input vector, the output vector is computed, given the weight matrices:

$$y_k = g(\sum_{i=0}^{M} w_{kj}^{(2)} h(\sum_{i=0}^{d} w_{ji}^{(1)} x_i))$$

4.2 General Regression Neural Network(GRNN)

The GRNN [59] falls into the category of Probabilistic Neural Networks. It is characterized by four layers:

- Input
- Hidden
- Summation
- Output

The General Regression Neural Network is represented by the following figure:

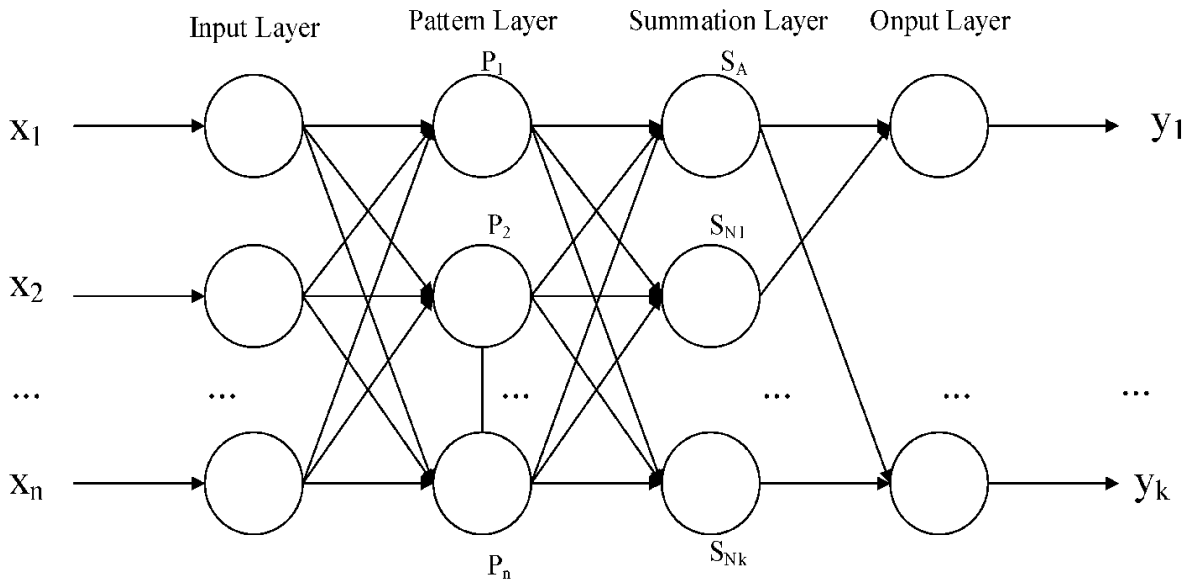

Figure 26: General Regression Neural Network

The GRNN use the Normal Distribution as probability density function. The network makes use of each training example, X_i, and utilize it as the normal distribution's mean:

$$Y(x) = \frac{\sum_{k=1}^{n} Y_k exp(\frac{-D_k^2}{2\sigma^2})}{\sum_{k=1}^{n} exp(\frac{-D_k^2}{2\sigma^2})}$$

$$D_k^2 = (X - X_k)^T * (X - X_k)$$

To measure how good the prediction is represented by the the position of the training sample, we use the distance, D_k, between the training example and the predicted value. The

value of $exp(\frac{-D_k^2}{2\sigma^2})$ vary based on the distance D_k between training example and predictions. If this distance is small, its value becomes big.

When the value of D_k is equal to zero, $exp(\frac{-D_k^2}{2\sigma^2})$ becomes 1 and the training example represent the point of evaluation.

For all the other training examples the distance is bigger.

On the contrary when the value of D_i is big, the $exp(\frac{-D_k^2}{2\sigma^2})$ becomes smaller. For this reason all the other training examples give a small contribution to the prediction

Most of the contributions to the prediction is given by $Y_i * exp(\frac{-D_k^2}{2\sigma^2})$, that is the biggest for the jth training example. The only parameters that need to be set in this case, is the smoothness parameter or standard deviation σ. To conclude we can say that the only parameter of the network is the smoothness parameter. The search space of this parameter can be quite big and most important application dependent, since different problems configurations may lead to different optimal σ values [59].

4.3 Random Forest Regressor (RF)

Random Forest is an Ensemble method, that is based on a technique called Bagging or Bootstrap Aggregation.

Lets suppose we have a dataset D. At each iteration a sample Di of the dataset is created using sampling with replacement (bootstrap aggregation).

A classifier Ci is learned from Di. Each classifier makes a prediction on a new data X and the final classifier C* will take the majority vote as final label. The basic way is to give to each

model the same weight, but we can also use different weights for each model.

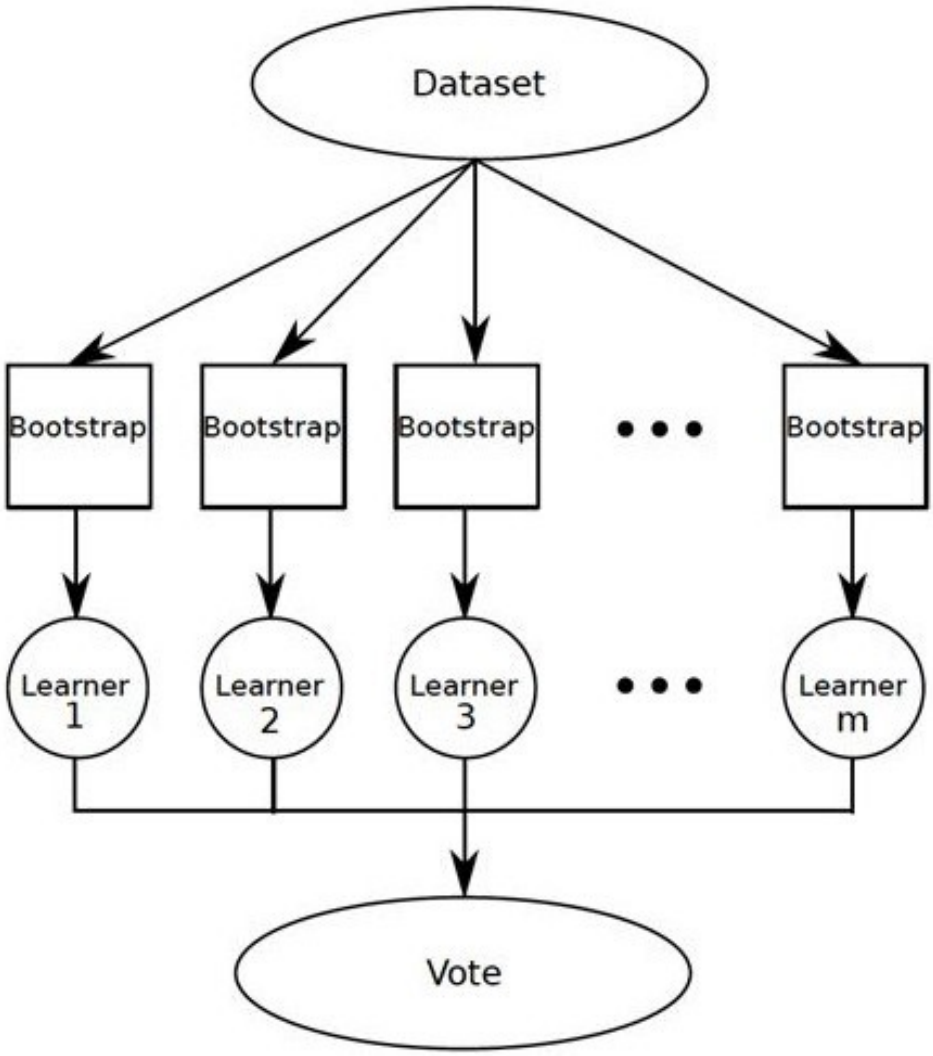

Figure 27: Bagging

In case the prediction is a continuous value, we can take the final value by averaging the results. In general it works good because it reduce the variance, the more classifier we have the better it is.

The Random Forest Regressor is a particular case of the Random Forest in which the variable we want to predict is a continuous variable. This methods is based on the Regression Tree model, in particular at each node the data are split into daughter nodes, by using a splitting criterion. The splitting criterion in the case of Regression is the residual sum of squares:

$$RSS = \sum_{left}(y_i - y_L^*)^2 + \sum_{right}(y_i - y_R^*)^2$$

where y_R^* is the right node mean y value, while y_L^* is the left node mean y value .

The predicted value at a node is the average response variable for all observations in the node.

CHAPTER 5

INTERPOLATION BACKGROUND

5.1 Standard Techniques for Air Quality Inference

In this section we will describe some of the most used Standard Techniques used for air quality inference:

5.1.1 Inverse Distance Weighting Interpolation (IDW)

When we talk about interpolation some of the most common names that come to mind are Inverse Distance Weighted Interpolation(IDW) and Kriging. In our study we selected the first one has methodology to interpolate the pollutants concentration at each station after the prediction stage.

IDW interpolation is a deterministic technique that can be used to interpolate a set of points whose values are known [60].

The main assumption on which this technique is based is the following:

"things that are close to one another are more alike than those that are farther apart."

More precisely, by using the weighted average of the neighborhood values of the unknown points, which are known, it is possible to determine its value. The weights used to calculate the value of the unknown set of points are inversely related to the distances between the sampled locations and the prediction location.

This implies that the closest the known values are to the predicted location, the higher will be their influence on the prediction with respect to the location located far away.

Inverse Distance Weighting Interpolation is based on the assumption that the local influence of each measured point decrease with the distance. The points that are located closer to the location we want to predict have grater weights. These weights are dependent on the distance, since when the distance increase the weights diminish.

This is the main reason for which this technique is called Inverse Distance Weighting [61].

Given a point x based on samples $u_j = u\,(x_j)$ for j = 1,2 N, we can find the interpolated value u applying the IDW as interpolation function.

$$u(\mathbf{x}) = \left\{ \begin{array}{cc} \frac{\sum_{j=1}^{N} w_j(\mathbf{x}) u_j}{\sum_{j=1}^{N} w_j(\mathbf{x})} & if\ dist(\mathbf{x}, \mathbf{x_j}) \neq 0\ for\ each\ j \\ u_j & if\ dist(\mathbf{x}, \mathbf{x_j}) = 0\ for\ some\ j \end{array} \right\}$$

with

$$w_j(\mathbf{x}) = \frac{1}{dist(\mathbf{x}, \mathbf{x_j})^p}$$

As defined by Shepard [62], w_j is the IDW weighting function while x_j is a known interpolated point, x represents an arbitrary interpolated point, N represents the amount of known points involved in the interpolation, *dist* represent the distance metric from the unknown point x to the known point x_j and p is the power parameter.

As we describe before an increase of distance from the interpolated points corresponds to a weight decrease. The magnitude of the weights is used to assign difference influence, depending

on the distance from the interpolated point. The result of the interpolation is a Voronoi diagram made by mosaic of tiles, that for large values of p have almost constant interpolated values. When dealing with two dimensional data, the interpolated values are dominated by far away points in case a power parameter p <= 2 is used [60]. A power values p greater or equal to 1 is typically used in Geostatistical Analysis.

In case, the power parameter p is equal to 2, the technique assume the name of inverse distance squared weighting interpolation. Even if there is not a fixed rule to choose a value of p over another one, the power parameter is usually set to 2 as default. As a general rule, we can say that the parameter p can vary depending on the type of problem. A good way to decide its value, can be to investigate the effect of changing p by examine the cross-validation statistics and previewing the output.

5.1.2 Gaussian Interpolation

Gaussian interpolation is another interpolation technique based on a Gaussian distribution. The parameter represent the average distance between two ground monitoring stations [35]. It can be defined as follow:

$$u(x) \sum x_i f(x),$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{x^2}{\sigma^2}}$$

5.1.3 Kriging

Kriging is an interpolation technique whose basic idea is to compute a function value in a specific point using the weighted average of the known function's values in the points neighborhood. The interpolated values are the result of a Gaussian process governed by prior covariances. [63] The method is very similar to regression analysis. Both the techniques, based on the covariance assumption, try to obtain the best linear unbiased estimator. To prove the independence of the error and the estimate, they use similar formulas that make use of the Gauss-Markov theorem. Nevertheless, they are used in different scenarios: regression models use multiple observations from a multivariate set of data, while kriging is used to estimate a single random field realization. [64] Similarly to IDW, Kriging derive the prediction of unmeasured points by giving a weight to the surrounding measured values:

$$u(x) = \sum \lambda_i u(x_i)$$

where:

- $u(x_i)$: the value registered at the ith location
- **x**: the prediction location
- **N**: the number of measured values
- λ_i: an unknown weight for the measured value at the ith location

The main difference is that between Inverse Distance Weighted Interpolation and Kriging is that, in the first one the weight, λ_i is only based on the location's prediction, while in

the second one, the weights are also based on the measured points spatial arrangement. The quantification of the spatial autocorrelation is a prerequisite in the use of the weights. For these reasons, in Kriging the weight,λ_i, depends on the distance to the prediction location, the spatial relationships between the values measured around the location predicted and a fitted model to the measured points.

5.1.4 Land Use Regression

Land Use Regression (LUR) is a technique that is often used to analyze air pollution in specific areas. [65]

This algorithm is based on the main assumption that the pollutants concentrations at a specific location is highly influenced by the environmental characteristics of the surrounding area. For this reason, LUR model the dependency between the measured pollutant concentration at a specific monitoring station and a set of explanatory variables like traffic data and land-use data. By constructing multiple regression equations, it is possible to describe the relation between the environmental variables and the monitoring locations. Typically, for each temporal resolution (year, season, month, bi-week, week, day, semi-day) a separate model is created. [66]

The final equation resulting from the prediction variables can be used to predict the pollutants level at any unmeasured locations.

Two main types of predictions can be done:

- Point location (e.g.: residential addresses)
- Grid

In the latter case, it is possible to create grid maps of the area considered in the study.
In general, a comprehensive study of the history and applications of Land Use Regression[66], shows that the most important explanatory variables to estimate the air pollutants concentration are land cover, elevation, traffic and road type. In general traffic has been identified as one of the most important factors.

LUR is typically evaluated using the coefficient of determination R2, a measure that is able to capture the dependent variables variance explained by the independent ones. In the air pollutants field, several studies show that the R2 ranged from 0.54 to 0.81. [66]

In the case of Land Use Regression a good performance can be reached for high temporal resolution like yearly, monthly or weekly models. The daily and semi-daily models start to show a decreasing in the performance of LUR models. For this reason, one of the main drawbacks of this technique is that it is not applicable for higher temporal resolution (e.g.: hourly), since the are not enough explanatory variables with the required resolution to calculate the pollutants concentration of a grid cell.
For our study, LUR doesnt show any practical utility for the creation of hourly pollution map of a city. Based on the previous considerations, this technique wont be considered for comparison.

5.1.5 Dispersion Models

Dispersion models combine several fields including geophysics, meteorology and chemistry to model the transformation and transport of air pollutants in the atmosphere. This process is mainly governed by the wind but also by the characteristics of the pollutants transported and other processes such as chemical reactions and turbulence [67].

During the years, several approaches have been developed to model the dispersion of air pollutants. Most of them are based on the Transport Equation. Assuming horizontal turbulence, and incompressible fluid and neglecting the molecular diffusion, it is possible to express the Transport Equation as follows:

$$\frac{\partial c}{\partial t} = -v\Delta c + S_c + \Delta h(k_h \Delta hc) + \frac{\partial}{\partial} K_z \frac{\partial c}{\partial z}$$

The steady points, in case of the several interpolation methods is assumed to be the monitoring station on which the pollution values are available.

Where Δh is the divergence operator, K_z is the vertical and k_h is the horizontal eddy diffusivity that describe the intensity of the turbulence. [67]

5.1.5.1 Gaussian Plume

The Transport Equation can be integrated analytically and by assuming a steady-state point source at (0, 0, h) and a homogeneous, steady state flow, results into the Gaussian Plume Distribution:

$$c(x,y,z) = \frac{Q}{2\pi\sigma_y\sigma_z u} exp\left(\frac{-y^2}{2\sigma_y^2}\right) exp\left(\frac{-(z-h)^2}{2\sigma_z^2}\right) + exp\left(\frac{-(z+h)^2}{2\sigma_z^2}\right)$$

Looking at the Gaussian Plume Distribution, we can notice that Q is the source term, x is the downwind, c is the time-averaged concentration in a specific point, z is the vertical direction, y is the crosswind and u is the time-averaged wind speed at height h. The standard deviations σ_z and σ_y describe the vertical mixing of the pollutant and the crosswind. [67]

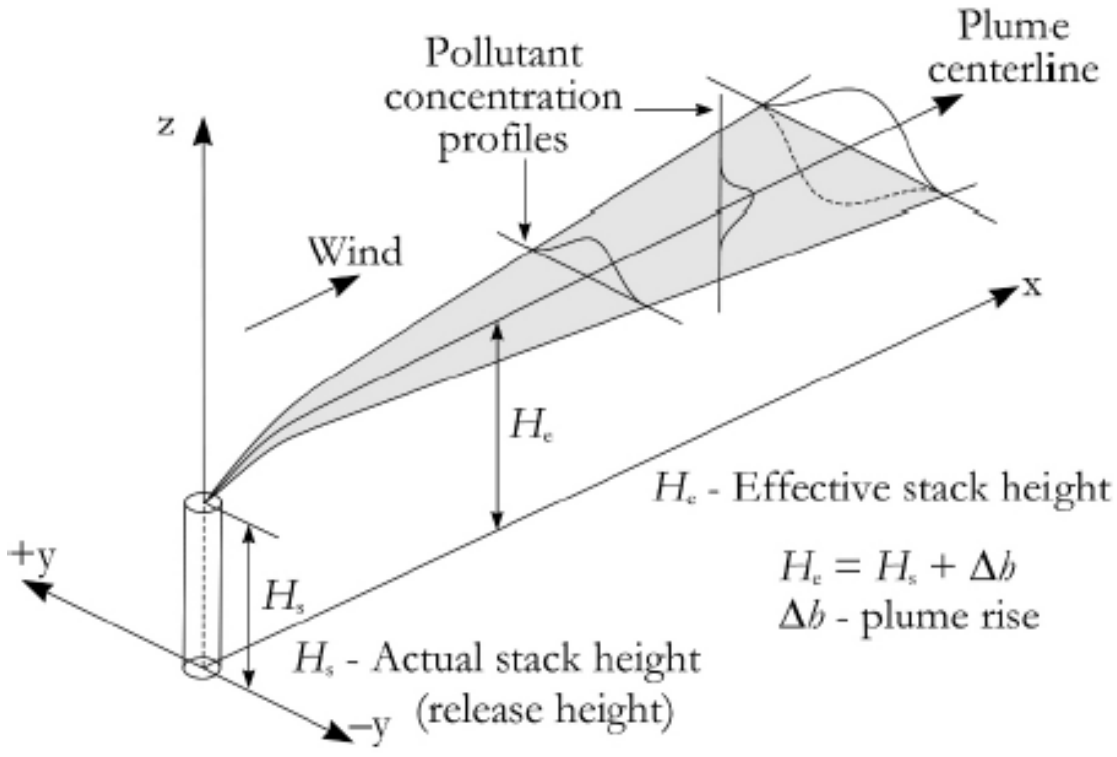

Figure 28: Gaussian Plume

One of the main advantages of the Gaussian Plume is that it is very fast since it only calculates a single equation, even if the turbulence parametrization and meteorological data preprocessing can increase the computational complexity. This model shows poor performance when the wind speed is low. [67]

5.2 Machine Learning Approaches for Air Quality Inference

Machine Learning techniques have been widely for air quality inference. In particular, the problem is seen as a regression problem, in which the target to predict is a continuous value that represent the level of pollution in a specific location. For this reason, most of machine learning techniques that are typically used for regression problem can be used for this purpose. In particular, the most used are Support Vector Regressor, Decision Tree, K-Nearest Neighbor, Nave Bayes and Random Forest.

We wont go through the detailed description of each technique, since they are some standard Machine Learning technique that are well known and widely used in every regression problem.

5.3 Deep Learning Approaches for Air Quality Inference

Recently, new approaches based on Deep Learning have been proposed to deal with the problem of air quality inference. Neural Networks show great results in producing air pollution maps with high temporal resolution and seem to outperform the standard techniques.

U-Air [35] is considered one of the milestones of the application of Deep Learning to Air Quality Inference and for this reason it has been used for comparison for several study in this field.

5.3.1 U-Air: When Urban Air Quality Inference Meets Big Data

In the U-Air study [35], the main goal is to infer real time and fine-grained air quality information ($PM_{2.5}$, SO_2, NO_2 and PM_{10}) in the city of Beijing using several data sources including air monitoring stations data, traffic flow, meteorology, human mobility, point of interests (POIs) and structure of road networks.

Using a semi-supervised learning techniques called co-training, the air quality of an unknown location is inferred, thanks to the use of unlabeled data. To classify the air pollution levels two different classifiers are used: a linear-chain conditional random field (CRF) for modelling the temporal dependencies and an artificial neural network (ANN) that captures the spatial dependencies. The key features of this work include the identification of the most relevant features that contribute to air pollution and their incorporation into a data analytics model.

5.3.1.1 Setting

The setting of the experiment was divided into grid cells of 1km x 1 km based on the assumption that the air quality inside a grid g is uniform. Each grid can be either labeled if it already has an air monitoring station located inside or to be inferred (unlabeled) in case it hasnt. The main assumption is that the air quality of a grid is highly influenced by neighbor grids, in particular, the affecting region considered in the experiment consist of the 8 grids around each cell.

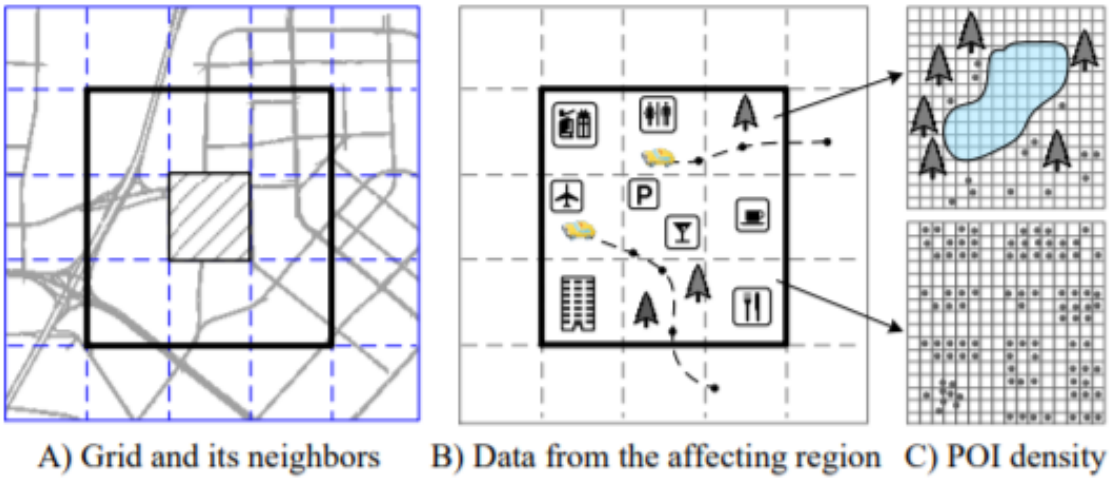

Figure 29: U-Air interpolation setup

5.3.1.2 Framework

The U-Air framework is composed by two separate processes: offline learning and online inference. These two parts generate 3 different data flows: preprocessing, inference and learning. During the preprocessing data flow taxicabs trajectories are mapped on a map and used for offline learning.

In the learning data flow, the features of the affecting regions are extracted from a variety of data sources. The features can be divided into two categories, that include spatially (point of interests and road length) and temporally related features (meteorological and trajectories features). During this stage, the two different classifiers, linear-chain conditional random field (CRF) and the artificial neural network (ANN), are trained using a co-training semi-supervised

approach.

Finally, during the inference data flow, the two classifiers are used to generate two probability scores and the most possible class is selected as label. The inference is a temporal resolution of 1 hour.

5.3.1.3 Features

Meteorological Features: five features were identified, in particular humidity, pressure, temperature, wind speed and weather.

Traffic Related Features: traffic is considered one of the main sources of pollution. For this reason, using the taxi trajectories, 3 features were identified. The features included the expectation of speed, the standard deviation of speed and the distribution of speeds.

Human Mobility Features: these features include the number of people arriving a leaving a grid in the past hour. These features were extracted from the taxicabs drop off and pickup.

Road Network Features: these features were related to the structure of the road network. In particular, for each grid 3 features were identified: the highway length , the road segments length and the number of intersections.

POIs Related Features: Point of Interest related features were used to indicate the land use and the traffic patters of a region. The features included the number of POIs in some category (sports, parks, companies, entertainment) in a specific grid, the number of vacant places in a grid (grid were further divided in sub-cells) and the change in the number of POIs.

5.3.1.4 Co-Training

The co-training framework is a semi-supervised learning technique in which the data are described by two complementary feature sets. Ideally, given the class, these features sets are conditionally independent. In U-Air two different classifiers were proposed: a linear-chain conditional random field (CRF) as temporal classifier (TC) and the artificial neural network (ANN) as spatial classifier (SC).

The two classifiers are trained separately using two disjoint features sets. The Temporal and Spatial Classifiers are used for the inference of the unlabeled grids. The examples that are most confidently classified, are then added to the training set and the process is repeated until there are no more unlabeled grids. The final AQI of a grid is obtained by multiplying the probability scores obtained from the two classifiers.

The *Temporal Classifier (TC)* is a linear chain Conditional Random Field (CRF). It is an undirected probabilistic graphical model to parse sequential data. Typically, it is used for language text, but in this case, it is used with the temporally related features.

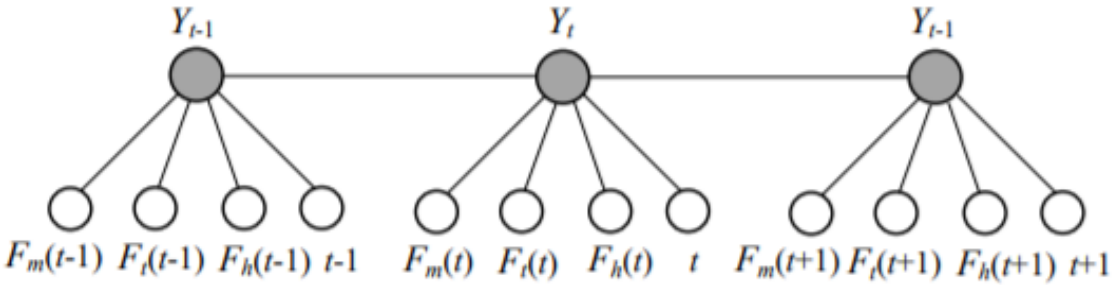

Figure 30: Conditional Random Field

The *Spatial Classifier (SC)* is an artificial neural network (ANN) that makes use of the spatially related features. The type of ANN used is the traditional Back-Propagation (BP) neural network with one hidden layer. For the hidden and output layer a sigmoid function is used.

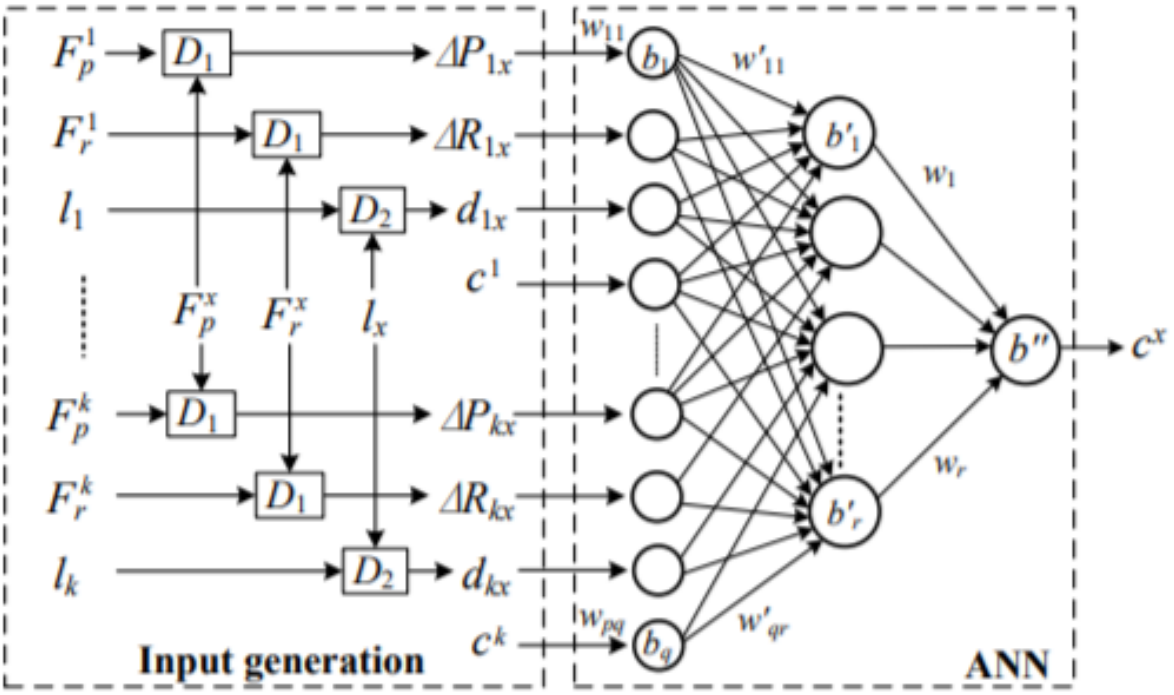

Figure 31: U-Air Neural Network

5.3.1.5 Evaluation

U-Air was compared with other 5 baselines including Inverse Distance Weighted Interpolation, Gaussian Interpolation, Classical Dispersion Model, Decision Tree and CRF-all/ ANN-all. One station was removed from the grid and the remaining station were used to infer its air quality. The AQI of that station was then used as ground truth. This process was repeated for each grid containing a station every hour. The problem was viewed as a classification problem. The AQI value were divided into range using the standard table and then converted into AQI

labels (Good, Moderate , Hazardous). The results were evaluated in term of Precision and Recall. Overall, U-Air outperform the other techniques mentioned above.

CHAPTER 6

INTERPOLATION EXPERIMENTAL METHODOLOGY

In our study we proposed a new approach based on Neural Network with self-training able to infer the Air Quality in a grid with a spatial resolution of 1km x 1km. Following what has been done in other recent studies in the literature [35][33][34], we will apply the inference every hour. In particular, in our case the inference will be performed after the prediction part done by our Deep Bidirectional Neural Network. . In this way it is possible to create pollution maps for the next n hours, where n is the number of future hours predicted by the Deep Bidirectional LSTM.

6.1 Self-training vs Co-Training

The Co-Training methodology was proposed for the first time by Blum et Al. [68]. The Co-training framework divide the features into disjointed set, in which each set should be able to train a good classifier. The training process consists in training two separate classifiers on each feature set and the predictions of one classifier are used to increase the size of the training set of the other classifier. The main assumption on which Co-Training is based is that the two feature sets are independent and they can be divided.

Self-Training instead is based on the following idea: the classifier is first trained on the labeled data.The unlabeled data are then classified using the previously trained model and the most confidently classified examples (labeled this time) are added to the training set. The process

is repeated in a loop, until a specific condition is met. In this way the classifier is gradually refined and it use its own prediction to learn itself. [69].

6.2 Interpolation Setup

The region chosen for the experiment has been divided into disjointed grids (1km x 1km) for a total of 791 cells. Given the following collection of grids:

$$G = G1 \ U \ G2 = \{g1, g2, gn\}$$

where $g.Q$ ($g \ \epsilon \ G1$) is known and $g^{'}.Q$ ($g^{'} \ \epsilon \ G2$) is unobserved, $|G1| << |G2|$ and Q indicating the label of a grid.

The grids containing a monitor station, are the one for which the label Q is known, while the grid without monitor stations are the one for which the label Q is unknown. The grid with an unknown label Q will be inferred thorough and interpolation technique, in a periodic interval of one hour.

The label Q, will be the $PM_{2.5}$ concentration expressed as micrograms per cubic meter (i.e., $\mu g/m^3$). Figure 32 instead show, the entire map of the Chicago Urban Area, divided in grid of 1km x 1km, and containing the EPA stations. As we can notice one of the stations (Station 1) is outside the boundaries of the city. In any case, since it is very close to the considered are we decide to use it in out computations.

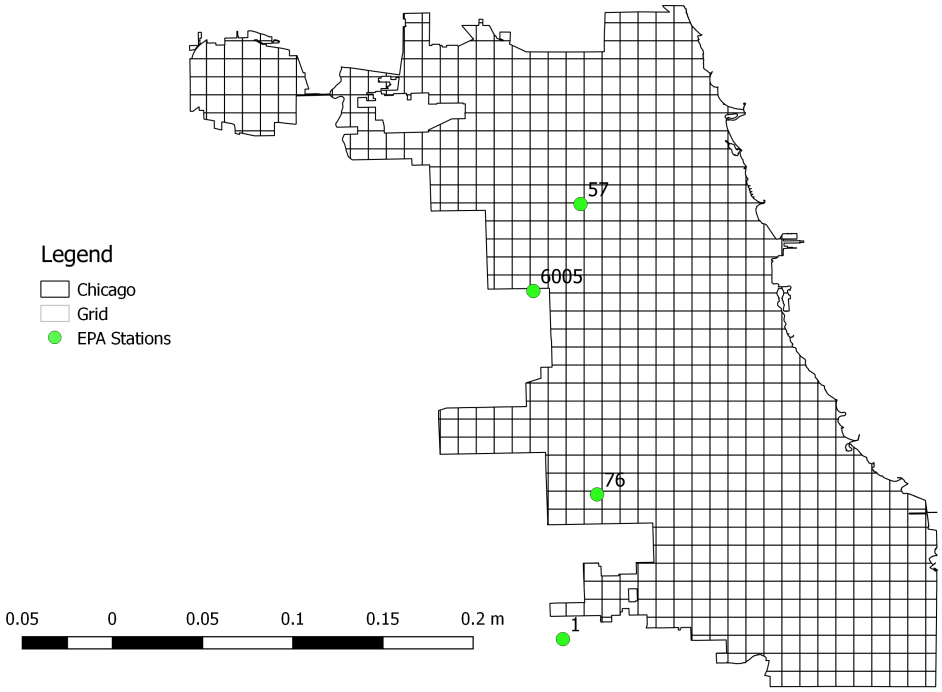

Figure 32: Chicago map with 1km x 1km grid and EPA stations

After creating the grid map of 1km x 1km we map the traffic data and land use data on it. We then match them with the meteorological data and the PM2.5 data coming from the monitoring stations.

We calculate the distance between each labeled and unlabeled cells, this create three different features Distance from closest station, Distance from second closest station and Distance from

third closest station.

Differently from what has been done in the U-Air paper, where two different classifiers a CRF and an ANN were trained using a co-training framework, we use a single model to model the spatio-temporal dependences between the features. The model chosen is an Artificial Neural Network that is trained into two different steps using a self-training approach, opposed to what is done in the U-Air paper where 2 models are used following a co-training approach.

In the first step, we will train the Neural Network using only the 4 labeled cells. For every hour and for every station, we will first remove a station and infer its value with the other 3 station using as ground truth the pollution level of the grid containing the station we removed. In this way we will have (4 stations x 24 hours) 96 instances per day.

Differently from what have been done in the U-Air experiment, where the predicted label for each cell was only one of the 6 possible level of concern (Good, Moderate , Hazardous) of the table provided by the United States Environmental Protection Agency, we decided to predict not just the level of concern of each grid cell but also its specific pollution level of the cell expressed as micrograms per cubic meter (i.e., g/m3).

In this way we were able to deal at the same time with a classification (6 label representing the levels of concern) and regression (pollution level) problem using a single Neural Network.

In order to accomplish this goal, we created a modified version of the neural network model, with two different branches at the end of the network, to be able to produce two different outputs. The first output, a label representing one of the six level of concern, was obtained using a Softmax activation function. The Softmax is a common activation function used in

Deep Learning models. The Softmax output can be compared to a categorical probability distribution, that tells what is the probability of each category to be true. In our case, it outputs 6 probabilities (one for each of the 6 class). The probability with the highest value was used as label for the cell.

The second output, a continuous value representing the pollution level of the cell, was instead obtained using a linear activation function. A linear activation function, simply output the result of the layer as it is, without applying any non-linear function like ReLu or Softmax on it.

We train the network using two different losses, a categorical cross-entropy for the classification branch and the mse for the regression branch. 'Cross-Entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1'. Categorical Cross Entropy is used for multiclass classification instead of the Binary Cross-Entropy used for binary classification. The mse(mean squared error) instead is simply the average squared difference between the estimated values and what is estimated.

We also give specific weights to each loss, in particular we assigned a weight of 0.4 to the classification loss and 0.6 to the regression loss. We decided to do that because we think it was more important to have more precise pollution level values.

The main reason why we decided to use two different branches instead of producing a single output with a continuous value and then use the EPA table to convert the predicted value into one of the six AQI labels, is that with our methodology we were able to output also the Softmax probabilities that we then used for the self-training of the network. Without these probabilities,

it was not possible to apply this semi-supervised learning technique.

After this first step, we extracted the probabilities of each prediction (one for each of the six class). We used a threshold of 0.99 for the probability and we gave a new label to all the unlabeled cells that exceed this threshold. All the new labeled examples that were above or equal to the threshold were added to the training set.

This process, known as Self-Training can be seen as a Pseudo-Labeling approach in which we will increase the number of labeled cells, to improve the performance of our algorithm since we only have few labeled cells (4) and lot of unlabeled cells (more than 700).

In the last step, we will retrain the neural network using the new labeled data added to the training set in the previous step to obtain better results. At this point we will apply our trained neural network to all the remaining unlabeled cells and we will produce some interactive interpolated maps using the plotly library.

The general framework of the interpolation part, is presented in the following schema:

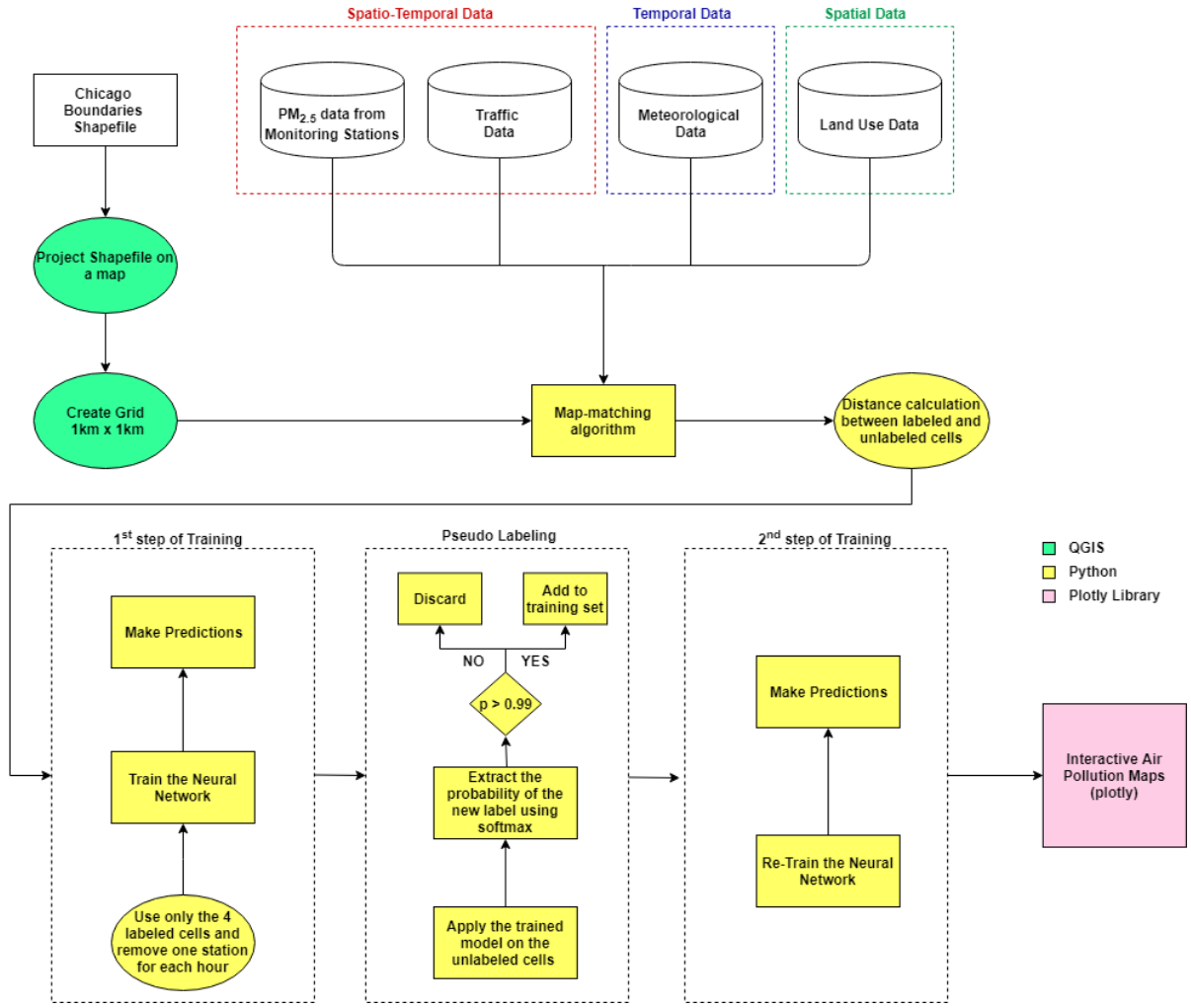

Figure 33: Interpolation Framework

6.3 Features

The $PM_{2.5}$ data and the meteorological data (Humidity, Wind Speed, Pressure) are the same features described in the section regarding the prediction part, so we wont go through a

detailed description of them in this section.

The Land Use Data were obtained from the US Data gov website [70] and mapped on a map. For each 1km x 1km grid cell we calculate the percentage of each land use class that is contained inside it.

In total eleven features were extracted: % of Agriculture Land, % of Commercial Land, % of Industrial Land, % of Institutional Land, % of Non-Parcel Land,% of Open Space, % of Transportation Land,% of Unclassified Land, % of Urbanized Land, % of Vacant Land, % of Water.

These features were extracted by intersecting the grid containing the 1km x 1km cells and the Land Use data. In particular for each grid cell we calculated the sum of each polygon belonging to the same category (see Figure 34) and divide that value by the total area of the cell to obtain the percentage value.

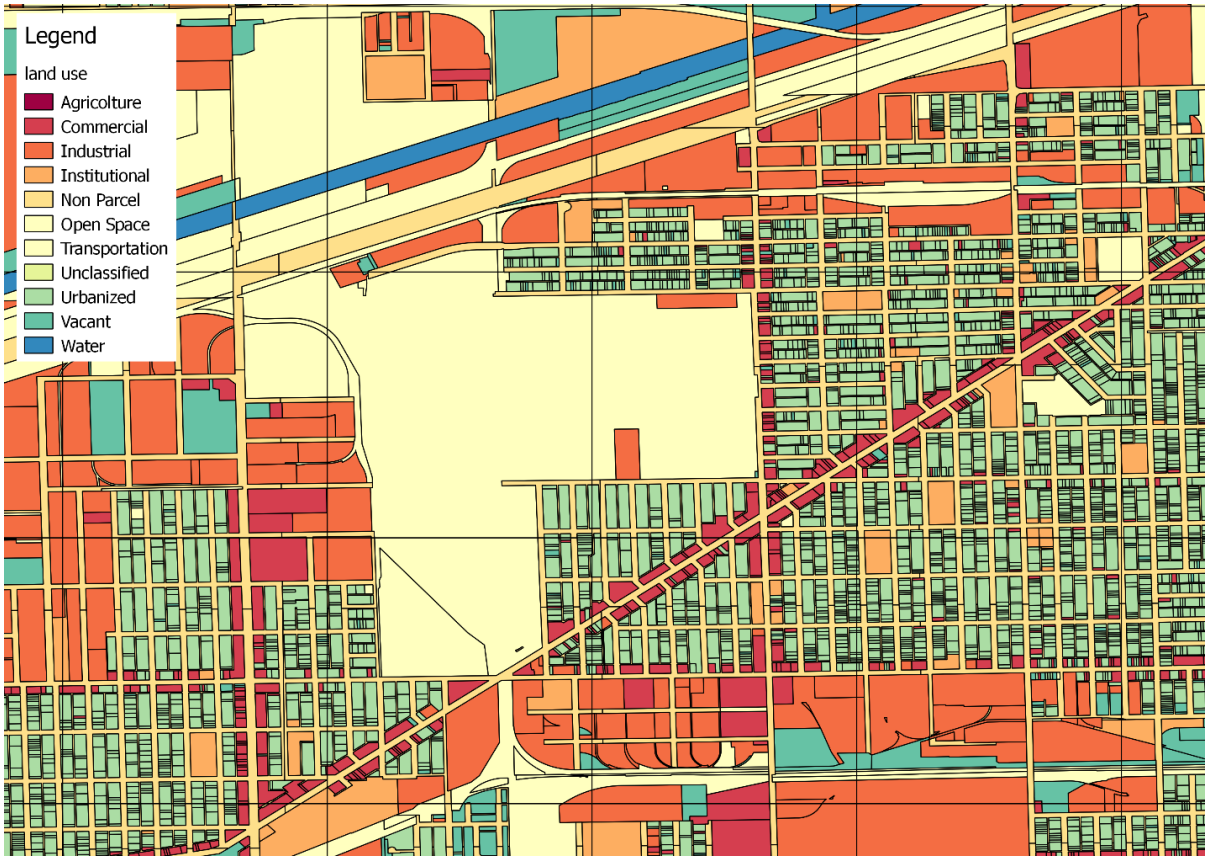

Figure 34: Land Use Data

The Traffic Data were retrieved from the Chicago Traffic Tracker [71]. By using the GPS traces retrieved from the CTA buses, real time hourly traffic congestion on Chicago' s arterial streets was estimated by the Chicago Traffic Tracker. Two or three community areas with similar traffic patters are typically grouped together to form a region . In total, the Chicago Traffic Tracker created 29 regions to cover the entire city (with the exception of the OHare

airport area).

Each grid cell of 1km x 1km will be assigned to the corresponding traffic regions using a spatial intersection.

From these data 2 features were extracted for each grid cell: the current speed that is used to estimate the real-time estimated congestion level. Although expressed in miles per hour, this value is more a reflection of the congestion level in the region than it is indicative of the average raw speed vehicles are travelling within the region. For congestion advisory and traffic maps this value is compared to a 0-9, 10-14, 15-19, 20-24, and 25 over scale to display heavy, medium-heavy, medium, light and free flow conditions for the traffic region.

The second feature extracted is the bus count in a specific region during the last hour.

An example of a congestion estimate by region is presented in the following picture:

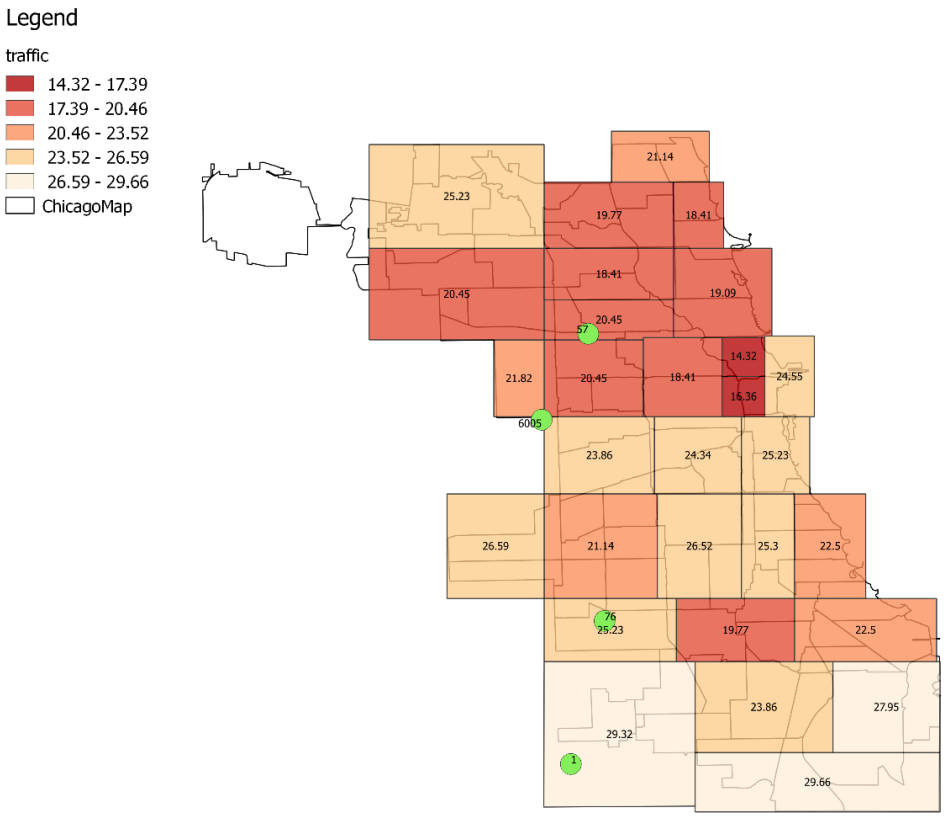

Figure 35: Traffic Regions

6.3.1 Haversine Distance

During the interpolation part, to properly calculate the distances between station and cells, we used the Haversine Distance to compute the distance between two points on Earth.

Since the earths shape can be approximated to a sphere, with a circumference of about 40,000 km, it is not possible to calculate distance between two points on sphere using the traditional

Euclidean Distance (by considering the Earth as flat), because that will introduce errors in the distance calculation.

The Haversine Distance use Latitude and Longitude to calculate the distance between two points on a spherical shape. This formula uses the spherical law of cosines.

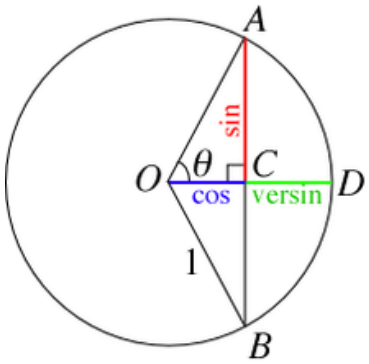

Figure 36: Haversine Distance

The Haversine formula is the following:

$$d = 2r\, arcsin(\sqrt{sin^2(\frac{\phi_2 - \phi_1}{2}) + cos(\phi_1)cos(\phi_2)sin^2(\frac{\lambda_2 - \lambda_1}{2})})$$

where ϕ_2 and ϕ_1 are the Latitude of point 2 and Latitude of point 1, while λ_2 and λ_1 are the Longitude of point 2 and Longitude of point 1.

6.4 Evaluation

The results of the experiment will be evaluated in term of Accuracy, Precision, Recall and F1 score. F1 score is an evaluation metric obtained by the combination of precision and recall. F1 Score is needed when you want to seek a balance between Precision and Recall. The formula of the F1 score is:

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Figure 37: F1 Score

We first conduct the first part of the training and the evaluation using only the 4 labeled cell and comparing the result with the corresponding ground truth, by removing one station and using the other 3 stations to infer its value. In total we will have 4 rows for each hour, that correspond to the 4 possible triplets obtained by removing one station.

After this step, we apply the pseudo label procedure to obtain a better performance on the unlabeled cell. We add the most confidently classified examples to our training-set and we retrain the neural network with more labeled examples. We continue this iterative process as long as there are unlabeled examples that satisfy the threshold condition of 0.99 and the evaluation metrics continue to improve. In case the threshold condition is satisfied but the

evaluation metrics dont improve, we stop the training and use the model weights of the previous iteration, since the model was starting to overfit. We evaluate again the previous metrics (Accuracy, , F1 score) on the new data and at the end we will apply our trained model to label all the other unlabeled cells.

CHAPTER 7

DATA PREPARATION

7.1 Basic Data Preparation

The hourly ground measurements of PM $_{2.5}$ from the 4 monitoring stations were retrieved and combined with the hourly weather conditions data that consist of temperature, humidity, pressure, wind speed, wind direction and weather description.

The weather description was in a string format, for example ‘light rain‘, ‘broken clouds‘, ‘sky is clear‘ and other weather conditions. To use these data, we apply a one hot encoding to convert the categorical values in numerical value. Then we applied a linear interpolation to fill the None values of Humidity and Sample Measurement.

We then divided the data in 4 different datasets, one for each station, since we used 4 Deep Bidirectional LSTM neural networks.

Additionally, we added the $PM_{2.5}$ measurement of the closest station to each dataset since from our analysis the data of the closest station showed a high correlation with the data of the current station considered. The data were split in training and test. In particular, an 80-20 holdout was used, with 80% of the dataset used for training and the remaining 20% for test. A 10% validation set was extracted by the training set and used to tune the hyperparameters.

7.2 LSTM Data Preparation

This stage involved the transformation of the dataset into a supervised learning problem and the normalization of the data. The main objective of this study is to forecast the pollution concentration of the next hour (or the next hours) using the data of the previous n hours where n represents the number of lags used by our model. The choice of this number will be explained in the result section.

By transforming the dataset into a supervised learning problem, the $PM_{2.5}$ measurement of the current timestamp will become the target that we want to predict while the previous n-hours PM2.5 measurement and weather conditions will be used as predictor variables.

We then applied a scaling to normalize the data for our deep neural network model. The main reason behind this normalization, should be searched in the necessity of having all the values in the same range, so that when we fit the network on the input there won't be very large values that can slow down the convergence and the learning process. In the worst case, large values can generate wrong results by preventing the network to effectively learn the problem. [72].

The data were scaled in a range between 0 and 1 before training the model. This scaling was inverted before the calculation of the RMSE to have consistent results (in the same range of the original dataset).

CHAPTER 8

FORECASTING RESULTS AND DISCUSSION

We will now discuss more in detail the hyperparameter's choice for the different type of Neural Networks that we compared in this study.

Two different approaches were tested:

- One model for each ground monitoring station (4 models in total)
- One single model for all the 4 stations with multiple output, one output for each station

The two approaches tested, presented different pros and cons that we will analyze in the following sections.

8.1 Multiple Models Deep Bidirectional LSTM

In this first approach we consider each ground station as independent from each other and we created four different models, one for each monitoring station. Each model present different characteristics, in terms of hyperparameters and results.

Different configurations were tested for the multiple models Bidirectional LSTM. Due to the randomness in the weights initialization of the neural network, the model produced different results at each run. To solve this issue, we first used a random seed to have reproducible results and tune the model. For the final testing part we removed the random seed and we averaged the results over 5 runs.

We also implemented an early stopping method with patience that allows us to train the model for a big number of epochs (1000 in our experiment).

The early stopping method with patience, allows the training of the neural network to be stopped, if after a fixed number of epochs(patience), no improvements in the loss value were registered. The early stopping methods save the weights of the best model found and then use them to make predictions.

A very important part of the implementation of this model consisted in the input data preparation, for more details see Chapter 6.

We tried lot of different combinations of hyperparameters, including different number of layers, neurons, activation functions and number of epochs.

After repeated experiment the best configuration found were the following:

Station 1

- 1 Bidirectional LSTM hidden layer with 5 neurons in each direction (10 in total)
- Dropout of 0.2
- 1 Bidirectional LSTM hidden layer with 3 neurons in each direction (6 in total)
- Dropout of 0.2
- 1 Fully Connected layer with 5 neurons
- 1 Fully Connected layer with 5 neurons
- 1 Fully Connected output layer with 1 neuron

Station 57

- 1 Bidirectional LSTM hidden layer with 20 neurons in each direction (40 in total)
- Dropout of 0.2
- 1 Bidirectional LSTM hidden layer with 10 neurons in each direction (20 in total)
- Dropout of 0.2
- 1 Fully Connected layer with 10 neurons
- 1 Fully Connected layer with 10 neurons
- 1 Fully Connected output layer with 1 neuron

Station 76

- 1 Bidirectional LSTM hidden layer with 12 neurons in each direction (24 in total)
- Dropout of 0.2
- 1 Bidirectional LSTM hidden layer with 7 neurons in each direction (14 in total)
- Dropout of 0.2
- 1 Fully Connected layer with 12 neurons
- 1 Fully Connected layer with 12 neurons
- 1 Fully Connected output layer with 1 neuron

Station 6005

- 1 Bidirectional LSTM hidden layer with 12 neurons in each direction (24 in total)

- Dropout of 0.2
- 1 Bidirectional LSTM hidden layer with 7 neurons in each direction (14 in total)
- Dropout of 0.2
- 1 Fully Connected layer with 12 neurons
- 1 Fully Connected layer with 12 neurons
- 1 Fully Connected output layer with 1 neuron

The choice of using this kind of configurations have been inspired by some other deep neural network configurations present in the literature and used in some application that were not directly related to Time Series Forecasting.
In our deep neural network, we used a min-batch approach, in particular the batch size chosen was 96, corresponding to 4 days. This number was chosen after repeated experiments with different batch size. The optimizer used was the Nestorov Adam and the loss function chose was the MSE.

The activation functions applied were tanh for the Bidirectional LSTM and linear for the Fully Connected. We also applied a Dropout of with a rate of 0.2 to avoid overfitting.

One of the most important parameters to select was the number of lags, so the number of previous timestamps to use in our model. The final number of lags used was 24, that means that our model predicted the pollution concentration at a given timestamp for every input using the previous 24 hours data.

The choice of the time lags is very important, in fact a small number of lags (for example 1) doesnt allow the neural network to exploit the power of the LSTM architecture. On the contrary a big number can confuse the neural network since observation that are too far from a temporal point of view are used. From our experiment 24 can reach a good compromise between these two extremes.

The final RMSE of each station were the following:

TABLE II: RMSE BDLSTM Multiple Models

	Best RMSE			
	Station 1	Station 76	Station 57	Station 6005
RMSE	2.928	3.422	4.009	3.291

By doing the average of the RMSE of each station we obtained an overall RMSE of **3.4155**.

The following figure show the predicted pollutant concentration over the next hour and real value data for station 1, used as test set to calculate the model performance :

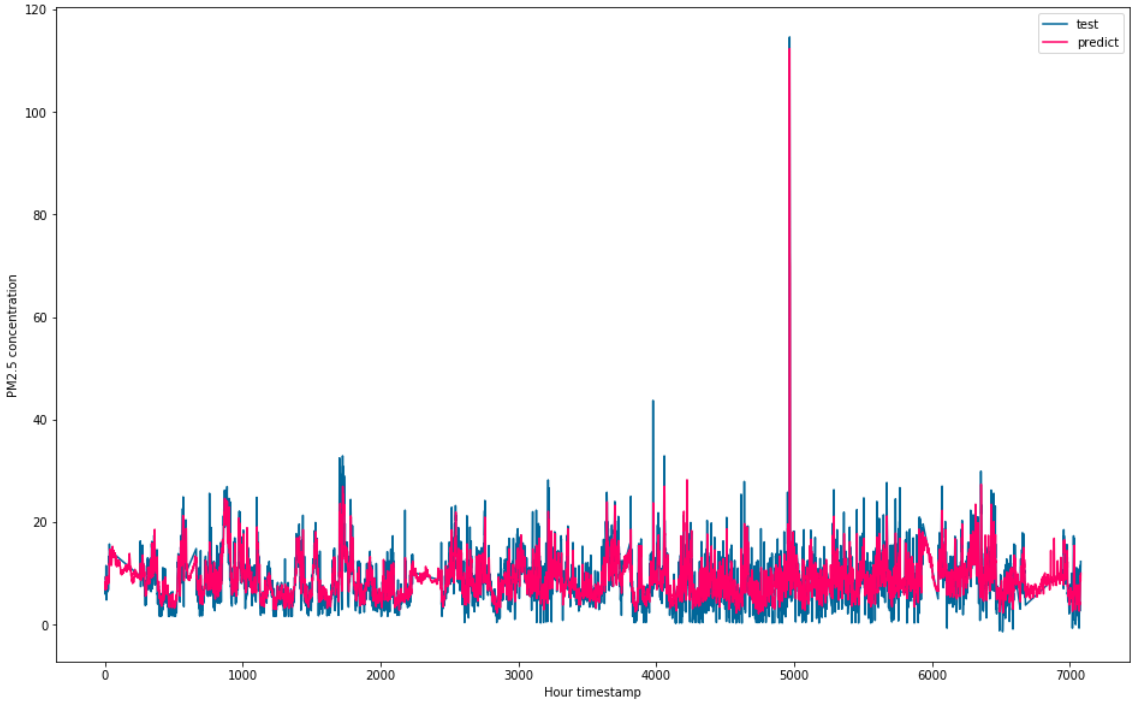

Figure 38: Predicted vs Real Data

Since the number of data predicted is very high, it is not easy to visualize all of them in a single plot.

For this reason, to better understand how the predicted values compare to the test values, we can take a look to a zoomed view of the previous graph that is more readable and easy to understand since only 100 timestamps are shown.

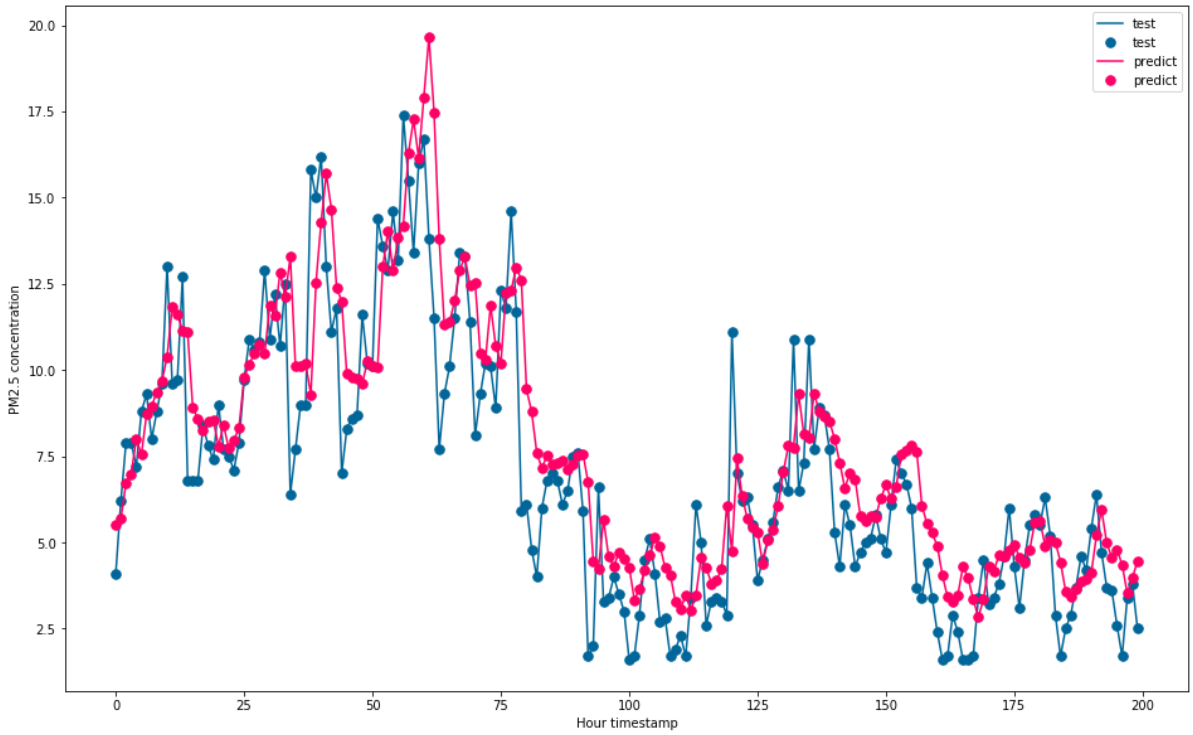

Figure 39: Zoom of Predicted vs Real Data

8.2 Deep Bidirectional LSTM Single Model

In the second approach proposed, we considered a single dataset for all the stations, with 4 additional columns representing the pollution concentration at each station.

The model used in this case, was different from the previous one, because the network was a multi-input and multi-output model, in which each output represented the predicted concentration of a specific station.

As in the previous approach, several network configurations were tested, and the best architecture found was the following:

All Stations

- 1 Bidirectional LSTM hidden layer with 10 neurons in each direction (20 in total)
- Dropout of 0.2
- 1 Bidirectional LSTM hidden layer with 20 neurons in each direction (40 in total)
- Dropout of 0.2
- 1 Fully Connected layer with 20 neurons
- 1 Fully Connected layer with 20 neurons
- 1 Fully Connected output layer of 4 neurons

The final RMSE calculated over the 4 stations was **3.631**.

The following table shows a comparison between the multiple stations model and the single station model:

TABLE III: Multiple Model vs Single Model

	Best RMSE				
	Station 1	Station 76	Station 57	Station 6005	Overall Result
Multiple models	**2.928**	**3.422**	**4.009**	**3.291**	**3.412**
Single model	3.145	3.687	4.13	3.558	3.631

8.3 Multi-Layer Perceptron

In this case we tested lot of different combinations of neurons and layers. We fixed the time lags to 24 and we use all the other parameter used for the LSTM model.

The neural network was trained for 1000 epochs using the early stopping method.

The results of the experiments can be summarized in the following tables:

TABLE IV: RMSE MLP Station 1

	Number of Neurons					
Layers	5	10	20	50	100	200
1 layer	3.080	3.155	3.121	3.075	3.161	3.17
2 layers	3.076	3.256	3.113	3.151	**3.074**	3.096
3 layers	3.108	3.116	3.087	3.111	3.079	3.089

TABLE V: RMSE MLP Station 76

	Number of Neurons					
Layers	5	10	20	50	100	200
1 layer	3.798	3.842	3.837	3.895	3.671	3.673
2 layers	3.848	4.002	3.960	3.710	3.711	3.669
3 layers	3.768	3.746	3.745	3.685	3.669	3.644

TABLE VI: RMSE MLP Station 57

	Number of Neurons					
Layers	5	10	20	50	100	200
1 layer	4.054	4.034	4.051	4.043	4.014	4.023
2 layers	4.014	4.050	4.016	4.030	4.019	4.008
3 layers	4.012	4.028	4.008	4.033	4.010	4.008

TABLE VII: RMSE MLP Station 6005

	Number of Neurons					
Layers	5	10	20	50	100	200
1 layer	3.371	4.376	3.391	3.440	4.412	3.358
2 layers	3.418	3.543	3.462	3.515	**3.347**	3.508
3 layers	3.812	3.732	3.561	3.679	3.510	3.356

8.4 Long Short-Term Memory Neural Network (LSTM)

We tested different configurations of LSTM, in particular only some of them shows results similar to the MLP but the majority of them are worse in term of RMSE.

The configuration tested for each station were the same in terms of number of neurons and layers of the Bidirectional model but without the fully connected layers.

In this case the number of lags used was fixed to 24 as in our proposed method. All the other hyperparameters used were identical to the Deep Stacked Bidirectional LSTM.

TABLE VIII: RMSE LSTM

	Best RMSE			
	Station 1	Station 76	Station 57	Station 6005
RMSE	3.131	3.617	4.1	3.342

8.5 Random Forest

The Random Forest Algorithm was tested, using a window size of 24 like in the case of the Multilayer Perceptron and 200 trees:

TABLE IX: RMSE Random Forest

	Best RMSE			
	Station 1	Station 76	Station 57	Station 6005
RMSE	3.258	3.810	4.097	3.334

8.6 General Regression Neural Network

In this case the choice of parameters is much more limited with respected to the previous example. In fact, General Regression Neural Networks have only one free parameter, the smoothness.

The search for the smoothness parameter may vary depending on the application. In this study we will choose a set of smoothing parameters and we will evaluate the results in term of RMSE. This neural network falls into the category of probabilistic neural networks and its single-pass learning so no backpropagation is required.

The following table show the results of this technique:

TABLE X: RMSE GRNN

	Smoothness Parameter					
	0.1	1	1.5	2	5	10
Station 1	Nan	Nan	**5.1216**	5.18	5.268	5.284
Station 76	Nan	Nan	**6.534**	6.587	6.671	6.687
Station 6005	Nan	Nan	**5.616**	5.623	5.634	5.645
Station 57	Nan	Nan	**6.553**	5.768	5.784	5.892

8.7 Persistence

In Machine Learning, the most common baseline for supervised problems is the Zero Rule algorithm.

In the case of classification, this algorithm predicts the majority class, while in the case of regression the average outcome . In Time Series Forecasting problems this rule cannot be applied because of the constraint of the time series dataset.

For this reason, the equivalent of the Zero Rule for Time Series Forecasting is the so called, persistence model.

The persistence algorithm works in a very simple way. It predict the expected outcome of the future time stamp (t+1), just by forwarding the previous value at time (t). It is basically a shifting process of the previous time stamp value.

A persistence model looks like this:

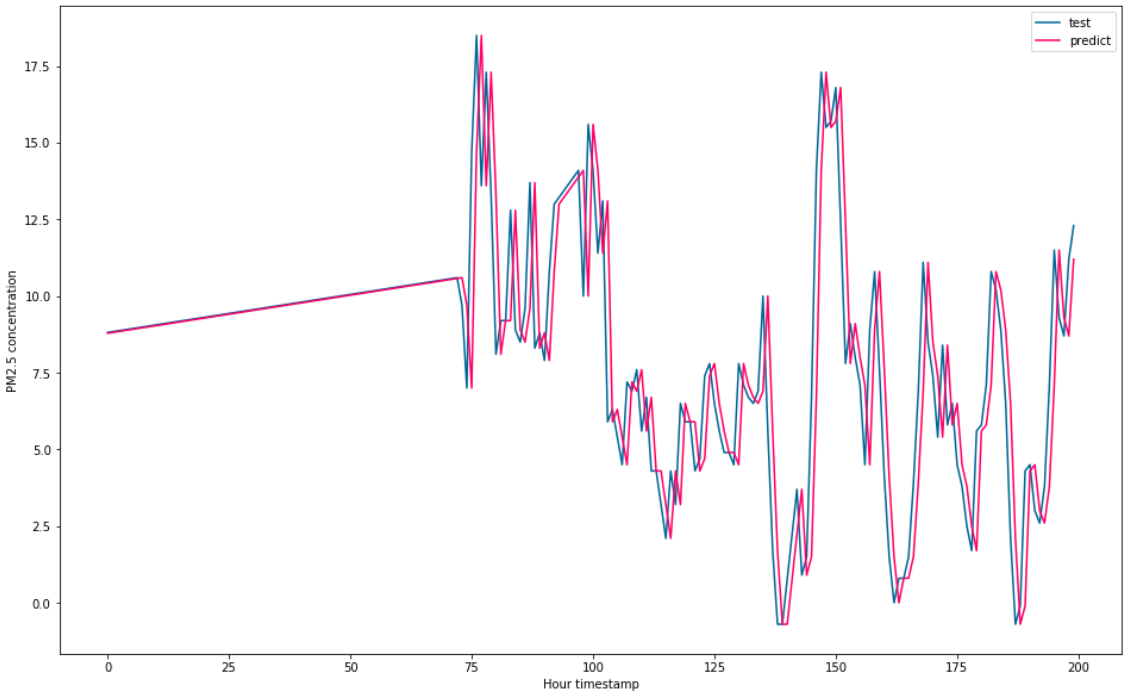

Figure 40: Predicted vs Real Data using Persistence

It is possible to observe that the predicted values are exactly a version of the test values shifted by 1 timestamp.

The RMSE of this model is shown in the following table:

TABLE XI: RMSE Persistence

	Best RMSE			
	Station 1	Station 76	Station 57	Station 6005
RMSE	4.083	4.118	4.341	3.928

8.8 Multi Step Ahead Forecasting Results

For the multiple step ahead forecasting, we selected the best model configuration with respect to the single hour prediction.

The number of timestamps to predict in the future has been set to 4 and the approach used was the **Recursive Multi-Stage Approach** described at the end of section 3.4.4.

TABLE XII: RMSE Multi Step Ahead

	Future Hours Predictions			
	1 hour	**2 hours**	**3 hours**	**4 hours**
Station 1	2.928	3.598	4.087	4.383
Station 76	3.422	4.797	5.696	6.345
Station 6005	3.291	4.154	4.859	5.469
Station 57	4.009	4.768	5.271	5.520

A common way to visualize the multi-step ahead forecasting, is to plot the results with a padding equal to the number of future timestamps predicted.

For example we can plot the first timestamp prediction starting at time t + 1, and then for the following N timestamps, where N represents the number of predicted steps in the future, we plot the value of t+2, t+3 t+N.

Then we will plot the next prediction with a padding of N, so starting at timestamp t+N +1.

To better understand this concept, we can take a look at this figure, where we use a padding of 4 to plot our predictions, since we were predicting 4 steps in the future:

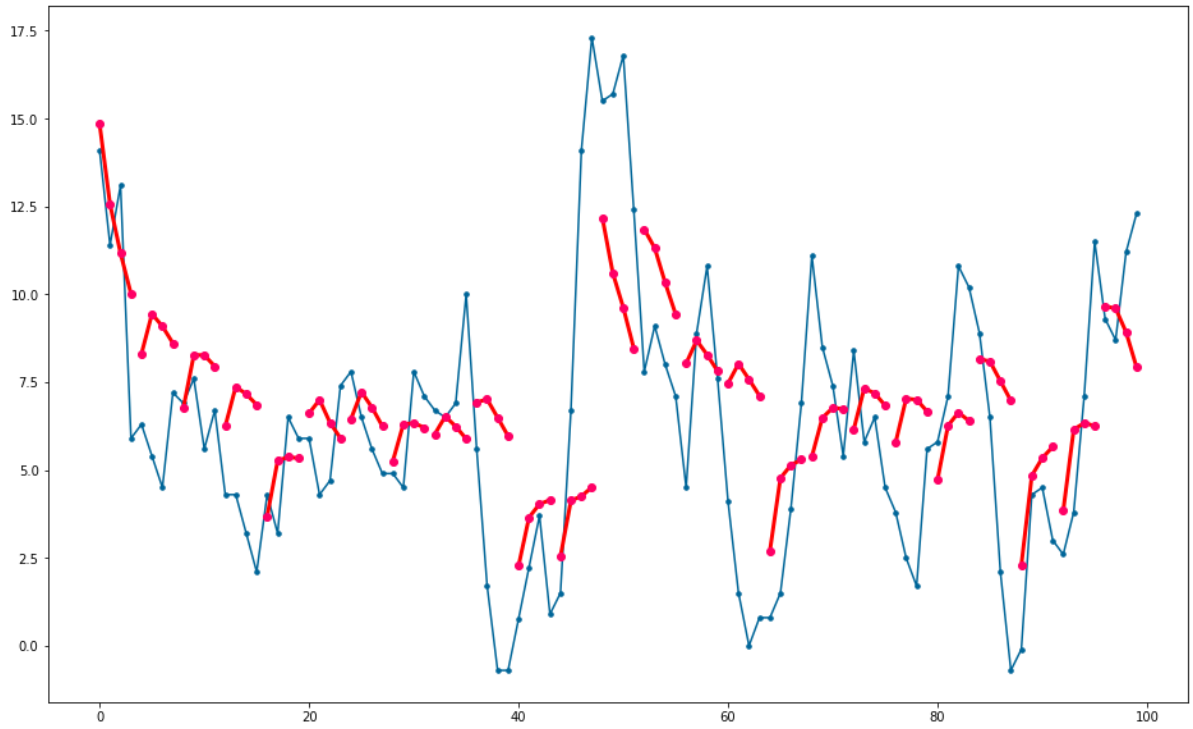

Figure 41: Multi Step Ahead Forecasting

In the figure we can see an example of multi-step ahead forecasting for Station 1. By looking at the plot we can notice that our model does a good job in predicting the future direction of the prediction, since we can notice that the correct future direction is predicted for almost all the predictions.

8.9 Discussion

The previous results show that the proposed method outperform the other methods in term of RMSE. In particular, General Regression Neural Network seems to be not suitable for this

kind of problem since the RMSE never falls down 5.

The Multilayer Perceptron with a large window (24 time lags) that is usually used as state of the art for autoregression problems, shows good predictive performance. Autoregression problems are problems in which the future time stamp can be expressed as a function of a specific number of previous (or lag) observations.

The proposed Deep Stacked Bidirectional LSTM shows a good predictive power reaching a best RMSE for station 1, 76 and 6005. In particular, it seems it can follow the signal and has predictive skills. If we zoom on a section of the plot of the results, we can notice that even where there is a linear interpolation our model doesnt simply follow a straight line like in the case of the persistence model but oscillate.

This is a sign that our model is not just predicting using the previous measurement value in time but have learned some patterns from the data.

The model on the left represent what is predicted by our recurrent neural network while the one on the right what is predicted by the persistence model in case of linearly interpolated values.

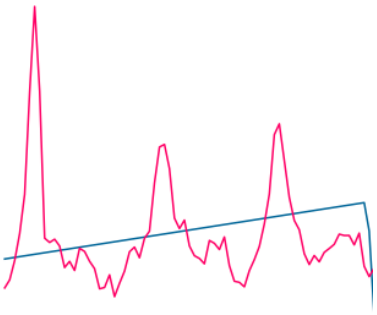

Figure 42: Result of Proposed Model

Figure 43: Result of Persistence Model

In conclusion even if from the previous figures the model on the right seems to be better because it follows the original values, this is not true because that model is simply following a straight linear interpolation line, so the persistence model doesnt ignore them but it follows them. From our results we can conclude that our framework contradicts some statements present in the literature :

"Time series benchmark problems found in the literature are often conceptually simpler than many tasks already solved by LSTM. They often do not require RNNs at all, because all relevant information about the next event is conveyed by a few recent events contained within a small time window." [73]

In this paper LSTMs are applied to 2 Time Series Forecasting problems and the final results are compared with different neural network architectures. In this paper, the results show that the Multilayer Perceptron (MLP) based on a large window size is capable to outperform the LSTM model on some Time Series problems, that can be easily solved just by using few recent data.

Our work instead shows that the Deep Bidirectional LSTM is not just able to follow the general trend of the data, but it also outperforms the Multilayer Perceptron.
This table summarize all the results obtained:

TABLE XIII: RMSE Final Comparison

	Best RMSE			
	Station 1	Station 76	Station 57	Station 6005
DBLSTM	**2.928**	**3.422**	4.009	**3.291**
LSTM	3.131	3.617	4.1	3.342
MLP	3.074	3.644	**4.008**	3.347
RF	3.258	3.810	4.097	3.334
Persistence	4.083	4.118	4.341	3.928
GRNN	5.126	6.534	6.553	5.616

CHAPTER 9

INTERPOLATION RESULTS AND DISCUSSION

The algorithms were tested on unseen data, in particular the test data were randomly selected using a holdout method 80-20, where the testing data correspond to the 20% of the total data available. In total 19277 instances were present in the test set.

9.1 Inverse Distance Weighting Interpolation (IDW) Results

For the Inverse Distance Weighting Interpolation, the only features used were the distances and the pollution levels of the 1st, 2nd and 3rd closest stations. In this case the only parameter is the value of the power parameter p. Different values of p were tested to see the effects of the power parameter on the evaluation metrics.

TABLE XIV: IDW Interpolation Results

	IDW			
	p = 1	p = 2	p = 3	p = 4
RMSE	**4.976**	5.178	5.378	5.490
Accuracy	**0.80**	0.79	0.78	0.78
Precision	**0.80**	0.79	0.79	0.78
Recall	**0.80**	0.79	0.79	0.78
F1	**0.80**	0.79	0.79	0.78

The results obtained from the IDW interpolation were also converted to AQI label using the EPA table, so that it was possible to calculate also the Accuracy, Precision, Recall, F1 and not only the RMSE.The results suggested that the best value for the power parameter in this case is 1, with an RMSE of 4.976, an accuracy and F1 score 0.80.

The plot in Figure 44 show the difference between the inference of the IDW with different values of p versus the original pollution value (in blue) over a period of 30 hours taken from the shuffled test set. . On the x-axis we have the hours and y-axis the pollution values.

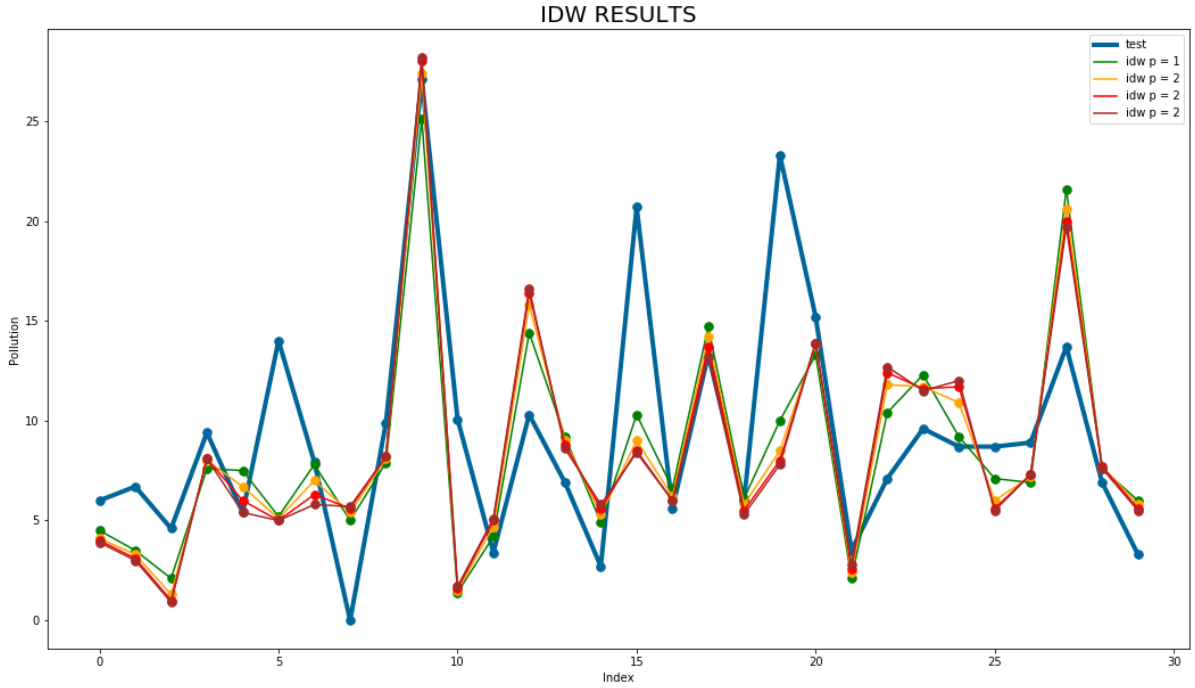

Figure 44: IDW Results

In Figure 45 we can see an example of a pollution map created after applying Inverse Distance Weighted Interpolation on the 03.17.2017 at 21:00:00.

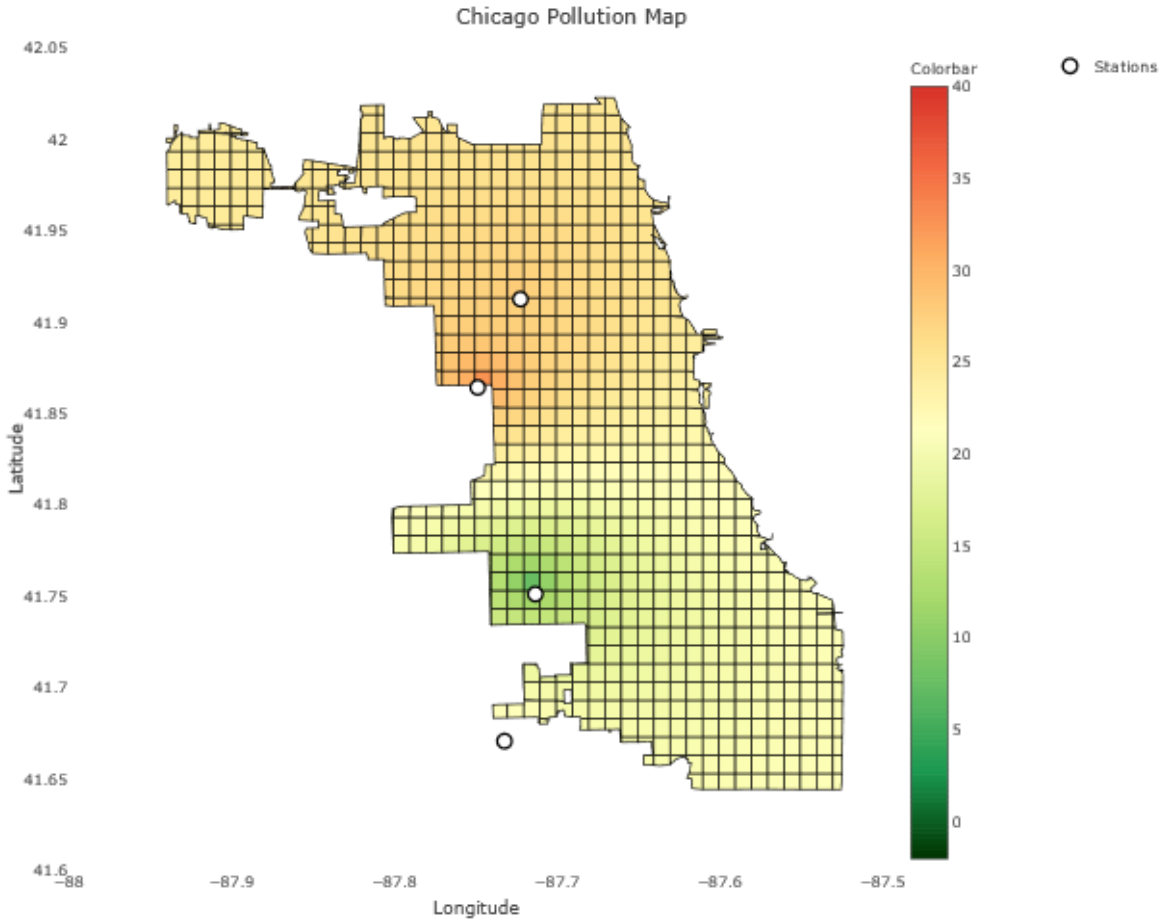

Figure 45: IDW Pollution Level Interpolated Map

The map has been created using the plotly library, in order to have interactive maps. This is a great advantage since it is possible to produce these maps in few second just after the inference part and to visualize them directly in python without the use of any additional software.

9.2 Neural Network Results

The neural network used was a single model that used both spatial and temporal features. The model has 3 layers with Relu activation function, a Softmax on the output layer of the classification branch to output the 6 classes probabilities and a Linear activation function on the output layer of the regression branch to output the precise value of pollution. "The Relu (Rectified Linear Unit is one of the most commonly used activation function in Deep Learning models. The function returns 0 if it receives any negative input, but for any positive value it returns that value back. So it can be written as f(x)=max(0,x))".

The results obtained over the 4 stations are the following:

TABLE XV: Neural Network Results

	Neural Network	**Neural Network Self-Training**
RMSE	4.372	**4.246**
Accuracy	0.83	**0.88**
Precision	0.83	**0.88**
Recall	0.84	**0.88**
F1	0.83	**0.88**

An example of Air Pollution map produced on the same day 03.17.2017 at 21:00:00 using the Neural Network inference is presented in Figure 46.

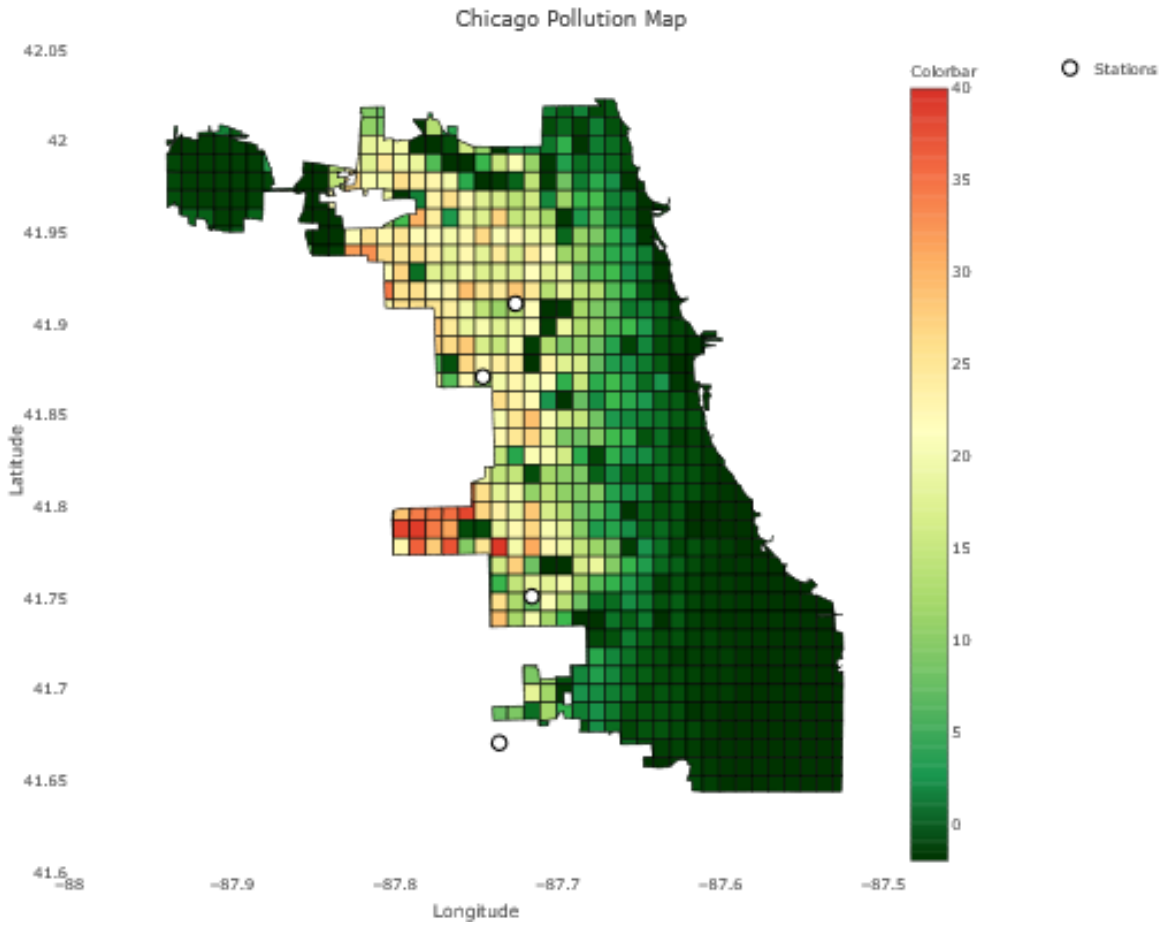

Figure 46: Self Training Pollution Level Interpolated Map

Using the Self Training results we obtained also the following map(Figure 47) for the same day with the AQI level of concern instead of the pollution values.

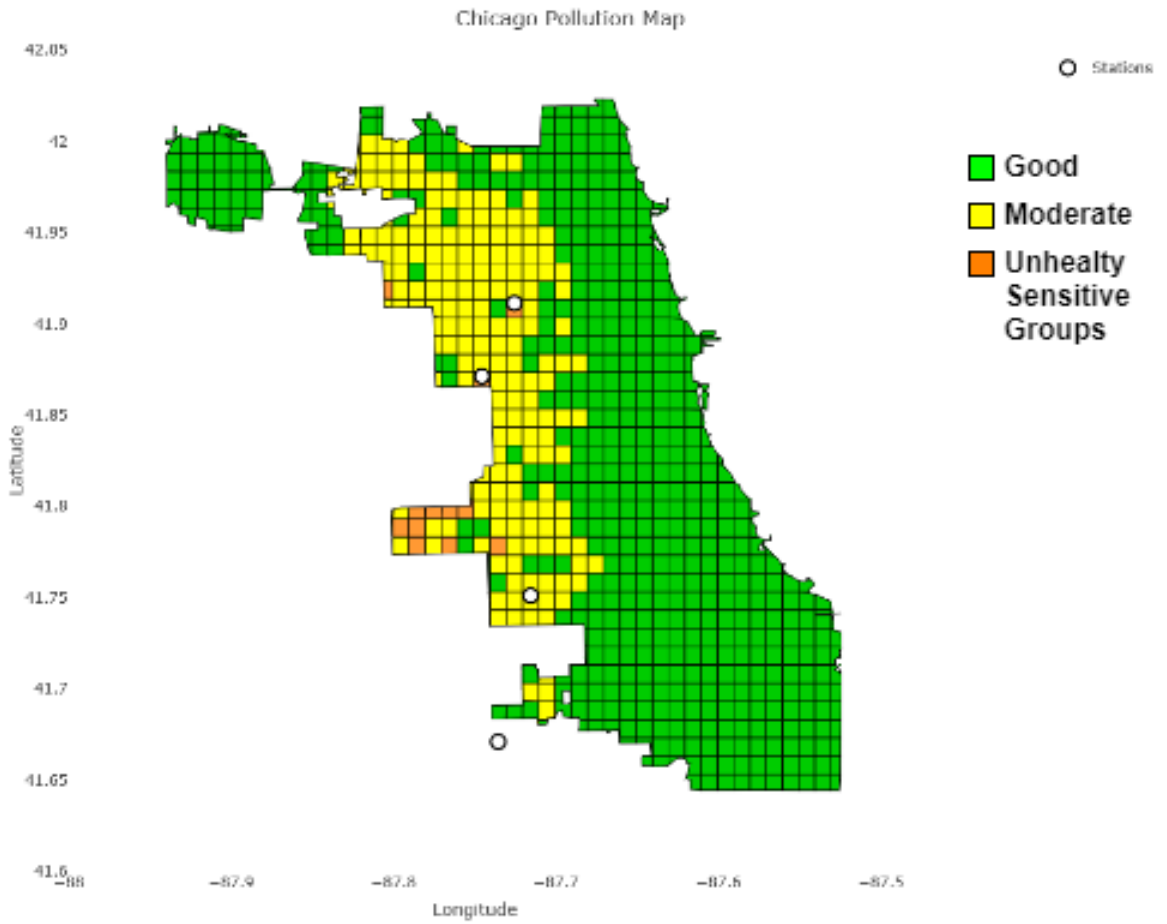

Figure 47: Self Training AQI Label Interpolated Map

9.3 Co-Training Results

The Co-Training algorithm used in the U-Air [35] paper was tested using the same models and configuration presented. The only difference was in the type of data used, due to the different setting of the experiment and data availability. We tryed to use the same approach and divide the features into sets: the temporal features were feed into a CRF (Conditional Random Field) and the spatial features were feed into a Neural Network.

We also proposed an optimized version of the Co-Training algorithm used an LSTM as temporal classifier and an Artificial Neural Network as spatial classifier. The Artificial Neural Network was used also in the U-Air study[35] but with a better hyperparameter optimization and network structure (3 layers instead of 1), since in the U-Air experiment the Neural Network used has only one hidden layer. In this way the comparison between the different methodology was as fair as possible.

We also want to highlight that for the Co-Training algorithm we dont have the RMSE results, since we used the same implementation of the original paper in which the problem is seen only as a classification problem.

The results obtained are the following:

TABLE XVI: Co-Training Results

	Co-Training (U-Air)	**Co-Training (Optimized)**
Accuracy	0.82	0.86
Precision	0.82	0.85
Recall	0.82	0.86
F1	0.82	0.86

The map presented below show the interpolation map created using the Co-Training framework. It is possible to notice that in this case only the 6 possible labels have been used, since we dont have the precise level of pollution of each cell.

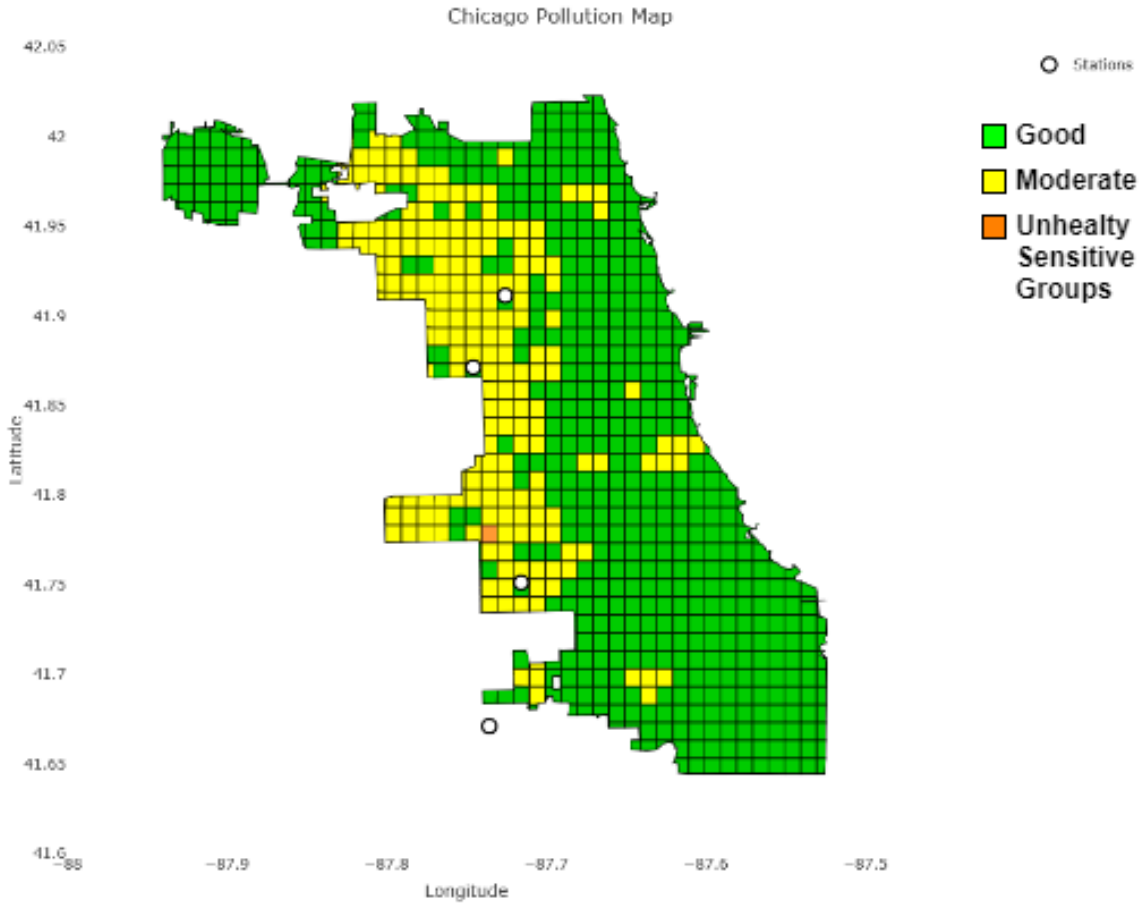

Figure 48: Co-Training AQI Label Interpolated Map

9.4 Discussion

The numerical results show that the proposed Neural Network model, with Self-Training can produce better numerical results than the other baseline used for comparison. In particular, the standard Inverse Distance Weighted interpolation doesnt give good performance, since the

interpolation is based just on the distance and on the pollution values.

The Co-Training algorithm presented in the U-Air paper was reproduced using the same network structure described. The results are comparable to Neural Network model without Self-Training and outperforms the IDW interpolation. By optimizing the hyperparameters and the network structure of the Co-Training model it was possible to obtain better results.

A final comparison between our model and the optimized Co-training framework shows that our model is able to outperform the other methodology. Besides our model give us the possibility to predict both the level of concern and the precise pollution value of each cell, while this is not possible using the Co-Training framework. This represent a great advantage since it is possible to produce maps with much more details, since each cell has is own pollution value, inferred using Land Use Characteristics, Traffic Patterns, Meteorological Data and the Pollution values.

A comparison of the maps produced by the different algorithm is presented below:

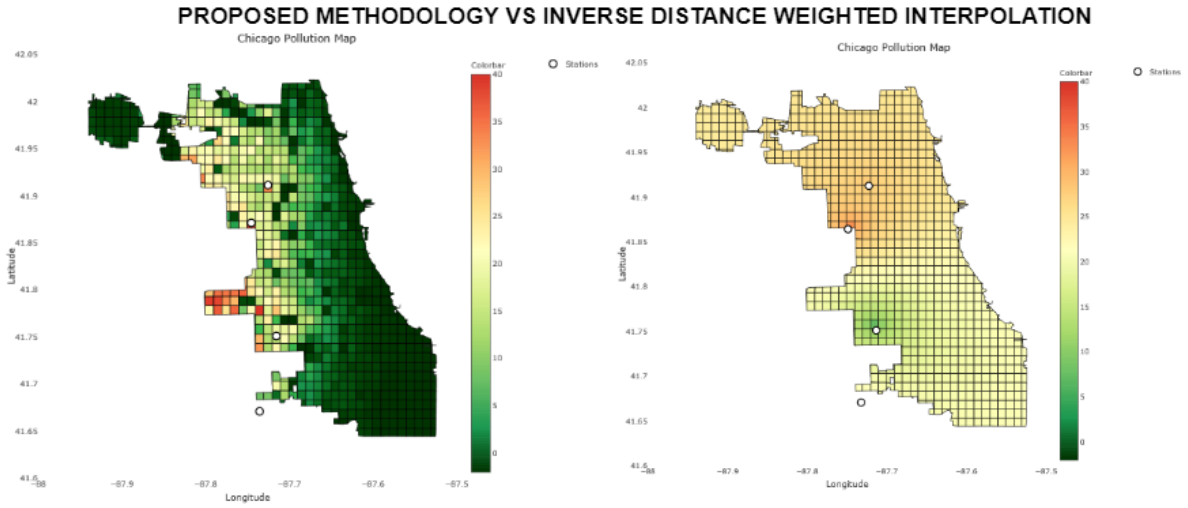

Figure 49: Self-Training VS IDW Pollution Level Interpolated Maps

From this comparison we can notice how the map produced by our model dont follow any linear pattern like in the case of IDW interpolation. The two maps refer to the same day and hour and they show big differences.

Even if it is not possible to compare the pollution level map between our methodology and the co-training model, we will make a comparison based on the AQI labels. For this comparison we can notice that the two maps are not so different, the main reason is that each label corresponds to a wide range of pollution value, so it is not so common to have very different label for each grid cell.

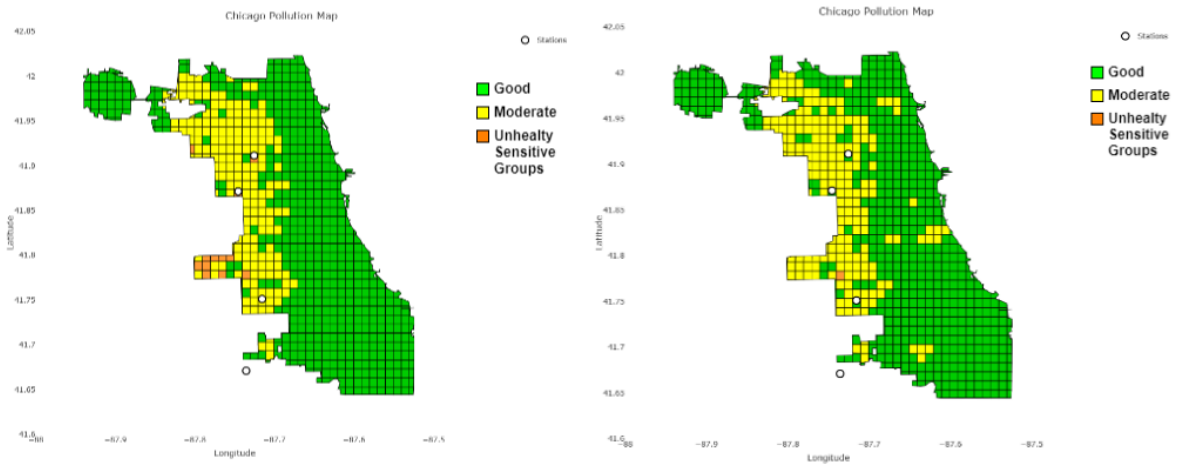

Figure 50: Self-Training VS Co-Training AQI Labels Interpolated Maps

Finally, we can compare the pollution map generated by our model with the map present on the AirNow archive website [74]. Even if the two maps have a different spatial resolution, since our model use a fine-grained resolution of 1km x 1km which is much smaller than the one used by AirNow right now, we can still show a comparison to highlights the difference between them and the advantages of our methodology.

In Figure 51 we can observe the map produced by the Air Now website on the same day 03.17.2017 of the other maps shown previously. As we can notice, the whole Chicago area has a unique label (Moderate = yellow). The main reason is that AirNow, as explained before, use a much bigger spatial resolution and dont consider most of the factors we used in our air quality inference, like Traffic pattern, Land Use and Meteorological factors.

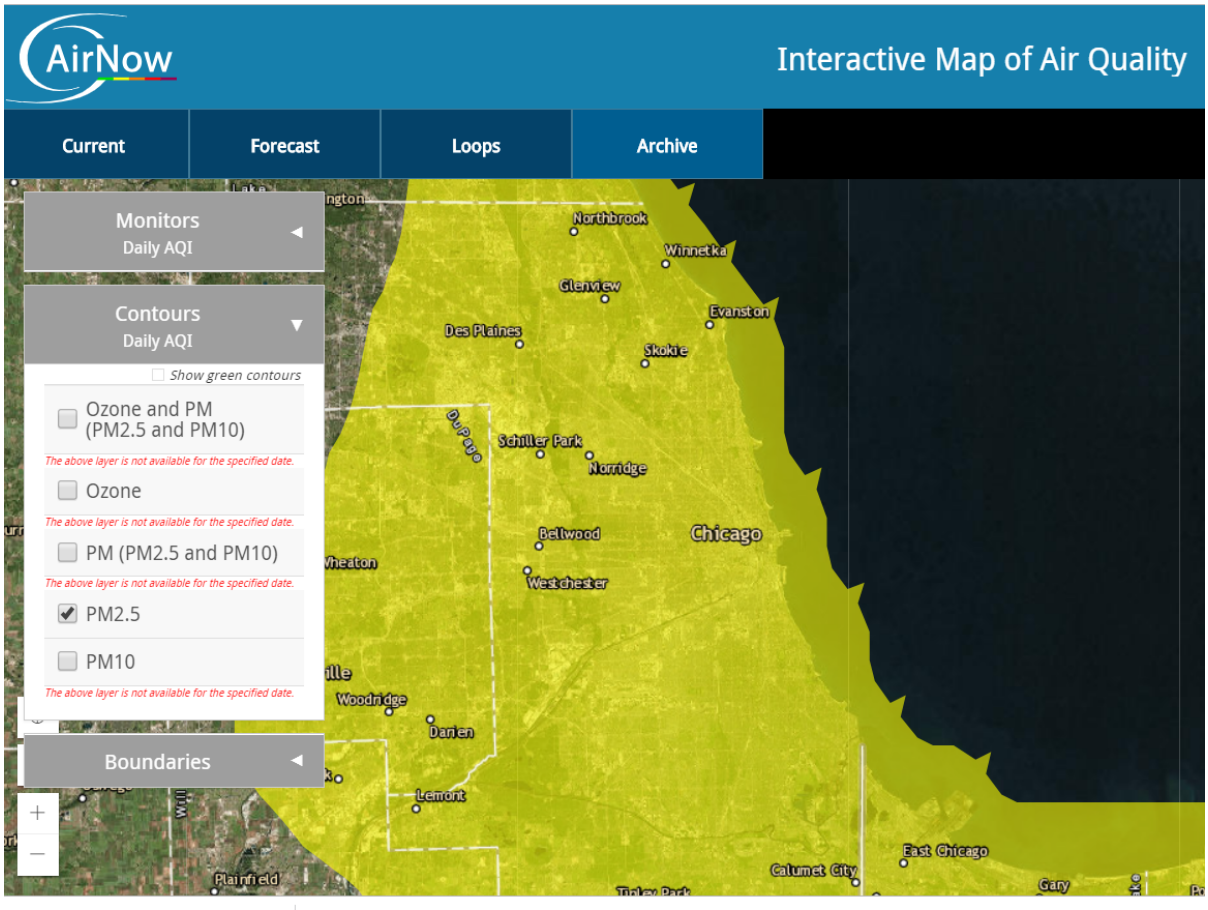

Figure 51: AirNow Maps

In conclusion, we believe that our model can provide good results in term of classification metrics and RMSE. Besides it provides the possibility of creating very detailed maps of the city of Chicago, with a much higher level of details and precision, with respected to the other models

presented in this study.The comparison with other baselines and the AirNow maps, shows the advantages of the proposed methodology both for the numerical and visual results.

CHAPTER 10

CONCLUSION

10.1 Limitations

As we can see from the numerical results obtained and the visualization of the predicted values in comparison with the real values, the model seems to do a very good job in predicting the next hour concentration and in general in following the general pattern.

By looking at the multi-step head forecasting, we can also notice that our model does a good job in predicting the future direction of the values. Nevertheless, the model is affected by some limitations.

In particular, by observing the graph of the predicted vs real values Figure 38, we can notice that even if the model is very good in reacting to smooth variations, on the contrary it cannot react to fast nonlinear changes(spikes).

This is a very common limitations in time Time Series Forecasting models, because often the future values are independent from previous values. In our case, these sudden variations can be due to external factors, other than meteorological conditions. For this reason, our model cannot take them into account.

A similar consideration can be done for the multi-step ahead forecasting where the unavoidable propagation of the error in the future predictions, lead to results that get worse with the increasing of future hours predicted.

Another limitation of this model is the computational complexity. In fact, using a different Neural Network model for each station, require more training time, since we have to train 4 different neural networks instead of a single one.

Finally, the last limitation relates to the interpolation part. Even if the model outperform the baseline in both classification and regression, the model doesn't show a good performance for the grid cell that are too far from the stations. Looking at the map we can notice that the part on the right of the map is almost uniform. More monitoring stations could drastically improve the results.

10.2 Conclusion

This study proposed a novel Bidirectional LSTM framework, to forecast and interpolate short term air pollution concentration in a city.

Numerical results on the Chicago dataset showed that taking advantage of the combination between meteorological data, pollution levels and the LSTM memory property our model was able to outperform the baseline including the Multi-Layer Perceptron, that is one of the most used techniques for this kind of task. In the future, we intend to improve this basic version of the framework in different ways.

We will try to add new data to our framework, that in this moment are not available or dont have the required temporal resolution, like satellite data and traffic data in order to improve both the prediction and interpolation results.

This will also imply further research in the hyperparameter optimization, such as batch size and number of LSTM cells, but also in finding the optimum time lags, that can help the model

to obtain a lower RMSE.

Finally, we can try to extend our model and try different techniques including CNN and the Attention mechanism that can further improve the model performance.

CITED LITERATURE

1. Anderson, J. O., Thundiyil, J. G., and Stolbach, A.: Clearing the air: a review of the effects of particulate matter air pollution on human health. Journal of Medical Toxicology, 8(2):166–175, 2012.

2. Jiang, X.-Q., Mei, X.-D., and Feng, D.: Air pollution and chronic airway diseases: what should people know and do? Journal of thoracic disease, 8(1):E31, 2016.

3. Li, X., Peng, L., Yao, X., Cui, S., Hu, Y., You, C., and Chi, T.: Long short-term memory neural network for air pollutant concentration predictions: Method development and evaluation. Environmental Pollution, 231:997–1004, 2017.

4. Chen, J., Lu, J., Avise, J. C., DaMassa, J. A., Kleeman, M. J., and Kaduwela, A. P.: Seasonal modeling of pm2. 5 in california's san joaquin valley. Atmospheric environment, 92:182–190, 2014.

5. Wang, Z., Maeda, T., Hayashi, M., Hsiao, L.-F., and Liu, K.-Y.: A nested air quality prediction modeling system for urban and regional scales: Application for high-ozone episode in taiwan. Water, Air, and Soil Pollution, 130(1-4):391–396, 2001.

6. Saide, P. E., Carmichael, G. R., Spak, S. N., Gallardo, L., Osses, A. E., Mena-Carrasco, M. A., and Pagowski, M.: Forecasting urban pm10 and pm2. 5 pollution episodes in very stable nocturnal conditions and complex terrain using wrf–chem co tracer model. Atmospheric Environment, 45(16):2769–2780, 2011.

7. Li, C., Hsu, N. C., and Tsay, S.-C.: A study on the potential applications of satellite data in air quality monitoring and forecasting. Atmospheric Environment, 45(22):3663–3675, 2011.

8. Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M.: Time series analysis: forecasting and control. John Wiley & Sons, 2015.

9. Nieto, P. G., Combarro, E. F., del Coz Díaz, J., and Montañés, E.: A svm-based regression model to study the air quality at local scale in oviedo urban area (northern spain): A case study. Applied Mathematics and Computation, 219(17):8923–8937, 2013.

CITED LITERATURE (continued)

10. Hooyberghs, J., Mensink, C., Dumont, G., Fierens, F., and Brasseur, O.: A neural network forecast for daily average pm10 concentrations in belgium. Atmospheric Environment, 39(18):3279–3289, 2005.

11. Kolehmainen, M., Martikainen, H., and Ruuskanen, J.: Neural networks and periodic components used in air quality forecasting. Atmospheric Environment, 35(5):815–825, 2001.

12. Paschalidou, A. K., Karakitsios, S., Kleanthous, S., and Kassomenos, P. A.: Forecasting hourly pm 10 concentration in cyprus through artificial neural networks and multiple regression models: implications to local environmental management. Environmental Science and Pollution Research, 18(2):316–327, 2011.

13. Lu, W., Wang, W., Fan, H., Leung, A., Xu, Z., Lo, S., and Wong, J.: Prediction of pollutant levels in causeway bay area of hong kong using an improved neural network model. Journal of environmental engineering, 128(12):1146–1157, 2002.

14. Mishra, D. and Goyal, P.: Neuro-fuzzy approach to forecast no2 pollutants addressed to air quality dispersion model over delhi, india. Aerosol Air Qual. Res, 16:166–174, 2016.

15. Antanasijević, D. Z., Pocajt, V. V., Povrenović, D. S., Ristić, M. Đ., and Perić-Grujić, A. A.: Pm10 emission forecasting using artificial neural networks and genetic algorithm input variable optimization. Science of the Total Environment, 443:511–519, 2013.

16. Feng, Y., Zhang, W., Sun, D., and Zhang, L.: Ozone concentration forecast method based on genetic algorithm optimized back propagation neural networks and support vector machine data classification. Atmospheric Environment, 45(11):1979–1985, 2011.

17. Lv, Y., Duan, Y., Kang, W., Li, Z., and Wang, F.-Y.: Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 16(2):865–873, 2015.

18. Felder, M., Kaifel, A., and Graves, A.: Wind power prediction using mixture density recurrent neural networks. In Poster Presentation gehalten auf der European Wind Energy Conference, 2010.

CITED LITERATURE (continued)

19. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 961–971, 2016.

20. Hochreiter, S. and Schmidhuber, J.: Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

21. Sak, H., Yang, G., Li, B., and Li, W.: Modeling dependence dynamics of air pollution: Pollution risk simulation and prediction of pm _{2.5} levels. arXiv preprint arXiv:1602.05349, 2016.

22. Fan, J., Li, Q., Hou, J., Feng, X., Karimian, H., and Lin, S.: A spatiotemporal prediction framework for air pollution based on deep rnn. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 4:15, 2017.

23. Reddy, V., Yedavalli, P., Mohanty, S., and Nakhat, U.: Deep air: Forecasting air pollution in beijing, china.

24. Cui, Z., Ke, R., and Wang, Y.: Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:1801.02143, 2018.

25. Huang, C.-J. and Kuo, P.-H.: A deep cnn-lstm model for particulate matter (pm2. 5) forecasting in smart cities. Sensors, 18(7):2220, 2018.

26. Lin, Y., Mago, N., Gao, Y., Li, Y., Chiang, Y.-Y., Shahabi, C., and Ambite, J. L.: Exploiting spatiotemporal patterns for accurate air quality forecasting using deep learning. 2018.

27. Wong, D. W., Yuan, L., and Perlin, S. A.: Comparison of spatial interpolation methods for the estimation of air quality data. Journal of Exposure Science and Environmental Epidemiology, 14(5):404, 2004.

28. Mercer, L. D., Szpiro, A. A., Sheppard, L., Lindström, J., Adar, S. D., Allen, R. W., Avol, E. L., Oron, A. P., Larson, T., Liu, L.-J. S., et al.: Comparing universal kriging and land-use regression for predicting concentrations of gaseous oxides of nitrogen (nox) for the multi-ethnic study of atherosclerosis and air pollution (mesa air). Atmospheric Environment, 45(26):4412–4420, 2011.

CITED LITERATURE (continued)

29. Keler, A. and Krisp, J. M.: Spatio-temporal visualization of interpolated particulate matter (pm2. 5) in beijing. GI_Forum–Journal for Geographic Information Science, pages 464–474, 2015.

30. Contreras, L. and Ferri, C.: Wind-sensitive interpolation of urban air pollution forecasts. Procedia Computer Science, 80:313–323, 2016.

31. Arystanbekova, N. K.: Application of gaussian plume models for air pollution simulation at instantaneous emissions. Mathematics and Computers in Simulation, 67(4-5):451–458, 2004.

32. Mok, K., Miranda, A., Leong, K., and Borrego, C.: A gaussian puff model with optimal interpolation for air pollution modelling assessment. International journal of environment and pollution, 35(1):111–137, 2008.

33. Beelen, R., Voogt, M., Duyzer, J., Zandveld, P., and Hoek, G.: Comparison of the performances of land use regression modelling and dispersion modelling in estimating small-scale variations in long-term air pollution concentrations in a dutch urban area. Atmospheric Environment, 44(36):4614–4621, 2010.

34. van der Swaluw, E., de Vries, W., Vieno, M., Sauter, F., Aben, J., Velders, G., Kruit, R. W., Fagerli, H., and van Pul, A.: Modelling air quality and deposition at high resolution in the netherlands with plume and grid models. In International Technical Meeting on Air Pollution Modelling and its Application, pages 245–248. Springer, 2016.

35. Zheng, Y., Liu, F., and Hsieh, H.-P.: U-air: When urban air quality inference meets big data. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1436–1444. ACM, 2013.

36. Chen, L., Cai, Y., Ding, Y., Lv, M., Yuan, C., and Chen, G.: Spatially fine-grained urban air quality estimation using ensemble semi-supervised learning and pruning. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pages 1076–1087. ACM, 2016.

37. Chen, Y., Jiang, X., Wang, Y., and Zhuang, D.: Spatial characteristics of heavy metal pollution and the potential ecological risk of a typical mining area: A case study in china. Process Safety and Environmental Protection, 113:204–219, 2018.

38. Air quality. `https://www3.epa.gov/airquality/`.

CITED LITERATURE (continued)

39. Bishop, C.: Pattern recognition and machine learning. Berlin: Springer. ISBN 0-387-31073-8, 2006.

40. Ma, X., Tao, Z., Wang, Y., Yu, H., and Wang, Y.: Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies, 54:187–197, 2015.

41. Williams, R. J. and Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural computation, 1(2):270–280, 1989.

42. Zheng, H., Yuan, J., and Chen, L.: Short-term load forecasting using emd-lstm neural networks with a xgboost algorithm for feature importance evaluation. Energies, 10(8):1168, 2017.

43. Duan, Y., Lv, Y., and Wang, F.-Y.: Travel time prediction with lstm neural network. In Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on, pages 1053–1058. IEEE, 2016.

44. Graves, A., Jaitly, N., and Mohamed, A.-r.: Hybrid speech recognition with deep bidirectional lstm. In Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on, pages 273–278. IEEE, 2013.

45. Schuster, M. and Paliwal, K. K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11):2673–2681, 1997.

46. Time Series. `https://en.wikipedia.org/wiki/Time_series`.

47. Time Series Forecasting. `https://machinelearningmastery.com/time-series-forecasting`.

48. Sak, H., Senior, A., and Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Fifteenth annual conference of the international speech communication association, 2014.

49. Step-by-Step LSTM Walk Through. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

50. Akshay Sood: Long Short Term Memory.

CITED LITERATURE (continued)

51. Graves, A., Mohamed, A.-r., and Hinton, G.: Speech recognition with deep recurrent neural networks. In Acoustics, speech and signal processing (icassp), 2013 ieee international conference on, pages 6645–6649. IEEE, 2013.

52. Yu, Z., Ramanarayanan, V., Suendermann-Oeft, D., Wang, X., Zechner, K., Chen, L., Tao, J., Ivanou, A., and Qian, Y.: Using bidirectional lstm recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech. In Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on, pages 338–345. IEEE, 2015.

53. Osogami, T., Kajino, H., and Sekiyama, T.: Bidirectional learning for time-series models with hidden units. In International Conference on Machine Learning, pages 2711–2720, 2017.

54. Sainath, T. N., Vinyals, O., Senior, A., and Sak, H.: Convolutional, long short-term memory, fully connected deep neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, pages 4580–4584. IEEE, 2015.

55. Keras. `https://keras.io/`.

56. Tensorflow. `https://www.tensorflow.org/`.

57. Cheng, H., Tan, P.-N., Gao, J., and Scripps, J.: Multistep-ahead time series prediction. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 765–774. Springer, 2006.

58. Popescu, M.-C., Balas, V. E., Perescu-Popescu, L., and Mastorakis, N.: Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems, 8(7):579–588, 2009.

59. Specht, D. F.: A general regression neural network. IEEE transactions on neural networks, 2(6):568–576, 1991.

60. Inverse Distance Weighting Interpolation. `https://en.wikipedia.org/wiki/Inverse_distance_weighting`.

61. How IDW works. `http://pro.arcgis.com/en/pro-app/help/analysis/geostatistical-analyst/how-inverse-distance-weighted-interpolation-works.htm`.

CITED LITERATURE (continued)

62. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In Proceedings of the 1968 23rd ACM national conference, pages 517–524. ACM, 1968.

63. Kriging. https://en.wikipedia.org/wiki/Kriging.

64. Kriging2. http://desktop.arcgis.com/en/arcmap/10.3/tools/3d-analyst-toolbox/how-kriging-works.htm.

65. Land Use Regression. http://www.integrated-assessment.eu/eu/guidebook/land_use_regression.html.

66. Ryan, P. H. and LeMasters, G. K.: A review of land-use regression models for characterizing intraurban air pollution exposure. Inhalation toxicology, 19(sup1):127–133, 2007.

67. Leelőssy, Á., Molnár, F., Izsák, F., Havasi, Á., Lagzi, I., and Mészáros, R.: Dispersion modeling of air pollutants in the atmosphere: a review. Open Geosciences, 6(3):257–278, 2014.

68. Blum, A. and Mitchell, T.: Combining labeled and unlabeled data with co-training. In Proceedings of the eleventh annual conference on Computational learning theory, pages 92–100. ACM, 1998.

69. Zhu, X.: Semi-supervised learning literature survey, department of computer sciences, university of wisconsin at madison, madison. Technical report, WI, Technical Report 1530. http://pages. cs. wisc. edu/˜ jerryzhu/pub , 2006.

70. Land Data. https://catalog.data.gov.

71. Traffic. https://webapps1.cityofchicago.org/traffic/.

72. LSTM scaling. https://machinelearningmastery.com/how-to-scale-data-for-long-short-term-memory-networks-in-python/.

73. Gers, F. A., Eck, D., and Schmidhuber, J.: Applying lstm to time series predictable through time-window approaches. In Neural Nets WIRN Vietri-01, pages 193–200. Springer, 2002.

74. AirNow Maps. https://gispub.epa.gov/airnow/.

CITED LITERATURE (continued)

75. Cheng, W., Shen, Y., Zhu, Y., and Huang, L.: A neural attention model for urban air quality inference: Learning the weights of monitoring stations. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

VITA

NAME	Marco Miglionico
EDUCATION	
	Master of Science in Computer Science, University of Illinois at Chicago, December 2018, USA
	Pursuing Master of Science Degree in Computer Science and Engineering, Politecnico di Milano, Italy
	Exchange program at the Hong Kong University of Science and Technology (HKUST), January 2016, Hong Kong
	Bachelor's Degree in Computer Science and Engineering, Feb 2017, Politecnico di Milano, Italy
LANGUAGE SKILLS	
Italian	Native speaker
English	Full working proficiency
	2016 - IELTS examination (6.5/9)
	A.Y. 2017/18 One Year of study abroad in Chicago, Illinois
	A.Y. 2016/17. Lessons and exams attended exclusively in English
	A.Y. 2015/16 Six Months of study abroad in Hong Kong, HK
SCHOLARSHIPS	
Spring 2015	Research Assistantship (RA) position (20 hours/week) with monthly stipend
WORK EXPERIENCE AND PROJECTS	
Jan 2018 - Current	Artifical Intelligence Architect at Stanley Black and Decker Digital Accelerator, Atlanta, GA.
Sep 2018 - Dec 2018	Data Science Internship at Norfolk Southern, Atlanta, GA
January 2018 - May 2018	Research Assistant an the University of Illinois, Chicago, IL

VITA (continued)

2018	Movies Recommender System using Stacked Autoencoders and Boltzman Machines
2018	Long Short Term Memory(LSTM) for Google Stock Price Prediction
2017	CS412 Machine Learning Final Project. Build a classifier able to predict whether a borrower,based on their loan application details, will either fully pay off their loan or not.