CS230

Multivariate Time Series Forecasting of Air Pollution Using Deep Learning

Abhishek Kumar
Civil and Environmental Engineering Department
Stanford University
abhi2947@stanford.edu

Ifdita Hasan Orney
Computer Science Department
Stanford University
ifdi1101@stanford.edu

Josue Solano-Romero
Computer Science Department
Stanford University
jsolanor@stanford.edu

Link to Github repo

1 Introduction

Air Pollution is one of the most serious environmental issues facing the world today that severely affects human health. In 2019, 99% of the world population resided in places that violated the air quality standards set by WHO (1). It is considered as the fourth greatest overall risk factor affecting human health globally (2). Among various air pollutants, fine particulate matter with diameter less than or equal to 2.5 micrometers (PM2.5) is the major culprit (3). This project attempts to develop a model that forecasts the PM2.5 concentration in four time steps ahead at different locations of a city to make timely warnings and mitigate the occurrence of severe air pollution events. The input of the model are the concentration of 8 air pollutants (PM2.5, PM10, CO, SO2, NO, NO2, Ozone, NOx) and 5 meteorological parameters (Average Temperature, Wind Direction, Atmospheric Pressure, Solar Irradiance, Relative Humidity) of last D hours at a particular station of a city. The model outputs the PM2.5 concentration of a station, in advance, sequence at time $t + 2$, $t + 3$, $t + 4$, and , $t + 5$ hours, where t is the current hour.

2 Related work

In recent years, there have been many efforts to apply deep learning models to improve the accuracy of PM2.5 concentration predictions. Given the complexity and many factors contributing to air quality prediction, it's difficult to obtain satisfactory results with a single prediction model (3). In general, efforts have shown hybrid deep learning architectures prove to be more effective than single deep learning models. For instance, Zhang, 2020 (3) and Faraji, 2022 (4) both use spatial-temporal methods that use CNN and GRU to predict PM2.5 concentrations. CNN is used to extract spatial features and GRU is used to extract time characteristics (5).

In other approaches, models preform data processing before feeding it to a prediction model. For example, in (Lin, 2021)(5), the authors use GRU to preform predictions, but they create different architectures based on the data process used before inputting the data into a GRU. In their GRUST13d model, they split spatial-temporal data into four categories according to the four seasons for the year. Lin, 2021(5) data process step is a good approach to improving prediction performance without adding complexities to the architecture. We plan to follow a similar approach in our models.

CS230: Deep Learning, Fall 2022, Stanford University, CA.

3 Data Pre-Processing Methods

Delhi, being one of the worst air polluted cities of the world(6), is chosen for the project. The data of 38 air monitoring stations of one year from 01 August 2021 to 31 July 2022 at a frequency of one hour has been web-scrapped from the website of Central Pollution Control Board, India(7). The data-set includes 8760 records of concentration of 8 air pollutants (PM2.5, PM10, CO, SO2, NO, NO2, Ozone, NOx) and 5 meteorological parameters (Average Temperature, Wind Direction, Atmospheric Pressure, Solar Irradiance, Relative Humidity) of each station. Any station with any feature column having more than 13% of the missing values has been dropped. In this consideration, only 50% of the stations have been qualified for the model development.

We employed 'Forward Fill' followed by 'Backward Fill' techniques to handle the missing values in the remaining stations. Each station data has been converted into supervised form for model development. Further, all supervised form 19 stations data have been merged and shuffled, and then normalized input features with min-max technique. We consider 90% of the shuffled data (159,709 examples) to train the network, while 5% (8318 examples) each for validation and testing purposes.

4 Learning Methods

We employed LSTM and CNN-LSTM based deep learning models to forecast PM2.5 concentration. The intuition behind choosing these two kind of models is that particulate matter concentration mainly depends on both spatial and temporal features, and these models are well know to extract information from the data. LSTM has good processing ability for time series data. It can handle multiple input variables perfectly and correlate the contextual information well.

4.1 LSTM

LSTM is a special kind of RNN, which shows outstanding performance on a large variety of problems. LSTMs were designed specifically to overcome the long-term dependency problem faced by RNNs due to the vanishing gradient problem. LSTMs have feedback loop which enables them to retain useful information about previous data points to process the new data points. LSTM uses three gates: Forget gate, Input gate, Output gate. The forget gate decides which pieces of the long-term memory should now be forgotten (have less weight) given the previous hidden state and the new data point in the sequence. Input gate determines what new information should be added to the networks long-term memory (cell state), given the previous hidden state and new input data. Output state decides and outputs the new hidden cell.

4.2 CNN-LSTM

The CNN-LSTM is the integration between the CNN and the LSTM. It combines the advantages of both of the models, leading to great breakthroughs in video recognition and classification, natural language processing and other fields.In the hybrid model of CNN-LSTM, we use the CNN model to extract the spatial features and the LSTM model for interpreting the data across the time steps.

5 Experiments

In this project, we designed, trained, and evaluated 3 different architectures each of LSTM and CNN-LSTM as depicted in Table 2. In total, we ran experiments on 4 different models based on the number of output time steps. We trained each of the architectures that we designed with approximately 160, 000 examples with different time lags. We complied the model with Adam optimizer and Mean Absolute Error loss function. Each experiment tested one of several hyper-parameters such as the number of units in each layer, dropout probabilities, learning rates, and the number of input and output steps. Table 2 displays our best results for each model architecture on $\{T+2, T+3, T+4, T+5\}$ output steps.

In each experiment, we trained models locally on 19 air monitoring stations. In total, we ran 130+ different instances of our models. Each instance had a training time ranging between 10min-6hrs on a standard computer. We set the baseline for our hyper-parameters by tuning a one-layer LSTM model.

5.1 LSTM

In our initial LSTM experimentation, we found the best learning rate to be 0.005. In one of our experiments, we used a one-layer LSTM with 80 units trained on 75 epochs. The lowest RMSE was found at 0.005 (1). After settling on our learning rate, we experimented with the number of units. Ultimately, we found that models with a higher number of units perform better than lower values. With our dataset, we found the best unit size to be around 200 units.

Table 1: Example of experiments run on learning rates

Learning Rate	Forecast Time	RMSE - Train	RMSE - Test
0.001	T + 2	13.60	20.60
0.003	T + 2	13.07	20.03
0.005	T + 2	12.67	19.83
0.007	T + 2	13.03	20.23
0.1	T + 2	20.65	28.31

We also saw an improvement in RMSE as we increased the number of epochs. Although the run-time to train our model increased significantly, $\approx$ 3hrs on a standard laptop. Overall, we settled on a learning rate of 0.005, 200 units, and a dropout rate of 0.5. Using these parameters, we experimented with LSTM - 2L and LSTM - 3L models and tuned parameters such as the number of in and out steps. The results of our experiments are found in Table 2.

5.2 CNN-LSTM

In our experimentation, we tuned parameters for our CNN L1 - LSTM L1. Initially, we used the same hyperparameters from our LSTM-L1 model and focused on finding the best parameters for the CNN layer. We found that the best input size for our CNN layer is around 24. That is, the best time step to consider in each example is 24 hours. We also found the best number of sub-sequences in our CNN layer to be 2, which gives us 12 output steps per sub-sequence.

We settled on a CNN-LSTM model which consisted of 100 units in the CNN layer and 200 units in the LSTM layer. We also decreased the dropout rate to 0.1 since a dropout rate of 0.5 underfits our data. We also included max pooling after the CNN layer to add some regularization in addition to the 0.1 dropout. We also performed similar experiments on other CNN-LSTM architectures, following the baseline for CNN L1 - LSTM L1.

6 Results and Discussions

Table 2 summarizes the best test RMSE of 6 different architectures of LSTM and CNN-LSTM with varying number of layers that are designed to forecast PM2.5 concentration of a particular station in advance at 4 consecutive time steps of one hour each. The simple architectures with lesser layers, in general, tend to perform either better or almost similar to the complex models for a combination of varying sizes of input time steps. For near time steps prediction, particularly for 2 and 3 hours, 1 - Layered LSTM outputs better results than any other structurally complex models made for this study. However, all architectures perform similar to each other in terms of 4 and 5 hours forecasting. Additionally, higher error is observed in the far away time steps than nearby prediction in case of all models.

Table 2: Summary of Test RMSE(ug/m3) Scores on Different Trained Model

Time	LSTM1	LSTM2	LSTM3	CNN1 - LSTM1	CNN1 - LSTM2	CNN2 - LSTM1
T + 2	12.72	22.21	22.49	17.54	24.24	19.80
T + 3	20.48	24.49	24.29	23.47	24.56	24.44
T + 4	24.01	25.28	27.24	25.83	28.74	28.74
T + 5	27.41	27.98	26.99	26.65	29.21	29.64

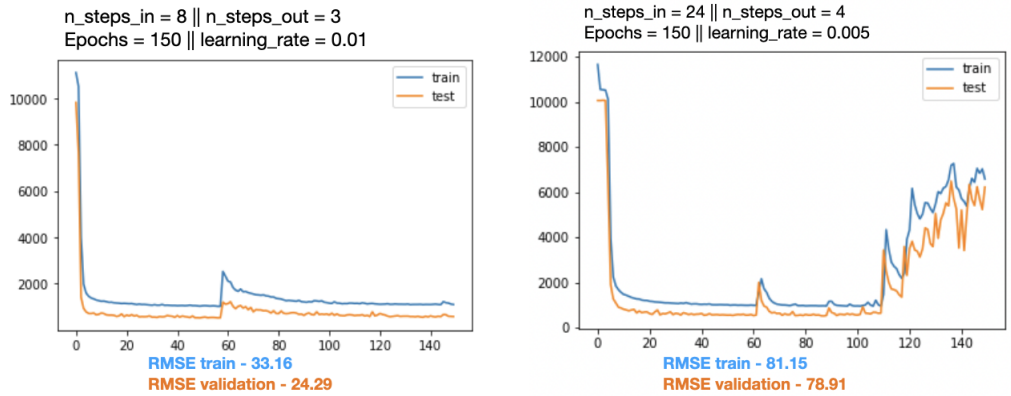
LSTM1: LSTM - 1 Layered, LSTM2: LSTM - 2 Layered, LSTM3: LSTM - 3 Layered, CNN1 - LSTM1: CNN - 1 Layered, LSTM - 1 Layered, CNN2 - LSTM1: CNN - 2 Layered, LSTM - 1 Layered, CNN1 - LSTM2: CNN - 1 Layered, LSTM - 2 Layered

It has been observed that adding more number of input time steps for prediction in the simpler models like LSTM - 1 layered, either enhances the performance insignificantly or degrades the performance. On the other hand, increasing number of input time steps on more complex models result in some unexpected figures. Figure 1 shows two instances of 3 - layered LSTM architectures that show weird results when trained with larger input time steps. The model on the left when feed with 8 input time steps does not improve even iterated for long duration, while the model on the right when input with 24 input time steps for prediction in 4 hours ahead, the loss does not appear to decrease for longer duration and then starts to increase after certain iterations. This behaviour is possibly due to the less dependence of particulate matter concentration on the very long past hours conditions, rather more depend on last 4 - 5 hours of meteorological conditions and other pollutants concentration.

Sheng et al.(8) also observed similar results indicating that a tiny time lag does not guarantee sufficient long-term memory input, while a large time lags does not necessarily improves the performance. Since large input time steps require more time to train and computationally expensive, it is essential to choose an optimum number for the better performance. In this study we observed that most of the instances of the trained models give better results for 4 and 5 input time lags.

On comparing LSTM and CNN-LSTM holistically, LSTM has slightly better results than CNN-LSTM at each time forecast. The probable reason could be that the dataset applied is integrated time series data, and then it is converted into a supervised dataset, indicating that CNN may not be skilled at processing time-series data. Utilizing CNN to handle the data initially destroys the inherent attributes of the data, which has a certain impression on the extraction of data features by LSTM following.

Figure 1: 3 - Layered LSTM Models with large input time steps



7 Evaluation - IFDITA + ABHI

In our evaluation, we used 5% of our data set. While evaluating our model, we recorded the trends and similarities between predicted and observed data. Evaluation parameter we used is the Root Mean Square Error (RMSE). RMSE is the metric of disparity between the values a model predicts from the modeled environment and the observed.

$$RMSE = \sqrt{\frac{1}{m}\sum_{t=1}^{m}(y_o(t) - y_p(t))^2}, \text{ where } y_o = \text{observed value and } y_p = \text{simulated value}$$

RMSE compares an expected value to an observed or known value. This is a fair indicator of error in overall prediction. RMSE represents deviation, hence lower the value of RMSE, higher the accuracy of the model.

We have compared the performance of our best architectures of each time steps with the two existing state-of-the-art time series models to verify the effectiveness of our designed novel architectures. As can be viewed from Table 3, the best models on each time step have better results than Multi-Output and Multi-Index of Supervised Learning Model (MMSL)(8), but 5 hours forecasting model has more RMSE than a deep spatial-temporal ensemble (STE) model developed by Wang et al.(9)

Table 3: Comparison of prediction performance between our models and other models

Model (PM2.5)	Forecast Time	RMSE - Others	RMSE - This Study
MMSL(8)	T + 2	20.22	**12.72**
	T + 3	23.65	**20.48**
	T + 4	26.02	**24.01**
	T + 5	29.76	**27.42**
STE(9)	T + 5	22.97	27.42

8 Conclusion and Future Works

This study attempts to explore different LSTM and CNN-LSTM architectures to forecast multiple outputs of PM2.5 concentration in advance at various time steps of a monitoring station in a city, utilizing various air pollutants concentration and meteorological parameters data of Delhi, the capital city of India. The best architectures, especially the models that output fewer time steps, perform better than some of the existing baseline models. However, some improvements are still required in the models that forecast high number of output time steps (like the models that forecast 4 and more time steps) to achieve the state-of-the-art results.

Given more time, we would like to explore different prepossessing data techniques. For instance, to train our CNN-LSTM models, we used season features (Summer, Monsoon, Post-monsoon, and Winter). We grouped all stations into one dataset and trained on a shuffled dataset. Instead, we would like to sort our data based on time to mimic real-world performance. We would train on examples before some date and use all examples after this date in our validation and test sets.

Since our dataset only spans one year, we decided to use random shuffling because of concerns with biases in our data. If we were to split by time, there would be more training examples labeled winter (Dec-Apr). Furthermore, applying some SQL analysis, the average PM2.5 across all seasons is about 145ug/m3 (winter), 60ug/m3 (summer), 40ug/m3 (monsoon), and 165ug/m3 (post-monsoon). If we used a 90/5/5 split, our validation and test sets would come from summer and monsoon seasons. Therefore, our training examples would include examples that have higher PM2.5 values compared to the validation and training.

Additionally, we would like to consider data from other cities as well for the model development to make the model more general to forecast PM2.5 concentration at any location of the globe.

9 Contribution

All members contributed equally to the project.

References

[1] "Ambient (outdoor) air pollution," Sep 2021. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health

[2] T. Yang, K. Zhou, and T. Ding, "Air pollution impacts on public health: Evidence from 110 cities in yangtze river economic belt of china," *Science of The Total Environment*, vol. 851, p. 158125, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S004896972205224X

[3] Q. Zhang, S. Wu, X. Wang, B. Sun, and H. Liu, "A pm2.5 concentration prediction model based on multi-task deep learning for intensive air quality monitoring stations," *Journal of Cleaner Production*, vol. 275, p. 122722, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959652620327694

[4] M. Faraji, S. Nadi, O. Ghaffarpasand, S. Homayoni, and K. Downey, "An integrated 3d cnn-gru deep learning method for short-term prediction of pm2.5 concentration in urban environment," *Science of The Total Environment*, vol. 834, p. 155324, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048969722024172

[5] C.-Y. Lin, Y.-S. Chang, and S. Abimannan, "Ensemble multifeatured deep learning models for air quality forecasting," *Atmospheric Pollution Research*, vol. 12, no. 5, p. 101045, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1309104221001057

[6] D. Pruthi and Y. Liu, "Low-cost nature-inspired deep learning system for pm2.5 forecast over delhi, india," *Environment International*, vol. 166, p. 107373, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0160412022003002

[7] "Central control room for air quality management - all india," Oct 2022. [Online]. Available: https://app.cpcbccr.com/ccr/#/caaqm-dashboard-all/caaqm-landing/data

[8] D. Seng, Q. Zhang, X. Zhang, G. Chen, and X. Chen, "Spatiotemporal prediction of air quality based on lstm neural network," *Alexandria Engineering Journal*, vol. 60, no. 2, pp. 2021–2032, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016820306438

[9] J. Wang and G. Song, "A deep spatial-temporal ensemble model for air quality prediction," *Neurocomputing*, vol. 314, pp. 198–206, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218307859