TRAINING RECURRENT NEURAL NETWORKS FOR PARTICULATE MATTER CONCENTRATION PREDICTION.

C. J. Masinde [1], J. Gitahi [1], M. Hahn [1, *]

[1] Faculty of Geomatics, Computer Science and Mathematics, Hochschule für Technik Stuttgart, Schellingstraße 24 70174 Stuttgart, Germany - (calebjuma27@gmail.com, joseph.gitahi@hft-stuttgart.de, michael.hahn@hft-stuttgart.de)

KEY WORDS: Recurrent neural networks, long short-term memory, gated recurrent unit, optimizers, Particulate Matter pollution, prediction, air quality monitoring, sensor network

ABSTRACT:

A high level of particulate matter in the atmosphere has an adverse long-term effect on human health. It has been associated with increased pulmonary tract and lung infections. It is more common in urban areas, especially megacities due to the confluence of industries and motorized machinery. Considering that most of the world's population lives in urban areas, there is a need to monitor air pollution arising from particulate matter in order to ensure clean and safe air in cities in accordance with goal 11 of the Sustainable Development Goals. One way of doing this is through the use of Recurrent Neural Networks (RNN), which are suited for time varying data. Particulate matter concentration recorded by a network of low-cost sensors in Stuttgart is trained on three of the most popular RNN variants: Standard LSTM, Peephole LSTM and Gated Recurrent Unit. Two optimizers are used, Stochastic Gradient descent and Adam. Training is done on a single sensor and the optimum weights transferred and used in the prediction of other sensor values. This study concludes that Gated Recurrent Unit with Stochastic Gradient Descent is the most effective of the three variants in predicting particulate matter $PM_{2.5}$ concentrations. In addition to this, weight transfer between sensors is not affected by temperature, wind direction, wind speed and geographic distance between sensors but rather by atmospheric pressure and the similarity of recorded Particulate matter levels.

1. INTRODUCTION

1.1 Particulate Matter

Particulate matter is a complex mixture made up of Sulphates and organic compounds. It is one of 4 elements that cause air pollution, the other three being Ozone, Nitrogen dioxide and Sulphur dioxide (WHO Regional Office for Europe, 2003). In the urban environment, it is majorly produced by motor vehicle fumes and coal combustion. Small dust particles also qualify to be considered as particulate matter but are at times left out due to their low inflammatory effect as compared to particles from combustion engine, coal, burning wood and oil. (Pope III et al., 2002) established that for every 10μg/m³ increase in particulate matter in the environment, there was a relatable increase by 4%, 6% and 8% of cardiopulmonary mortality.

Particulate matter dictates air quality and is measured in micrograms per cubic meter of air. It is classified based on its aerodynamic size; the size of a unit sphere with the same aerodynamic properties of the matter. The potency of the particulate matter is inversely proportional to its size with the smallest having the greatest chance of reaching the lungs and entering the bloodstreams (Badura et al., 2018). Primary particulate matter is created from combustion and emitted directly to the atmosphere. It is composed of fine particles with a diameter <2.5 µm in size ($PM_{2.5}$) while secondary particulate matter is produced through the mechanical breakdown or chemical combination of matter in the atmosphere and has a diameter < 10 µm in size (PM_{10}) (Amaral et al., 2015).

The presence of particulate matter in the atmosphere is empirically determined using two main methodologies; size distribution and the more common concentration principle. Size distribution as its name infers seeks to determine the size of the aerosol content using a number of instruments such as microscopes and Impactors (Amaral et al., 2015). Concentration on the other hand measures the mass, number or surface area of particulates per a unit volume of air that passes through the sensor. Concentration measurement devices utilize Gravimetric, Optical counters, microbalances and electrical charge.

Gravimetric falls under the concentration principle and is a traditional but very efficient way of measuring particulate matter, with an added advantage of determining the chemical composition of the particles (Whalley, Zandi, 2016). However, it is labour intensive, requires specialized instruments and lacks the instantaneous results needed in many of today's IoT processes as the measurements have to be processed first before shared. Due to its high accuracy, it has been adopted by regulatory bodies such as the Environmental Protection Agency.

The other two popular concentration methods include microbalances and Optical Particle Counters. Tapered Element Oscillating Microbalance (TEOM) provides for a near real time air quality index. It determines particles concentration by monitoring changes in the oscillating frequency of a quartz glass tube during interaction with particles in the environment (Amaral et al., 2015). TEOMS are also expensive for individuals leaving their operations to be handled by government departments responsible for environmental protection.

The Optical Particle Counter on the other hand utilizes light dispersion to quantify the concentration of particles in the atmosphere. The scattered light is detected by a photometer and gives feedback on the amount of particulate matter present in the

* Corresponding author

atmosphere (Badura et al., 2018). The low complexity behind its operation makes it affordable to produce and use. Furthermore, an environmentally aware public eager to participate in air pollution initiatives has created an ecosystem of volunteers and commercial entities manufacturing a wide variety of optical counters and their associated accessories (Karagulian et al., 2019). This has improved the spatial density of the particulate matter sensor grid and made available a huge pool of data that is transmitted and consumed in real time.

1.2 Recurrent Neural Networks

An artificial neural network is a computational replica of a biological neural network. A neuron is a unit which fires up when a linear combination of its inputs exceeds a particular threshold (Norvig, Russell, 2009). Many of these units interconnected together make up an artificial neural network.

There are two ways of connecting neurons in a system: through a feed forward mechanism and through a recurrent system. The feed forward network directs its flow in one direction whereas the recurrent network allows output to be fed back into the input. This makes recurrent networks capable of having memory as the previous outputs can be used to determine the response of the unit to the new inputs.

In theory, a Recurrent Neural Network should be able to learn over long time sequences but this is not the case. If one applies the sigmoid function (activation function) on the data for quite some time, it flattens out or rather vanishes thereby stopping the learning process. The problem of the vanishing gradient has been solved through the introduction of the Long Short-Term Memory cell by (Hochreiter, Schmidhuber, 1997). Thus, LSTM then enable the network to remember by preserving loss transfer during backpropagation allowing the network to continue learning.

Long Short-Term memory utilizes gates to aid the Recurrent Neural Network remember long term dependencies. These gates are: The input, forget and output gates.

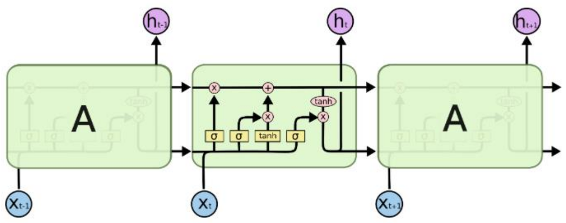


Figure 1. LSTM (Source: (Olah, 2015))

In the standard case, the three gates are present and separate. The activation function for the gates is the sigmoid function while the activation function for the main neuron is tanh.

$$f_t = \sigma\left(w_f[h_{t-1}, x_t] + b_f\right) \quad (1)$$
$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \quad (2)$$
$$\acute{C}_t = tanh(w_c[h_{t-1}, x_t] + b_c) \quad (3)$$
$$C_t = f_t * c_{t-1} + i_t * \acute{C}_t \quad (4)$$
$$o_t = \sigma(w_0[h_{t-1}, x_t] + b_o) \quad (5)$$
$$h_t = o_t * \tanh(C_t) \quad (6)$$

where x_t = input x at time t
f_t, i_t, o_t = forget, input and output gates
$b_{f,i\ or\ o}$ = bias factor
$\acute{C}_t$ = neuron output
C_t = current cell state
h_t = hidden layer
w = weight

A modification of the standard LSTM yields another RNN variant. This contains peep holes that allow the gates to know the previous cell state of the RNN layer. LSTM peephole allows for the RNN to be extended and be able to time and count sequences without the need for short-term training (Gers, Schmidhuber, 2000).

$$f_t = \sigma\left(w_f[C_{t-1}, h_{t-1}, x_t] + b_f\right) \quad (7)$$
$$i_t = \sigma(w_i[C_{t-1}, h_{t-1}, x_t] + b_i) \quad (8)$$
$$o_t = \sigma(w_o[C_{t-1}, h_{t-1}, x_t] + b_o) \quad (9)$$

where C_{t-1} = previous cell state
f_t, i_t, o_t = forget, input and output gates
$b_{f,i\ or\ o}$ = bias factor

The third variant that will be investigated is the Gated Recurrent Unit. It simplifies the standard LSTM architecture by collapsing the input and forget gates into a reset gate and an update gate. This results in the combination of the current cell state with the hidden state (Cho et al., 2014).

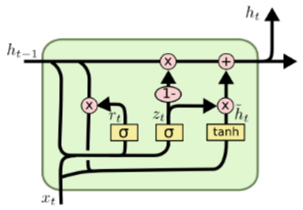


Figure 2. GRU (Source: (Olah, 2015))

$$z_t = \sigma(w_z[h_{t-1}, x_t]) \quad (10)$$
$$r_t = \sigma(w_r[h_{t-1}, x_t]) \quad (11)$$
$$\hat{h}_t = tanh(w[r_t * h_{t-1}, x_t]) \quad (12)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t \quad (13)$$

where x_t = input x at time t
z_t = update gate
r_t = reset gate
h_t = current hidden layer
h_{t-1} = previous hidden layer
w = weight

1.3 Gradient Descent

Gradient Descent is an optimization technique during backpropagation that seeks to achieve the lowest possible loss value by iteratively moving close to the minima of a differentiable function (Ruder, 2016). Two important optimizers used to implement gradient descent include: Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam).

A standard gradient descent will use all the datasets in a training sample to obtain the generalized weights for each iteration. However, when the training dataset is large, the method becomes computational uneconomical. This is compounded by the fact that sampled data units are individually composed of attributes known as features. To remedy this, stochastic gradient descent is used. SGD randomly selects one sample in a provided training batch set to compute the weights per each iteration (Srinivasan, 2019). Only one learning rate is used leading to a uniform stepping size being applied to the weights of all features present in the sample.

Adam is an extension of SGD (Kingma, Ba, 2014). However, instead of utilizing only one learning rate for all the features present in the single chosen dataset within a batch, it adapts the learning rate to each individual feature. This creates different step sizes for the available features and can be useful when irregular distribution patterns are present amongst features in a sample.

2. DATA

The geographical area of study is Stuttgart city administrative unit. It lies on coordinates 48.78° north of the equator and 9.18° east of the Greenwich meridian. As of 20th February 2020, the city has three functioning particulate matter sensors (Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg, 2020) : Stuttgart Arnulf-Klett-Platz, Stuttgart Am Neckartor and Stuttgart-Bad Cannstatt. Luftdaten, a project by OK Lab Stuttgart, has a dense network of low-cost PM sensors in the city, with approximately 350 sensors.

The data used in this study consists of PM_{10} and $PM_{2.5}$ measurements crowdsourced by OK Lab Stuttgart (OK Lab Stuttgart, 2019) as part of the 'Hack your city' citizen science project (Wissenschaft im Dialog, 2015) and hourly weather data composed of temperature, humidity, pressure, wind speed and wind direction obtained from OpenWeatherMap (OpenWeatherMap, 2019). Both sets of data are aggregated into hourly samples. Luftdaten temporal resolution is 2.5 minutes, but aggregated into hourly values in this study.

QGIS (QGIS Development Team, 2019) is used to visualize the hourly PM data and meteorological information. Weather data has a spatial coverage within and close to the Stuttgart region. The time window of data capture is close to 5 months spanning from 28th May 2019 until 11th November 2019, the date of the data download.

The PM data is clipped to the Stuttgart region and its data filtered using QGIS to a timeframe similar to the weather stations dataset. The resulting dataset consists of 351 particulate matter sensors having a total of 882305 records from 8th May 2019 at 15:00 Hrs GMT to 11 November 2019 15:00 Hrs GMT. The available 52 weather stations have a total of 176161 records containing meteorological data from 28th May 2019 at 15:00 Hrs GMT to 11 November 2019 15:00 Hrs GMT.

All Weather stations are used in the study. They are not clipped to Stuttgart as some stations are not within the study area but closer to sensors on the border of the study area.

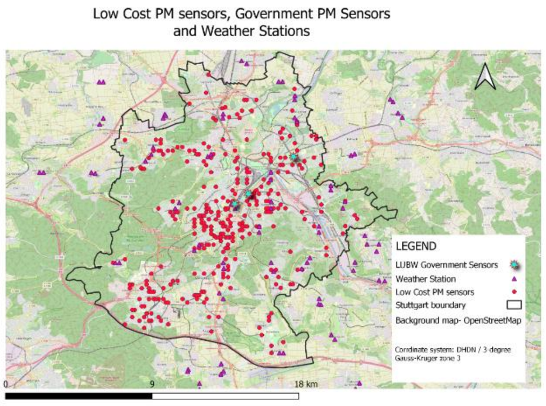

Low Cost PM sensors, Government PM Sensors
and Weather Stations
LEGEND
LUBW Government Sensors
Weather Station
Low Cost PM sensors
Stuttgart boundary
Background map- OpenStreetMap
Coordinate system: DHDN / 3 degree Gauss-Kruger zone 3
0
9
18 km

Figure 3. Weather Stations and PM sensors after filtering to 11th Nov 2019 and Stuttgart boundary

3. EXPERIMENT AND RESULTS

This research is made up of 4 categories: data preparation, selection of the base sensor, training of the recurrent neural networks and finally investigation of weight transfer. This is illustrated in Figure 4 below

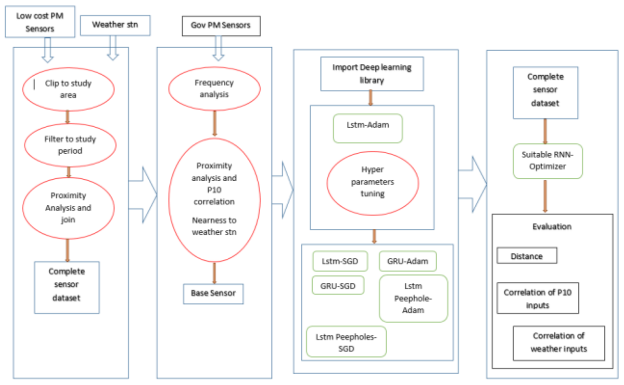


Figure 4. Workflow

3.1 Complete sensor dataset

Python (Python Software Foundation, 2019) available as PyQGIS in QGIS is used to sort through the data and identify the optimal sensors, which are sensors that remitted data continuously for a specified time duration. This is informed by the study period during which historical weather data is available. Thus, 5 months is chosen as the threshold. Taking 1 month to have 4 weeks, sensors considered as having enough data need to have a minimum of 3360 records corresponding to 3360 hours in a 5-month duration.

After the identification of the optimal sensors, the next step is to assign them meteorological information of their respective closest weather station. To accomplish this, proximity analysis is first carried out in PyQGIS and the nearest weather station to each sensor identified. Then, a join operation is done to move all the datasets of the selected weather stations to their respective particulate matter sensors.

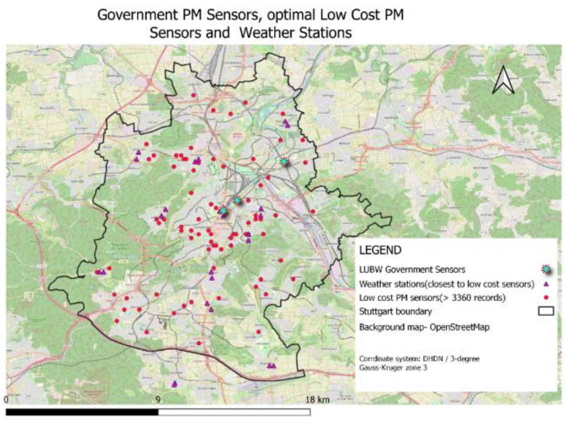

Government PM Sensors, optimal Low Cost PM
Sensors and Weather Stations
LEGEND
LUBW Government Sensors
Weather stations(closest to low cost sensors)
Low cost PM sensors(> 3360 records)
Stuttgart boundary
Background map- OpenStreetMap
Corrdinate system: DHDN / 3-degree Gauss-Kruger zone 3
0
9
18 km

Figure 5. Geographical distribution of the Complete Sensor Dataset

Out of the 351 sensors available in the Stuttgart area as of November 11th 2019, only 81 sensors were found to be active through the 5-month threshold period.

3.2 Base sensor selection

In this study, transfer of optimum weights from one sensor to the rest of the sensors in the network is of interest. Thus, tuning of the hyperparameters is only to be done on a single sensor, dubbed the Base sensor. Selection of a base sensor is dependent on the frequency of its recordings, nearness to a weather station as well as closeness to PM_{10} value of a government sensor.

First, the statistical mode is used to filter through the 81 sensors and identify their frequency with respect to the number of recordings.

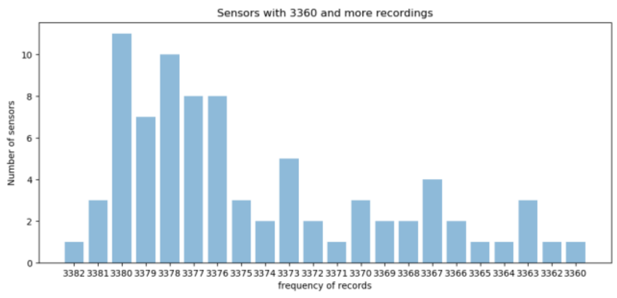


Figure 6. Frequency of recordings

11 sensors, which formed the majority had 3380 recordings. The mode is used as a measure of sensor stability with the recordings consisting of a high number of sensors assumed to be most reliable.

A new proximity analysis is run to locate the government sensors closest to the 11 sensors identified to have 3380 recordings. This is followed by a Pearson correlation coefficient calculation to determine the strength of correlation of the sensors to PM_{10} values of the government sensor. The average of the correlation coefficient is then calculated.

$$\rho LG = \frac{Cov(L,G)}{\sigma L \sigma G} \tag{14}$$

where L = Low-cost sensor PM_{10} reading
G = Government sensor PM_{10} reading
ρLG= Pearson correlation coefficient
$Cov(L, G)$ = covariance between L and G
$\sigma L \sigma G$ = product of the standard deviation of L and G

The proximity analysis reveals that two Government sensors: Stuttgart-Neckartor and Stuttgart Arnulf-Klett-Platz are nearest to the 11 selected sensors. Sensor ID 5127, 2590 and 4827 coefficient of correlation scores are closest to the average coefficient of correlation of all the sensor's PM_{10} readings with that of the government sensors.

Sensor ID	Distance to Gov Sensor(m)	Gov Sensor	P10 ρ	Weather station	Distance to weather station (m)
1434	1574	Stuttgart-Neckartor	0.38	Stuttgart-Ost	698
1148	1455	Stuttgart Arnulf-Klett-Platz	0.55	Stuttgart	1323
5127	1589	Stuttgart Arnulf-Klett-Platz	0.43	Gablenberg	822
757	2140	Stuttgart Arnulf-Klett-Platz	0.55	Degerloch	1916
1282	3838	Stuttgart Arnulf-Klett-Platz	0.37	Büsnau	691
2299	8306	Stuttgart Arnulf-Klett-Platz	0.33	Büsnau	390
2480	7677	Stuttgart Arnulf-Klett-Platz	0.34	Büsnau	3066
2590	7709	Stuttgart Arnulf-Klett-Platz	0.46	Büsnau	2077
3559	2952	Stuttgart Arnulf-Klett-Platz	0.51	Büsnau	530
4827	6086	Stuttgart Arnulf-Klett-Platz	0.47	Sonnenberg	1383
1186	2290	Stuttgart Arnulf-Klett-Platz	0.52	Gablenberg	1670
Average Coefficient of correlation ρ			0.45		

Table 1. Comparison of Sensors with 3380 recordings

Proximity to a weather station was also considered and Sensor ID 5127 emerged as the most suitable of the three due to its closeness to Gablenberg weather station.

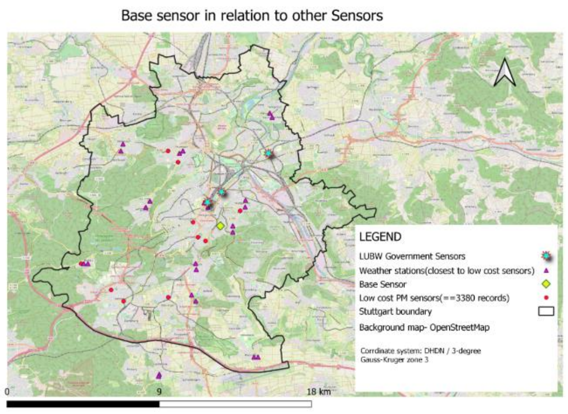

Base sensor in relation to other Sensors
LEGEND
LUBW Government Sensors
Weather stations(closest to low cost sensors)
Base Sensor
Low cost PM sensors(==3380 records)
Stuttgart boundary
Background map- OpenStreetMap
Corrdinate system: DHDN / 3-degree Gauss-Kruger zone 3
0
9
18 km

Figure 7. Base sensor location

3.3 Training of Recurrent Neural Networks

This stage involves training the recurrent neural network to accurately predict future $PM_{2.5}$ values. Training is only done on the base sensor's dataset. The tuned hyperparameters yield optimum weights which are then used to predict particulate matter values for the 81 sensors in the network.

3.3.1 Selection of Hyperparameters: The dataset is split into train and test sets. Sequential learning uses prior information to determine future occurrences and thus, the last records within a time series are normally set aside for validation (Hewamalage et al., 2019). In our dataset, the test sample consists of the last 256 records, corresponding to the first 11 days of November 2019.

The input features include PM_{10}, temperature, humidity, pressure, wind speed, month, date and time. These values are used to predict the output which is $PM_{2.5}$. Normalization is performed on the features to ensure that the calculations between the different entries would have a common denominator. Z score normalization, also known as Standard scalar is used.

$$new\ x = \frac{x-\mu}{\sigma} \tag{15}$$

where x = a record in a feature dataset
μ = mean of the feature dataset
σ = standard deviation

Later, an inverse z-coding procedure will be used to remove the normalization effect and enable the interpretation of predicted values within the real-world scale.

The first RNN variant to be investigated is the standard LSTM utilizing the Adam optimizer. Tensorflow 1.15 (GoogleBrain Team, 2019) is used.

Training involves tuning the hyperparameters to obtain a low Root Mean Square Error and a high R^2 score values. R^2 also known as the coefficient of determination is used to measure the agreeableness of the predicted values to the actual value. It ranges from 0 to 1. The hyperparameters that are considered include the input sequence length, output sequence length, batch size, hidden layers, epoch and learning rate.

To achieve the best quality of results, the output sequence length is set at 1, representing a prediction of 1 hour into the future. This is because we are interested in the comparison of different RNN variants at their best prediction time which is often within a short time step. Prediction quality starts to deteriorate with increasing time steps (Bui et al., 2018). Another hyperparameter that can be calculated is the hidden dimension. This is the size of the hidden layer that is encoded and later decoded during prediction of the outputs. By convention, it is two thirds of the sum of the input and output sequence (Reddy et al., 2017).

The input sequence length, the learning rate, the batch size and the epoch are then tuned iteratively to achieve low RMSE and high R^2 score values. The batch size is an important parameter as it directly affects the number of iterations or in other terms, the number of times new weights are applied and updated by the network. A small batch size equates to a higher number of iterations and by extension, more updates to the weights. The input sequence length refers to how far we want the RNN unit to remember previous states. It contributes to the loss calculation during the forward propagation process. The epoch refers to the number of runs the training data is passed through the RNN framework. In this particular study, a large epoch is needed to ensure that a large proportion of the training dataset is used in the training. The learning rate is then tuned to a positive small value. If the learning rate is set to large values, one risks missing the optimum weights while very small values increase the time needed to train the network. Once the hyperparameters have been set, a tensor flow session is started. A session is used to do the actual training by specifying computer resources and calling the graph function used to model the LSTM and the optimizer. At the moment, Tensor flow allows for only one session to run a single graph function. The following hyperparameters shown in Table 2 are found to be the most optimum to predict 1 hour into the future utilizing hourly dataset from the base sensor.

LSTM with Adam Optimizer	
Hyperparameter	**Value**
Input sequence	20
Output sequence	1
Hidden layer	14
Batch size	6
Epoch	5000
Learning rate	0.01

Table 2. Hyperparameters values

The optimum hyperparameters took 20 seconds to train the model and its predicted values when compared to the actual readings for the first 11 days of November yielded a high R^2 Score of 0.81, with a low RMSE of 2.38.

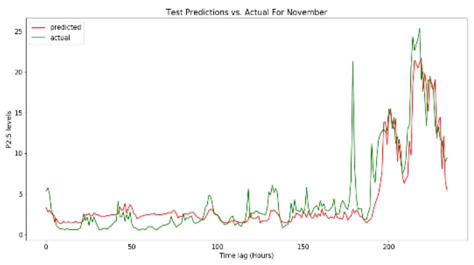


Figure 8. Prediction vs Actual PM 2.5 values using LSTM Adam

3.3.2 Choosing a suitable RNN Variant: The optimal hyperparameters tuned and trained on LSTM-Adam combination are transferred to 5 other RNN variants. These include LSTM with SGD optimizer, LSTM peep-holes with Adam optimizer, LSTM peephole with SGD optimizer, GRU with Adam Optimizer and GRU with SGD optimizer. The base sensor is still used to provide the training and testing dataset. To improve on robustness, each of the 6 variants is trained 100 times and the average RMSE, average R^2 score, average training and testing time recorded.

GRU RNN variant with an SGD optimizer outperformed the others attaining the lowest RMSE and consequently highest R^2 score after the 100 loops and is taken to be robustly the best RNN variant in this study.

RNN Variant + Optimizer	**Average R^2**	**Average RMSE**	**Average Time (seconds)**
LSTM-Adam	0.79	2.45	19
LSTM(Peephole)-Adam	0.79	2.41	26
GRU-Adam	0.80	2.44	21
LSTM-SGD	0.79	2.49	19
LSTM(Peephole)-SGD	0.79	2.53	25
GRU-SGD	**0.82**	**2.27**	20

Table 3. Comparison between different RNN variants and optimizer combinations

3.3.3 Optimum weight determination: GRU-SGD, after 100 loops, resulted in an average R^2 score of 0.82 and an average RMSE of 2.27. The weights leading to these results are not optimum as they have been aggregated. There is a need to locate the best performing weights within the 100 loops. This is achieved by identifying the minimum RMSE and maximum R2 score within the 100 loops, which are found to be weights generating RMSE of 1.98 and an R^2 score of 0.87. The subsequent graph is shown in figure 9.

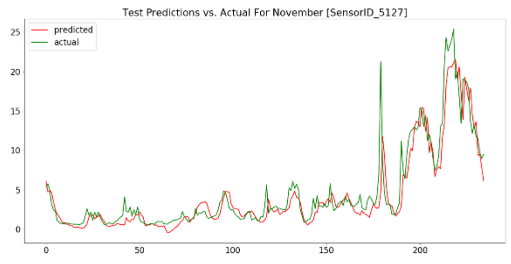
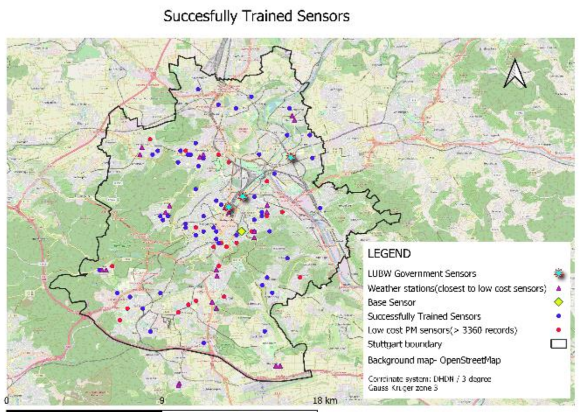


Figure 9. Prediction vs Actual $PM_{2.5}$ values using GRU SGD

While utilizing GRU-SGD, the base sensor's performance improved as illustrated in Figure 9. $PM_{2.5}$ levels are seen to be increasing as summer (May) gives way to winter days (November). This can be an indication of increased motor vehicle use as well as heating during the approaching cold season. GRU-SGD provides for a suitable forecasting model in predicting the first 11 days of November as evidenced by the trend that closely follows the peaks and troughs of the actual data.

3.3.4 Weight transfer: The optimum weights obtained are applied to all sensors in the complete sensor dataset. Out of 81 sensors, 57 sensors had an R^2 score between 0.7 and 1, an interval that agrees with (Amaral et al., 2015) who found out that most manufactured low-cost particulate matter sensor had a similar R^2 score range.


Succesfully Trained Sensors
LEGEND
LUBW Government Sensors
Weather stations(closest to low cost sensors)
Base Sensor
Successfully Trained Sensors
Low cost PM sensors(> 3360 records)
Stuttgart boundary
Background map- OpenStreetMap
Coordinate system: DHDN / 3 degree Gauss Kruger zone 3
0
9
18 km

. Figure 10. Successfully trained sensors using transferred weights.

3.4 Weight transfer investigation

3.4.1 Distance: The geographical distance of sensors relative to the base sensor does not produce a significant effect on the prediction of future $PM_{2.5}$ values after transfer of weights. Sensors further away from the base sensor had a good prediction as sensors closer to the base station. Thus, the weight transfer amongst sensors is not tremendously affected by the distance between sensors.

Fitting a linear regressive curve creates an inverse relationship between prediction success and distance as one moves further away from the base sensor.

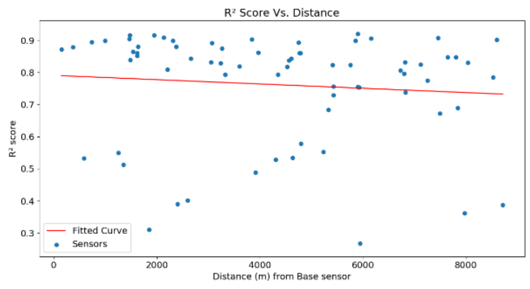


Figure 11. Effect of Distance from Base sensor on R^2 Score

3.4.2 Particulate matter 10: PM_{10} is a strong indicator of the future levels of $PM_{2.5}$. Sensors with a greater than 0.5 correlation of PM_{10} with the base sensor showed a higher prediction score than sensors with less correlation of PM_{10}.

Fitting a curve on the scatter data brought to light a directly proportional relationship between correlation of PM_{10} between sensors and the success in the transfer of weights.

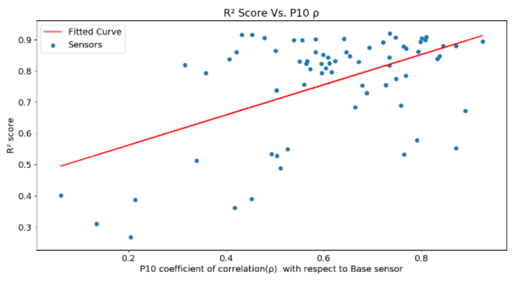


Figure 12. PM10 Correlation and R^2 Score

3.4.3 Particulate matter 2.5: $PM_{2.5}$ is the output of this prediction exercise. Sensors that exhibited a high correlation of $PM_{2.5}$ with the base sensor also fared better when it came to prediction using transferred weights.

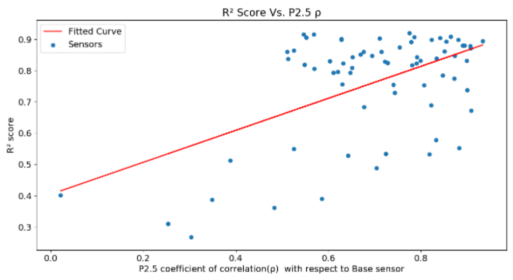


Figure 13. $PM_{2.5}$ Correlation and R^2 Score

3.4.4 Pressure: Pressure is the only weather variable in the study that showed a clear improvement effect on predictions using transferred weights. Overall, all the sensors in the complete sensor dataset had a greater than 0.5 pressure correlation coefficient with respect to the base sensor. However, sensors exhibiting close to 1 correlation with the base sensor performed the best when it came to prediction using transferred weights.

Pressure correlation coefficient showed a directly proportional relationship to the R^2 Score and by extension, transfer of weights.

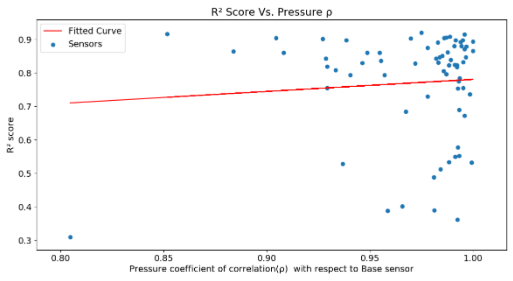


Figure 14. Pressure Correlation and R^2 Score

3.4.5 Temperature, wind direction and speed: The correlation of temperature, wind direction and wind speed of the sensors and the base sensor had little effect on the predictive power of the network.

Temperature correlation coefficient was greater than 0.5 for most sensors but that was not an advantage when it came to prediction using transferred weights. The correlation coefficient of wind direction and wind speed between the sensors and the base sensor was in the medium range and had no effect on their prediction performance

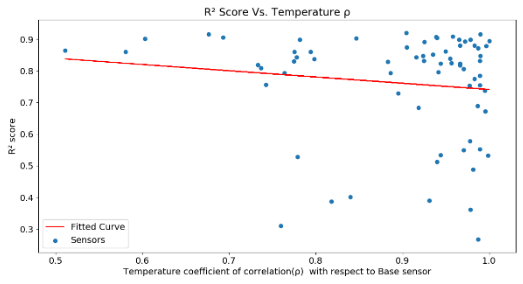


Figure 15. Temperature Correlation and R^2 Score

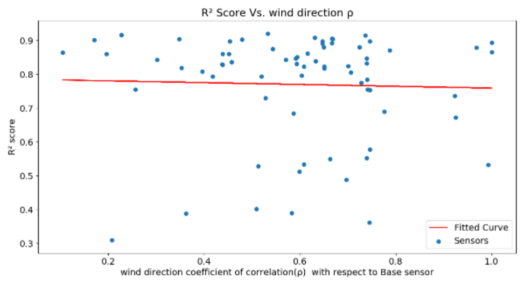


Figure 16. Wind direction Correlation and R^2 Score

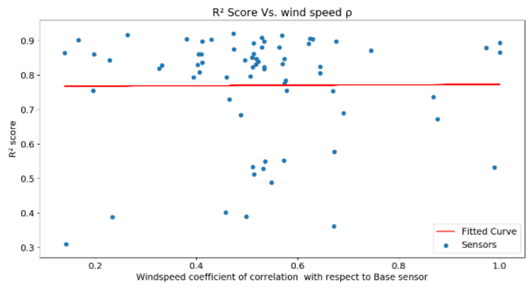


Figure 17. Wind speed Correlation and R^2 Score

3.5 Conclusion

In this study, 3 variants of recurrent neural networks and 2 optimizers resulting to 6 combinations have been evaluated and the Gated Recurrent Unit utilizing the SGD optimizer has been found to perform the best of the 6.

Hyperparameter tuning was successfully done on the LSTM Adam utilizing data from one sensor (the base sensor) and transferred to a different model, the GRU-SGD. The optimal weights obtained from the base sensor were used to predict the first 11 days of November for all the sensors in the dataset. Out of 81 sensors, 57 had an acceptable R^2 Score that ranges between 0.7 and 1.

A high $PM_{2.5}$, PM_{10} and pressure correlation of the sensors with the base sensor has an effect on the transfer of weights as evidenced by the R^2 score.

Distance, Temperature, wind direction and wind speed have little to no effect on the transfer of hyperparameters.

ACKNOWLEDGEMENTS

This study was supported by the Federal Ministry of Education and Research (BMBF) in the project "i_city: intelligent city" under the funding code 13FH9I01lA and supervised by the project management agency VDI Technologiezentrum GmbH on behalf of the BMBF.

REFERENCES

Amaral, S., Carvalho, J., Costa, M., Pinheiro, C., 2015: An Overview of Particulate Matter Measurement Instruments. *Atmosphere 6* (9), pp. 1327–1345. doi: 10.3390/atmos6091327.

Badura, M., Batog, P., Drzeniecka-Osiadacz, A., Modzel, P., 2018: Evaluation of Low-Cost Sensors for Ambient PM 2.5 Monitoring. *Journal of Sensors 2018*, pp. 1–16. doi: 10.1155/2018/5096540.

Bui, T., Le, V., Cha, S., 2018. A Deep Learning Approach for Forecasting Air Pollution in South Korea Using LSTM. https://arxiv.org/ftp/arxiv/papers/1804/1804.07891.pdf (2 October 2019).

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. https://arxiv.org/abs/1406.1078. (11 November 2019).

Gers, F.A., Schmidhuber, J., 2000. Recurrent nets that time and count, 189-194 vol.3. doi: 10.1109/IJCNN.2000.861302.

Google Brain Team, 2019. Tensorflow library. Machine learning library. www.tensorflow.org (1 October 2019).

Hewamalage, H., Bergmeir, C., Bandara, K., 2019. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions. https://arxiv.org/abs/1909.00590(2 September 2019).

Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computations 9 (8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (11 November 2019).

Karagulian, F., Barbiere, M., Kotsev, A., Spinelle, L., Gerboles, M., Lagler, F., Redon, N., Crunaire, S., Borowiak, A., 2019. Review of the Performance of Low-Cost Sensors for Air Quality Monitoring. Atmosphere 10 (9), 506. https://doi.org/10.3390/atmos10090506. (10 February 2020)

Kingma, D.P., Ba, J., 2014. Adam: A Method for Stochastic Optimization. https://arxiv.org/abs/1412.6980 (4th February 2020).

Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Würtemberg. https://www.lubw.baden-wuerttemberg.de/luft/feinstaub-stuttgart?stationId=Vergleich (23 December 2019).

Norvig, P., Russell, S., 2009*: Artificial Intelligence: A Modern Approach*. 3rd ed. New Jersey,Prentice Hall.

OK Lab Stuttgart, 2019. Measure Air Quality Yourself. OK Lab Stuttgart. https://luftdaten.info/en/home-en/ (11 November 2019).

Olah, C., 2015. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (20 November 2019).

OpenWeatherMap, 2019. We Deliver 2 Billion Forecasts Per Day. OpenWeatherMap. https://openweathermap.org/ (11 November 2019).

Pope III, C., Burnett, R., Thun, M., Calle, E., Daniel Krewski, D., Ito, K., Thurston, G., 2002: Lung Cancer, Cardiopulmonary Mortality, and Long-term Exposure to Fine Particulate Air Pollution. https://jamanetwork.com/journals/jama/fullarticle/194704 (15 November 2019).

Python Software Foundation, 2019: Python. Version 3.7.4: Python Software Foundation. https://www.python.org/ (10 November 2019).

QGIS Development Team, 2019: QGIS. Version 3.10.0. Available online at https://www.qgis.org/en/site/ (1 November 2019).

Reddy, V., Yedavalli, P., Mohanty, S., Nakhat, U., 2017. Deep Air: Forecasting Air Pollution in Beijing, China. https://www.ischool.berkeley.edu/sites/default/files/sproject_attachments/deep-air-forecasting_final.pdf (4 October 2019)

Ruder, S., 2016. An overview of gradient descent optimization algorithms. https://ruder.io/optimizing-gradient-descent/ (26 February 2020).

Srinivasan, A., 2019: Stochastic Gradient Descent — Clearly Explained. https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31. (2/25/2020).

Whalley, J., Zandi, S., 2016. *Particulate Matter Sampling Techniques and Data Modelling Methods*. IntechOpen, London, 27 pp.

WHO Regional Office for Europe, 2003. *Health Aspects of Air Pollution with Particulate Matter, Ozone and Nitrogen Oxide*. World Health Organization, 98 pp.

Wissenschaft im Dialog, 2015. Hack Your City. Wissenschaft im Dialog. wissenschaft-im-dialog.de/projekte/hack-your-city/ (22 February 2020).