Air Pollution Prediction Using LSTM Neural Networks

Bojan Evkoski, Zafir Stojanovski, Aleksandar Trajkovski and Dejan Gjorgjevikj
Ss. Cyril and Methodius University
Faculty of Computer Science and Engineering
Rugjer Boskovikj 16, 1000 Skopje, North Macedonia
Email: {bojan.evkoski, zafir.stojanovski, aleksandar.trajkovski}@students.finki.ukim.mk
dejan.gjorgjevikj@finki.ukim.mk

***Abstract*—Air pollution in North Macedonia is 20 times over the EU limit. Recently Skopje is mentioned as the most polluted city in Europe. As a result, this is believed to contribute to 2000 annual premature deaths in Skopje, Tetovo and Bitola only. Being able to forecast air pollution levels to take timely precaution could drastically reduce these numbers. Using state of the art recurrent neural networks known as LSTMs, we were able to predict these levels by combining historical pollution data and weather forecasts through meta models, achieving mean RMSE for all sensors around 20, with the best results having RMSE as low as 8.78, with PM_{10} measurements ranging from 0 to above 1000 and are usually accompanied by a lot of noise. In this paper we present several approaches we have tried for solving the problem and a basic comparison between them and we also propose a way to expand these models into a real-time system for multitarget predictions.**

Keywords— air pollution forecast; LSTM & Meta models; univariate vs. multivariate comparison

I. Introduction

In the late 70's and early 80's, researchers dismissed initial studies indicating that air pollution is directly associated to daily mortality rates [1] due to the fact that these experiments didn't account for cigarette smoking and other health related issues. However, in 1993 a 14-years long U.S. study emerged [2] providing results that despite the inability to dismiss the effects of other unmeasured risk factors with certainty, fine-particulate air pollution does in fact contribute to excess mortality. These particles (aerodynamic diameter less than 2.5 micromillimeters) are thought to pose a particularly great risk to health because they are more likely to be toxic than larger particles and can be inhaled deeper into the lungs.

Recent data released by the World Health Organization [3] show that both prenatal and postnatal exposure to air pollution can negatively influence neurodevelopment, lead to lower cognitive test outcomes and influence the development of behavioral disorders such as autism spectrum disorders and attention deficit hyperactivity disorder. More recently, the Institute of Public Health of Republic of North Macedonia [4] showed that 1,903 human lives (excess deaths) are lost annually due to $PM_{2.5}$ exposures (22.3% of total all-cause (natural) mortality). If the limit values of the $PM_{2.5}$ particles had complied with the existing EU and WHO limit values, 908 lives could have possibly been saved, and 1547 respectively.

Being aware of the possible hazards air pollution imposes, our goal is to explore the ways of efficiently forecasting the pollution levels of these fine particles. In this paper we focus on predicting PM_{10} particles for reasons connected to availability and abundance of data, but since they are highly correlated with the $PM_{2.5}$, that should not be a problem. Ultimately, the mission is to provide a tool that will allow the citizens of Skopje to be in control of how exposed they are to ambient air pollution.

The remainder of the paper is organized as follows: in section II we describe the theoretical background of LSTMs and their advantage to standard RNNs, while section III presents our data and the preprocessing stage. In Section IV we make an overview of the models we are using for prediction and finally in section V and section VI we compare the results, share future plans and conclude the paper.

II. Theoretical Background

In this paper, we propose using an RNN called long short-term memory (LSTM), to analyze time series air pollution in Skopje. The LSTMs take as input not only the current timestep input, but also what they have "perceived" previously in time. While RNNs (Recurrent Neural Networks) are suitable fit for modeling time series data, they have been known to have a serious issue - vanishing gradient [5]. This occurs as result of the many multiplications that occur within the hidden layers of the net, creating derivatives that vanish as they progress through the network while backpropagating. In simple terms, the network learns incredibly slow, or in some cases does not learn at all. On the other hand, LSTMs solve this problem by preserving the error in a gated cell, thus the gradient is calculated very differently from the standard RNN [6]. Since the LSTMs already demonstrated their usability for time series prediction in recent years, the decision to use this algorithm was unequivocal. The comparison between a simple RNN and an LSTM RNN is shown in Figures 1 and 2, respectively.

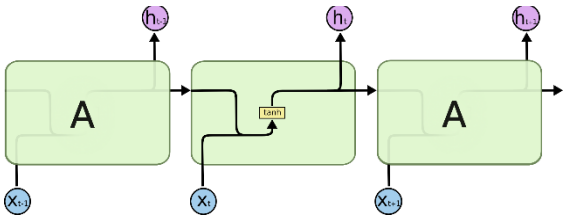

Fig. 1. RNN with one layer and no gated memory cells

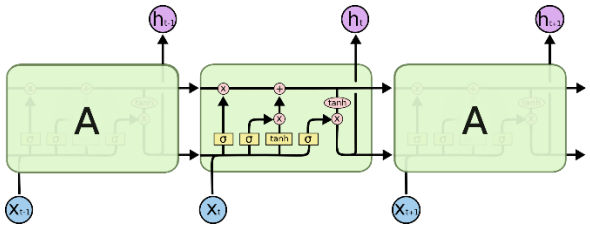

Fig. 2. LSTM RNN with gated memory and sigmoid activation functions

III. DATA

A. *Description*

The data were obtained from two separate sources. First, we queried air pollution data from Skopje Pulse (https://skopjepulse.mk/) generated by six sensors owned by the government located in the following locations around Skopje: Centar, Lisiche, Miladinovci, Karposh, Gazi Baba and Rektorat. Each of these sensors were put into use at different times, with the oldest one being online for 14 years, dating back to 2005. Next, we queried the data from Dark Sky's API (https://darksky.net), acquiring information about Skopje's hourly weather from 2012 until 2019.

B. *Preprocessing*

The preprocessing was done using the Pandas toolkit in Python [7]. Both the air pollution and the weather datasets were hourly, each having timestamp as an index column. The air pollution dataset had a single feature: value of the measured PM_{10} concentration. On the other hand, the weather dataset had the following features: cloud cover, dew point, humidity, temperature, UV index, visibility, wind speed and wind direction. The wind direction attribute contained categorical values such as the following: N (North), NE (North-East), NNE (North-North-East) etc. Instead of simply enumerating the values, we decided to represent this attribute as two separate ones – Wind X and Wind Y, expressing the spatial proximity between the values. If we were to enumerate them, 1 being N and 16 being NNW – we can immediately see the issue: naturally these values are extremely near, but their enumeration values are the furthest apart. Thus, given the spatial arrangement of the wind values, we obtained the X and Y components by appropriately taking the cosine and sine of the angle being formed. Fig.3 shows the circle of wind directions and their sine and cosine mappings.

Finally, we augmented the weather dataset by adding two additional attributes: a boolean indicating if it's a workday or not, and the month. Again, for reasons described above, we transformed the month value as two separate attributes (ex. January is as close to February as it is to December). From now on, we refer to this augmented dataset as *extras*.

C. *Data Partitioning*

Based on the model architectures described below, we split the datasets of all the different sensors in the following manner:

- TRAIN1 – for training the LSTM (from 2005 until 2015)
- TRAIN2 – for training the SVR meta model (from 2015 until 2017)
- TEST – for testing the both LSTM and meta model (only 2018)

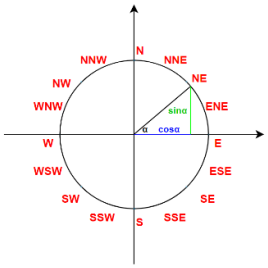

Fig. 3. Wind direction with 2D angle mappings

IV. METHODS

The focus was to train the LSTMs and then combine them into a meta model for every sensor separately. Generally, we can divide our experiments in two separate approaches: Univariate and Multivariate (describing the type of the LSTMs used in the first stage before the Meta models).

A. *Univariate*

In this approach, we used only the sensor values from TRAIN1, without the *extras* to train five univariate LSTMs on all sensors separately, with a time lag of 48 hours, while optimizing the hyperparameters (number of units, hidden layers, activation functions, regularization, optimizers and loss functions) with the help of a validation set extracted from TRAIN1 as a hold-out. Next, we used the models to make predictions for the TRAIN2. We then combined these predictions from the LSTMs with the *extras* dataset acquiring a train set for the Meta model. This train set contained 18 features (12 from the extras and the 6 separate LSTM predictions for all sensors) and one target for every Meta model. For instance, the Meta model for Centar would contain LSTM predictions for Centar, Lisiche, Miladinovci, Karposh, Gazi Baba and Rektorat (along with the *extras*), but the target value would be the actual Centar measurement on the exact timestamp from the sensor. From Fig. 4 and Fig. 5, we see the format of the univariate LSTM train sets and the Meta train sets respectively. Fig. 6. shows the complete architecture of the first approach

All meta models are Support Vector Regressors with a gaussian kernel [10], trained using the Scikit-Learn machine learning library [8] and optimized on their three hyperparameters (C, epsilon and gamma) using 10-fold cross validation.

	Value (t-48)	Value (t-47)	Value (t-46)	Value (t-45)	...	Value (t-3)	Value (t-2)	Value (t-1)	Value
8076	109.0	111.0	84.0	68.0	...	11.0	11.0	16.0	14.0
8077	111.0	84.0	68.0	57.0	...	11.0	16.0	14.0	9.0
8078	84.0	68.0	57.0	59.0	...	16.0	14.0	9.0	9.0
8079	68.0	57.0	59.0	78.0	...	14.0	9.0	9.0	15.0
8080	57.0	59.0	78.0	90.0	...	9.0	9.0	15.0	18.0

Fig. 4. Train data for Univariate LSTM (lag 48)

Cloud Cover	Dew Point	Humidity	Temperature	UV Index	Visibility	Wind Speed	Wind X	Wind Y	Nonwork Day	Month X	Month Y	Target Value (Rektorat)	LSTM Rektorat	LSTM Centar	LSTM Lisiche	LSTM Karposh	LSTM Miladinovci	LSTM Gazi Baba
0.00	40.99	0.40	66.09	4.5	6.22	2.24	0.95	-0.33	0	1.0	0.03	31.11	42.64	44.70	41.51	40.72	32.36	28.93
0.33	40.08	0.34	69.99	5.0	6.22	1.78	-0.41	0.91	0	1.0	0.03	34.20	36.79	39.97	30.21	32.82	29.27	30.32
0.00	37.39	0.29	71.49	3.5	6.22	1.11	0.98	0.19	0	1.0	0.03	27.77	39.28	35.43	15.08	31.20	26.86	17.64
0.19	35.58	0.24	75.07	2.0	6.22	2.24	0.98	0.19	0	1.0	0.03	28.13	33.07	32.85	13.63	29.03	25.05	27.49
0.38	35.73	0.24	74.59	1.0	6.22	1.95	0.33	0.95	0	1.0	0.03	30.88	33.30	30.68	23.22	25.69	21.78	21.03

Fig. 5. Train data for meta model with Univariate LSTM (Rektorat)

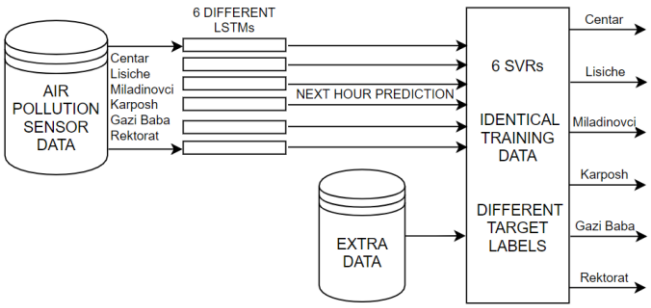

Fig. 6. Univariate architecture

B. *Multivariate*

The idea of the second approach was to use the *extras* from the beginning, by utilizing the multivariate system of the LSTMs implemented in Keras [9]. We used TRAIN1 with the 12 *extras* features along with the sensor values with a time lag of 3, granting a total of 39 features for every input. Since, Keras uses matrix 3D representation for multivariate problems, our vectors had the shape *of (#train_examples, 3, 13).* Having this format, we trained 6 LSTM multivariate models for every sensor and then we combined these into meta models for each respective sensor (Fig. 7). From Fig. 8 and Fig. 9 we see the format of the multivariate LSTM train sets and their respective meta train sets. Even though the two main approaches were quite the opposite, the results from the two separate methods were not significantly different.

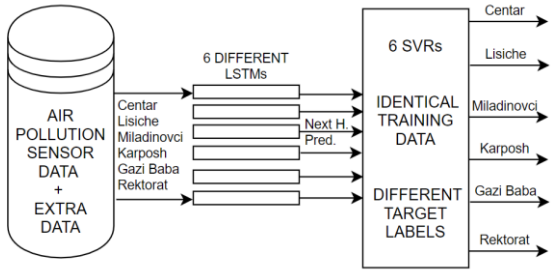

Fig. 7 Multivariate architecture

Cloud Cover (t-3)	Dew Point (t-3)	Humidity (t-3)	Temperature (t-3)	UV Index (t-3)	Visibility (t-3)	Wind Speed (t-3)	...	Wind X (t-1)	Wind Y (t-1)	Non Work Days (t-1)	month_x (t-1)	month_y (t-1)	Value (t-1)	Value
0.00	46.05	0.24	87.48	6.0	6.21	3.46	...	-0.60	-0.80	1.0	0.99	0.14	0.00	38.15
0.00	47.24	0.24	89.27	4.0	6.21	2.87	...	0.97	0.26	0.0	0.99	0.14	38.15	35.88
0.00	47.24	0.23	89.57	2.0	6.21	3.18	...	0.99	0.12	0.0	0.99	0.14	35.88	34.24
0.25	47.65	0.23	90.94	0.0	6.21	8.05	...	0.91	0.42	0.0	0.99	0.14	34.24	66.90
0.00	47.67	0.26	87.26	0.0	6.21	8.65	...	0.89	-0.45	0.0	0.99	0.14	66.90	79.74

Fig. 8. Train data for Multivariate LSTM (lag 3)

Target Value (Rektorat)	LSTM Rektorat	LSTM Centar	LSTM Lisiche	LSTM Miladinovci	LSTM Karposh	LSTM Gazi Baba
134.17	93.82	137.29	231.20	55.95	99.63	196.37
72.37	117.67	123.22	188.94	48.38	114.94	178.55
54.57	83.58	102.41	183.92	44.82	102.69	141.88
58.86	69.12	89.04	183.87	43.87	84.28	105.40
25.63	41.78	88.22	80.08	43.30	65.93	35.06

Fig. 9. Train data for meta model with Multivariate LSTM (Rektorat)

V. Results and Future Work

A. *Results*

RMSE (Root Mean Square Error) results are shown in Table I for every sensor and its respective four main predictions for the next hour: Univariate LSTM, Multivariate LSTM, meta derived from Univariate LSTMs and meta derived from Multivariate LSTMs. We can see that the meta models give a huge boost to both univariate and multivariate approaches for all sensors, while not showing a significant difference between both methods. Since the sensor values vary from 0 to 1000, being able to predict with RMSE around 10 could be very useful. Fig. 10 shows hourly timestep in subset of the predicted year (2018) from the test set for Centar's sensor. It is important to notice that predicting more than one hour ahead is possible in both cases. Univariate multistep prediction would utilize a circular movement going forward with the previous 47 hours plus the new predicted value from the meta, combining them with a forecasted weather data from an API like Darksky. The multivariate approach also allows the circular movement predictions but could easily implement immediate prediction for more hours since the LSTMs use the weather data instead of the meta, having no need for circular patterns forward in time.

TABLE I. Prediction Results for the Next Hour

Sensor	Root Mean Square Error (RMSE)			
	Univariate LSTM	***Multivariate LSTM***	***Univariate Meta***	***Multivariate Meta***
Centar	21.784	21.028	9.872	12.749
Lisiche	37.606	37.957	30.452	31.242
Miladinovci	23.728	24.558	20.528	21.697
Karposh	22.714	24.232	11.34	14.375
Gazi Baba	38.922	39.953	29.213	31.370
Rektorat	32.417	35.063	26.231	28.868

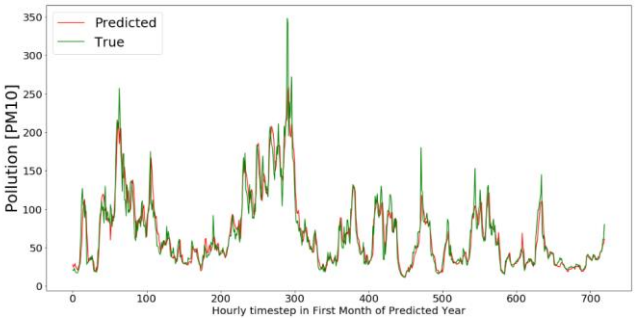

Fig. 10. Meta model Prediction vs. Actual

B. *Future Work*

Finding the optimal way to predict at least 24 hours straight is possible and is crucial for many stakeholders (government, ecologists, non-profit organizations etc.) for managing resources, detecting polluters, measuring successfulness of the safety measures and many more, but also it is important information for the everyday life of the citizens of North Macedonia. There are many experiments that can be done in the future on managing the LSTMs, but also in combining the Meta models in many different manners. If we would use multi-target LSTMs (predicting 24 hours in the future instead of one), we would have multiple outputs of every LSTM model, thus the attribute combinations for the Meta models are huge.

We could experiment in that time interval of 24 hours and find patterns that would be very useful for a precise prediction. For example, we could use future predictions of the LSTMs (10 hours ahead) as an input to the Meta model for just one hour ahead or vice versa.

There is also a room for improvement on the data gathering and quality, since there are many weather attributes that were not available for us for the time being (for example: cumulative rainfall and cumulative snowfall for the past hour, day, week etc.). We still have some ideas for data preprocessing and transforming the date and time parameters into some more useful, but also discovering some new ones too.

Even though the models recognize the overall pattern, the amount of data is a huge factor for the LSTMs, so making a system that can train daily on the new everyday data is our top priority for this project.

VI. Conclusion

Predicting air pollution accurately is possible, especially by using the right weather and air pollution sensor data from robust sensors placed on noiseless surroundings. In this paper, we presented a method for utilizing this correlation between sensors on different location by combining predictions from different models into one with the usage of Meta models. By training these correctors of the LSTMs, we drastically increased the prediction capabilities, thus emphasizing the connection between the measurements of different sensors, gaining RMSE results below 20 for most of the Meta models.

Because long-term prediction tasks are naturally more difficult, they require more relevant historical data, including optimum time lags, which adds another layer of optimization of the LSTM model. This means that many hyperparameters, such as batch size and number of LSTM cells, may still be optimized to return a lower RMSE for longer future forecasting. The hope is that by leveraging as much time series data as possible we can create stronger weights in the RNN, based on the sequence dependencies. As mentioned above, longer prediction times can help cities in policymaking and resource allocation, but more importantly, can help in the main battle against air pollution in order to solve this problem for humanity and all living species on the planet once and for all.

References

[1] Bobak, M & Leon, David. (1992). Air pollution and infant mortality in the Czech Republic, 1986-88. Lancet. 340. 1010-4. 10.1016/0140-6736(92)93017-H.

[2] Dockery, Douglas & Pope, C & Xu, Xuebing & Spengler, Jack & H. Ware, James & E. Fay, Martha & G. Ferris, Benjamin & Speizer, Frank. (1994). An Association Between Air Pollution and Mortality in Six U.S. Cities. The New England journal of medicine. 329. 1753-9. 10.1056/NEJM199312093292401.

[3] World Health Organization. (2018). Air pollution and child health: prescribing clean air: summary. World Health Organization. http://www.who.int/iris/handle/10665/275545. License: CC BY-NC_SA 3.0 IGO.

[4] Dimovska, Mirjana. (2018). Assessing Health Impact of Air Pollution in Macedonian Cities. Biomedical Journal of Scientific & Technical Research. 10. 10.26717/BJSTR.2018.10.001887.

[5] Bengio, Y & Simard, Patrice & Frasconi, Paolo. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council. 5. 157-66. 10.1109/72.279181.

[6] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation.9.1735-80.10.1162/neco.1997.9.8.1735.

[7] Wes McKinney. (2010). Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56

[8] Pedregosa. (2011). Scikit-learn: Machine Learning in Python et al., JMLR 12, pp. 2825-2830,

[9] P.W.D. Charles, Project Title, (2013), GitHub repository, https://github.com/charlespwd/project-title

[10] Basak, Debasish & Pal, Srimanta & Chandra Patranabis, Dipak. (2007). Support Vector Regression. Neural Information Processing – Letters and Reviews. 11.