XLIII CILAMCE
21-25, November, 2022
FOZ DO IGUAÇU - PR - BRAZIL

Recurrent Neural Networks for air-quality forecast models in the city of Rio de Janeiro

J. F. L. Leocadio[1], N. F. F. Ebecken[1]

[1] *COPPE- Programa de Engenharia Civil, Universidade Federal do Rio de Janeiro*
Avenida Horácio Macedo, 2030- BLOCO B, CEP 21941-914, Ilha do Fundão, Rio de Janeiro/RJ, Brasil
jose.leocadio@coc.ufrj.br, nelson@ntt.ufrj.br

Abstract. The tropical climate of the metropolitan region of Rio de Janeiro is especially susceptible to air pollutants such as Ozone and Particulate Matter, which are directly connected to serious cardiopulmonary illnesses. The goals of the present work were: to explore the local meteorological data to find useful patterns among the information and to exam the performance of an ensemble model of Recurrent Neural Networks on the prediction of daily maximum pollutant levels. The analyzed dataset is provided by the Rio de Janeiro local government and it is composed by hourly-levels for pollutants and meteorological features from eight different locations. The Spearman correlation test among the variables of different stations showed that adjacent locations have similar data, with values up to 95% of correlation depending on the examined variable. The experiments showed that the ensemble model has superior performance to simpler models in 3 out of 4 studied scenarios.

Keywords: Air-Quality, Recurrent Neural Networks, Computer Systems. Environmental Engineering

1 Introduction

According to the World Health Organization (WHO) [1] since 2018, 4.1 million people have died from air pollution-related diseases. 91% of these deaths occurred in third-world countries like Brazil. The fast industrial development of these nations is associated with big climate changes, which make these areas essentially vulnerable to this kind of misfortune. In this scenario, the tropical climate of the metropolitan region of Rio de Janeiro seems to be particularly susceptible to air pollutants that are directly linked to cardiopulmonary illnesses. Therefore air quality monitoring technologies must be able to precisely forecast extreme events.

Previously Ghoneim et al. [2], Luna et al. [3] and Li et al. [4] tested the performance of classical models such as Feed-Forward Neural Networks and Support-Vector Machines (SVM) on the task of predicting hourly Ozone and Particulate Matter levels, acquiring reasonable results. Nevertheless given the nonlinear nature of the meteorological data, classical methods for statistical forecast might not have satisfactory performance on the aforementioned task.

In order to better learn the intricacies among air-quality data, the literature recommends employing more robust time-series specific models, such as Recurrent Neural Networks (RNN). Kok et al. [5], Fan et al. [6], Li et al. [7], Bui et al. [8], Sak et al. [9], Pardo and Malpica [10], Wang et al. [11], Navares and Aznarte [12] applied Long-Short Term Memory (LSTM) on the task of prediction of hourly-pollutant levels in several cities around the World. This variant of RNN has the capacity of leaning the long term dependencies amongst time-series data and showed very promising results in many different fields. However when employed with air-pollution data the LSTM seems to increasingly lose performance when it comes to predict 4 hours ahead and above.

Once short time spam predictions would have no practical use, Hossain et al. [13] proposed using RNN and its variants to forecast maximum daily levels for each pollutant. To further increase the accuracy of the predictions given by the RNN, Du et al. [14] and Zhang et al. [15] suggested ensembling LSTM and Convolutional Neural Networks (CNN) so each neural network would be in charge of learning different facets of the data.

With that in mind, in its initial stages the present work proposes an ensemble model, using three different recurrent-type neural networks for the task of prediction of daily maximum pollutant levels. Supposedly in all of the previously mentioned works, it would be necessary to train one model to forecast each different variable, which could be impractical with large datasets. To solve this issue and to better understand the performance of the proposed models, the current work used the multiple-input and multiple-output technique, so the model will provide predictions to all the variables of the studied dataset at once.

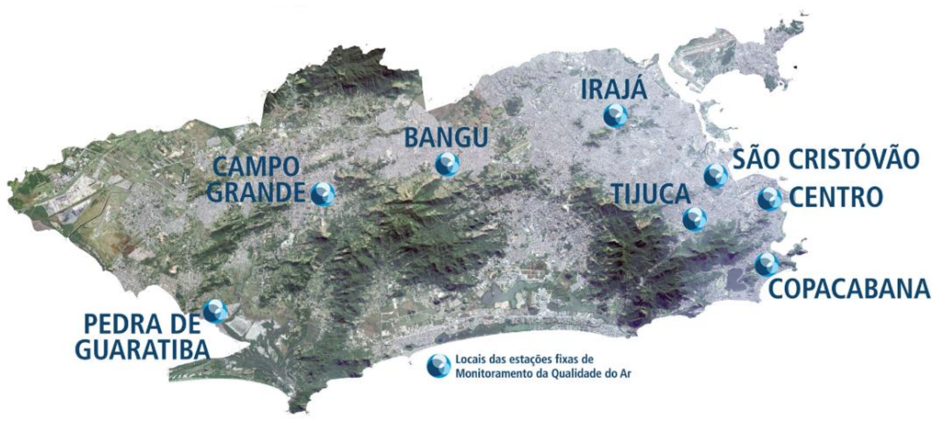
The contributions of this work are:

1. a clear guide on how to transform the air-quality data to obtain more meaningful forecast results with neural networks.
2. an underexplored technique to provide predictions to all the features from the data at once, which could potentially produce a lighter model in production.
3. a thorough investigation of the evidence that local nearby air-quality stations have similar data.
4. an investigation of several neural network architectures that will lead to a deep learning architecture in future works.
5. an ensemble of neural networks that outperforms single networks on forecasting air-quality daily levels.

2 Objectives

The objective of this work is to employ data mining techniques to understand the patterns amongst de air-quality data of the metropolitan region of Rio de Janeiro. The exploratory data analysis will be used to plan an alert-system that will warn the local population about possibly harmful pollution levels.

3 Methodology



3.1 Data Description

The raw data is composed by information collected over the last 10 years in eight different districts: Centro, Copacabana, São Cristóvão, Tijuca, Irajá, Bangu, Campo Grande e Pedra de Guaratiba. Each air-quality station can collect hourly levels of Carbon-Monoxide (CO), Particulate-Matter (PM10 and PM2.5,) Ozone (O_3), Sulfur-Dioxide (SO_2), and Nitrogen-Oxides (NOx). The same data set also provides hourly variations for wind speed, wind direction, solar radiation, rainfall, relative humidity, temperature, and atmospheric pressure. The UTM coordinates of all the stations are also part of the set.

All of the stations had missing data issues that had to be corrected before its usage. It's possible to notice that some of the data collect locations are geographically closer than others. The next stage of the work was to find out if the nearby stations had similar enough data, so it could be used to fill-in missing adjacent station values.

Figure 1 Location of the air-quality collect stations

3.2 Spearman correlation and missing value treatment

The following figures show the Spearman Correlation coefficient among the same pollutants in different locations. The values vary between -1.0 (color blue) for complete inverse correlation and 1.0 for complete direct correlation. Some variables are absent in certain stations so it is omitted from the graphs.

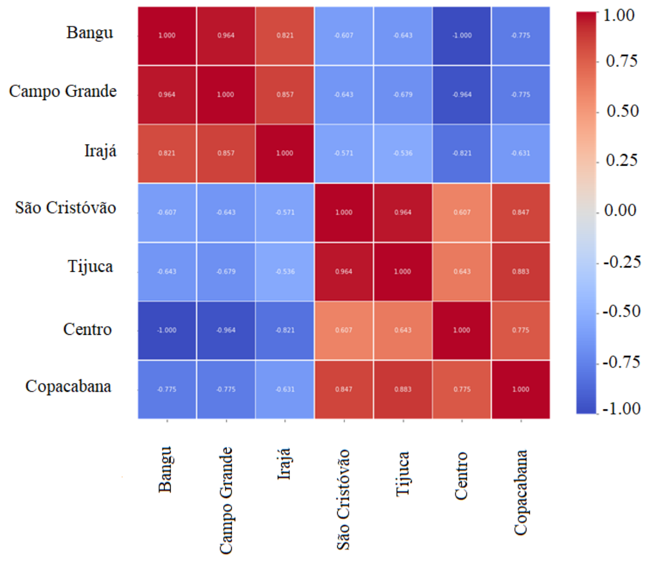


Figure 2 Spearman correlation levels for SO_2

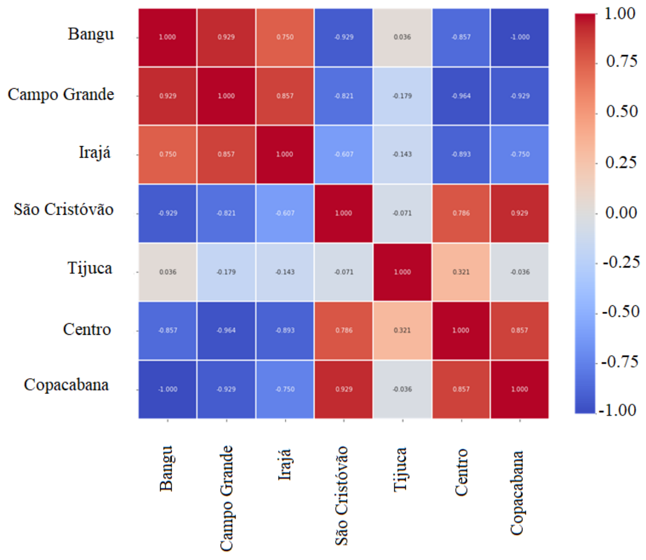


Figure 3 Spearman correlation levels for CO

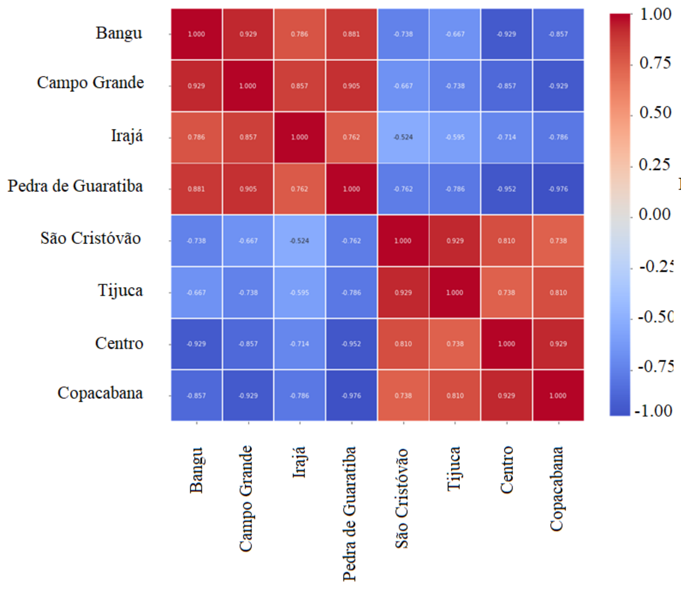


Figure 4 Spearman correlation levels for O_3

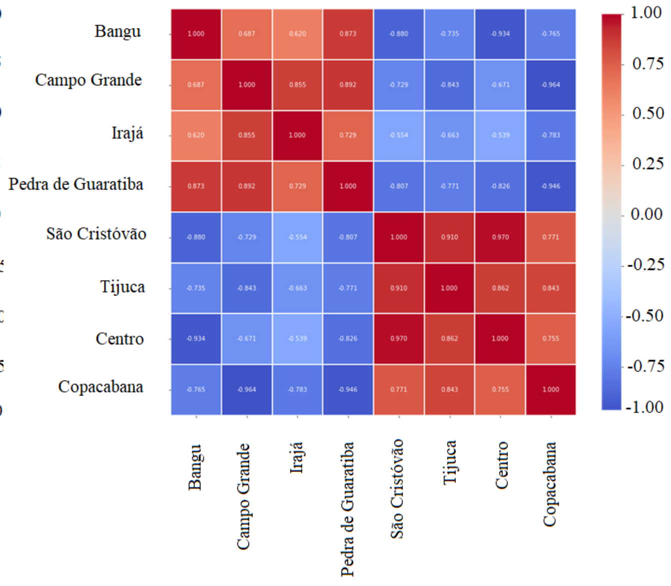


Figure 5 Spearman correlation levels for PM10

It is possible to notice from the figures above that apparently the closer the collect stations are from each other, the more directly correlated is the data. From the Figure 1 it is noticeable that there is a cluster of stations formed by Tijuca, São Cristóvão, Centro and Copacabana. The missing values treatment proceeded taking it into account, as it is explained ahead. Once the other four stations are too far from each other, the current work continued only with the data from the aforementioned cluster.

The filling of missing values followed the Inverse Distance Weighting technique as seen in [16] and more recently in [17]. In this relatively underexplored method, the blank values are obtained in according to the following formula:

$$x_A = \frac{x_B(1/d_{AB}) + x_C(1/d_{AC}) + x_D(1/d_{AD})}{(1/d_{AB}) + (1/d_{AC}) + (1/d_{AD})} \tag{1}$$

Where:

x_A : value for a variable from the dataset A at a given time
x_N : value for a variable from the dataset N at a the same given time
d_{AN} : distance between the stations A and N

3.3 Multiple-input and multiple output method

Given the nature of the multivariate time-series, the most common approach for building forecast models seems to be training one neural network to predict each variable separately. Petersen et al. [18] suggested that modern neural networks might be capable of forecasting multiple variables at once without any loss of performance. The main difference between this method and the previous one is that instead of outputting a single value, the model will provide a vector containing predictions for each variable served as input.

After the preparation stage, the data set for each station contained 12 variables, six meteorological and six related to air-pollutants. The input vector will contain data from the last seven days and will forecast the maximum levels for the next day. With that in mind, the input vector will have the shape (84 x 1), 12 variables times 7 days, and the output vector will have the shape (12 x 1) containing one prediction for each variable served as input to the model.

3.4 Cross-Validation for time-series data

Cross-Validation is an established method to acquire better generalizing models. Besides preventing over fitting of the trained neural network, this technique is supposedly a more reliable way to evaluate its performance, once the final error obtained by it is an average of k-values.

Once time-series data has a sequential nature, classical methodologies for Cross-Validation are not easily applied to it. It is not possible to randomly associate data to the training and test sets, since it could violate its sequential configuration, and it would not make sense to use future values to forecast the past.

Cerqueira et al. [19] and Schnaubelt [20] presented several attempts to overcome this limitation and apply the Cross-Validation technique to time-series data. One of the exposed methods is the Time Series Split. Its main idea is to split the data set in two blocks, a training block and a test block, in each training iteration (fold), always maintaining the test block ahead of the training block, which will be increased at each time, as shown in the following figure. All the models of the present work were trained with 5-fold Time Series Split Cross-Validation technique.

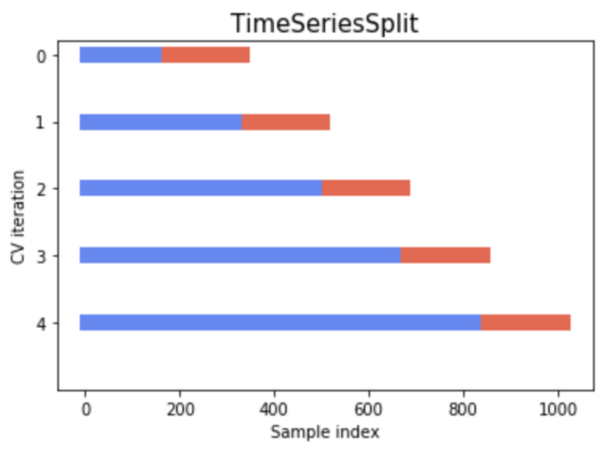


Figure 6 Time Series Split Cross-Validation scheme

3.5 Recurrent Neural Networks

In a classical neural network, also known as Feed-Forward (FNN), each layer is composed by neurons and the connections between these unities always follow the same direction. A Recurrent Neural Network (RNN) is a variant of FNN that possess cyclical nature. In a RNN each neuron has also a self-connection that allows a type of memorization of the data input, so it will have influence over the output of the network.

The inference process of a RNN with one input layer containing I neurons, one hidden layer with H neurons and one output layer containing K neurons, where the input of the neural network is as sequence X of length T, is given by the formula:

$$a_h^t = \sum_{i=1}^{I} w_{ih} x_i^t + \sum_{h'=1}^{H} w_{h'h} b_{h'}^{t-1} \tag{2}$$

$$b_h^t = \theta_h(a_h^t) \tag{3}$$

$$a_k^t = \sum^{H} w_{hk} b_h^t \tag{4}$$

Where:

x_i^t : value for the ith dimension on timestep t
w_{ij} : weight between neuron i and j
a_h^t : input of neuron j at timestep t
b_h^t : ativation of neuron j at timestep t
θ_h : activation function of neuron h

In a FNN the model training is executed in according to the basic Back Propagation algorithm (BP). However, once the RNN has sequential data, the memory transference has to be taken into account. So the training stage has to stack results from BP over the time dimension, which is done the by the Back Propagation Through Time algorithm, given by the following formula:

$$\delta_j^t = \frac{\partial L}{\partial {a^t}_j} \tag{5}$$

$$\delta_j^t = \theta'(a_h^t)\left(\sum_{k=1}^{K} w_{hk}\,\delta_k^t + \sum_{h'=1}^{H} w_{hh'}\,\delta_h^{t+1}\right) \tag{6}$$

$$\frac{\partial L}{\partial w_{ij}} = \sum_{t=1}^{T} \frac{\partial L}{\partial {a^t}_j}\frac{\partial {a^t}_j}{\partial w_{ij}} = \sum_{t=1}^{T} \delta_j^t b_i^t \tag{7}$$

Where:

L : loss function
δ_j^t : gradient of loss function over input of neuron j at timestep t

After obtaining the gradients, the weights of the neural networks are updated by the gradient descent algorithm. One issue of the RNN is that through the time steps the gradient values may become too small, causing the model to take far too long to train, this drawback is known as Vanishing Gradient.

To resolve that problem changes on the recurrent unity from the RNN were proposed. Hochretiter and Schmidhuber [21] created the Long-Short Term Memory (LSTM) in which the neurons are replaced by memory unities, each containing three gates: an input gate, that helps on the identification of important elements that need to be added to the cell state; an forget gate that controls the information that should be forgotten and a output gate that extracts useful information from the cell and presents it as an output. The gates ensure that gradient information of LSTM will not vanish during the back propagation. Chung et al. [22] proposed a further optimization of the memory unity creating the Gated-Recurrent Unity (GRU), that is similar to the LSTM but with fewer gates. The GRU has only a hidden state for memory transfer between the recurrent units, resulting in no separate cell state.

3.6 Combination of predictions with ensemble method

Given the stochastic nature of the neural networks, each time a model is trained one different version of the function that maps the output is learned, resulting in different performances for the same training and test sets. One method to reduce this variance is to train multiple models and combine their prediction. This technique, known as ensemble, adds a bias that balances the variance from the neural networks and usually improves the quality of the given predictions.

The simplest way to combine predictions is to get the average value returned from all the different models. This work used weighted averages as suggested from Bishop [23]. The weights varied between 1 and 3 and were assigned to each neural network results (RNN, GRU and LSTM) in according to their performance on the previous tests.

4 Results and Discussion

The performance of each model was measured by RMSE and MAE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{8}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{9}$$

In some cases the studied neural networks had very close performance, so the total time to finish the training had to be taken into consideration as well. Through the whole train process of all the models the optimizer was the Adam Algorithm and the Learning Rate was kept fixed at 0.0005.

The first test aimed to define both the ideal batch size and the number of training epochs for the models. Three sizes of batch were tested, 32, 64 and 128 with 50, 100 and 150 epochs for training duration. For the four stations, the best results were obtained by the models trained for 150 epochs with batch sizes of 32 or 64.

The proceeding test studied the performance variation of the three neural networks when the number of neurons on the hidden layer was changed. Results showed that overall the GRU had the best performance on the task, followed by the simple RNN and by the LSTM. The optimal number of neurons seems to be around 64, although networks with less neurons had faster train.

The next test studied the hypothesis that deeper models would have better performances on the task. Results showed that when the number of training epochs was kept the same, the best configuration contained only one hidden layer. It is possible that the deeper models should have larger training duration to acquire optimal results, once the number of trainable hyperparameters was increased with more layers.

The fourth experiment tested the hypothesis that the results obtained in previous stages could be further improved by the usage of dropout and batch normalization techniques. The results showed that batch normalization didn’t do any good for the predictions and in most cases it increased the total duration of the training. On the other hand, the dropout of 20% cut half of the training time without harming the results, which could be useful when dealing with larger data sets.

The last stage studied the hypothesis that an ensemble of neural networks could have the best performance on the prediction task. The following table shows the results for the station of Copacabana. It is possible to notice a slight discrepancy between the RMSE and MAE for the same variables. That may occur due to the RMSE being more sensible to outliers or extreme cases, once the error is squared to obtain the metric values.

From the same table, it is clear that for most variables the ensemble of models performed better than the singular networks. Other two stations, Centro and Tijuca, had very similar results, with the ensemble model also being the best. The results from the São Cristóvão station showed that on this scenario the ensemble had the same performance as the LSTM.

Tabela 1 model results for the Copacabana air-quality station

Variable	Metrics	Ensemble	RNN	LSTM	GRU
Temperature	RMSE	3,05	3,33	3,22	3,09
NOx	RMSE	23,88	24,20	24,18	24,64
CO	RMSE	0,20	0,20	0,20	0,20
Rain	RMSE	0,89	0,92	0,89	0,89
Atm. Pressure	RMSE	2,87	3,20	2,94	2,92
Wind Speed	RMSE	0,27	0,28	0,28	0,28
SO_2	RMSE	4,45	4,60	4,50	4,45
PM10	RMSE	18,68	18,68	19,17	18,62
O_3	RMSE	12,28	12,67	12,39	12,66
Relative Humidity	RMSE	6,81	6,99	6,89	6,97
NO	RMSE	15,69	16,13	15,28	16,04
NO_2	RMSE	14,45	14,50	14,43	14,85
Temperature	MAE	2,37	2,62	2,58	2,41
NOx	MAE	18,46	18,80	18,81	19,31
CO	MAE	0,17	0,17	0,17	0,17
Rain	MAE	0,60	0,62	0,63	0,62
Atm. Pressure	MAE	2,24	2,52	2,28	2,29

Wind Speed	MAE	0,22	0,23	0,22	0,22
SO_2	MAE	3,66	3,80	3,63	3,65
PM10	MAE	15,17	15,10	15,62	15,11
O_3	MAE	9,87	10,20	9,97	10,19
Relative Humidity	MAE	5,41	5,46	5,47	5,51
NO	MAE	12,05	12,42	11,85	12,42
NO_2	MAE	11,31	11,26	11,22	11,56

The F-test for the standardized results of the Copacabana station model gave F=9.195 and p=0.00015, meaning that the null hypothesis (there is no difference among any pairs of average metrics) can be rejected, or that there is a meaningful statistical difference on at least two pairs of mean metrics.

5 Conclusions

This work studied a method to fill in missing values and proposed a way to ensemble neural network predictions. The tests showed that the ensemble of models outperforms simple neural networks. The acquired results will be used to guide the further development of the models into deep learning neural networks in the future. Once the same configuration was used to compose the three neural networks of the ensemble, further tests are necessary to validate if different model architectures for each networks could further improve the given results.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] OMS (Organização Mundial da Saúde). *Burden of Disease from Household Air Pollution for 2016*. 2018.
[2] O. A. Ghoneim: Doreswamy: B. R. Manjunatha. "Forecasting of Ozone Concentration in Smart City using Deep Learning". *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI*). 2017
[3] A.S. Luna: M. L. L. Paredes: G. C. G. de Oliveira: S. M. Corrêa. "Prediction of ozone concentration in tropospheric levels using artificial neural networks and support vector machine at Rio de Janeiro, Brazil". *Atmospheric Enviroment 98*. 2014
[4] X. Li: L. Peng: Y. Hu: J. Shao: T. Chi. "Deep learning architecture for air quality predictions". *Environ Sci Pollut Res*.2016.
[5] I. Kök: M. U. Simsek: S. Özdemir. "A deep learning model for air quality prediction in smart cities**"**. *IEEE International Conference on Big Data (BIGDATA)*, 2017.
[6] J. Fan: Q. Li: J. Hou: X.Feng: H. Karimian: S. Lin. "A Spatiotemporal Prediction Framework for Air Pollution Based on Deep RNN". *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume IV-4/W2*. 2017.
[7] X. Li: L. Peng: X. Yao: S. Cui: Y. Hu: C. You. "Long short-term memory neural network for air pollutant concentration predictions: Method development and evaluation". *Environmental Pollution 231*. 2017.
[8] T. Bui: V. Le: S. K.Cha. "A Deep Learning Approach for Forecasting Air Pollution in South Korea Using LSTM". *Environmental Pollution 231*. 2018.
[9] H. Sak: G. Yang: B. Li: W. Li. "Modeling Dependence Dynamics of Air Pollution: Pollution Risk Simulation and Prediction of PM2.5 Levels". 2016.
[10] E. Pardo: N. Malpica. "Air Quality Forecasting in Madrid Using Long Short-Term Memory Networks". *IWINAC*. 2017.
[11] J. Wang: X. Zhang: Z. Guo: H. Lu. "Developing an early-warning system for air quality prediction and assessment of cities in China". *Expert Systems With Applications*. 2017.
[12] R. Navares: J. L. "Aznarte. Predicting air quality with deep learning LSTM: Towards comprehensive models". *Ecological Informatics*.2019.
[13] E. Hossain: M. A. U. Shariff: M. S. Hossain: K. Andersson. "A Novel Deep Learning Approach to Predict Air Quality Index". *Proceedings of International Conference on Trends in Computational and Cognitive Engineering*. 2020.
[14] S. Du: T. Li: Y. Yang: S. J. Horng. "Deep Air Quality Forecasting Using Hybrid Deep Learning Framework*".* *IEEE Transactions on Knowledge and Data Engineering*. 2019.
[15] Q. Zhang: V. O. Li: J. C. Lam: Y. Han. "Deep-AIR: A Hybrid CNN-LSTM Framework for Fine-Grained Air Pollution Forecast". 2020.
[16] D. Shepard. "A two-dimensional interpolation function for irregularly-spaced data". *Proceedings of the 1968 ACM National Conference*. 1968.
[17] M.J. Abedini: M. Nasseri. "Inverse Distance Weighting Revisited". 2009.
[18] N. C. Petersen: F. Rodrigues: F. C. Pereira. "Multi-output Bus Travel Time Prediction with Convolutional LSTM Neural Network". 2019
[19] V. Cerqueira: L. Torgo: I. Mozetic . "Evaluating time series forecasting models An empirical study on performance estimation methods". 2019.
[20] M. Schnaubelt. "A comparison of machine learning model validation schemes for non-stationary time series data". *FAU Discussion Papers in Economics, No. 11/2019*. 2019.
[21] S. Hochreiter: J. Schmidhuber. "Long-Short-Term memory". *Neural Computation 9(8):1735{1780*. 1997.
[22] J. Chung: C. Gulcehre: K. Cho: Y. Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". 2014.
[23] C. M. Bishop. "Neural Networks for Pattern Recognition". 1995.