Exploring Transformers for Urban Air Quality Inference

Hitarth Gandhi
hitarth.g@iitgn.ac.in
IIT Gandhinagar

Ishan Prayagi
sunil.pi@iitgn.ac.in
IIT Gandhinagar

Somesh Pratap Singh
somesh.ps@iitgn.ac.in
IIT Gandhinagar

Vishal Soni
jayesh.s@iitgn.ac.in
IIT Gandhinagar

Xhitij Choudhary
xhitij.cm@iitgn.ac.in
IIT Gandhinagar

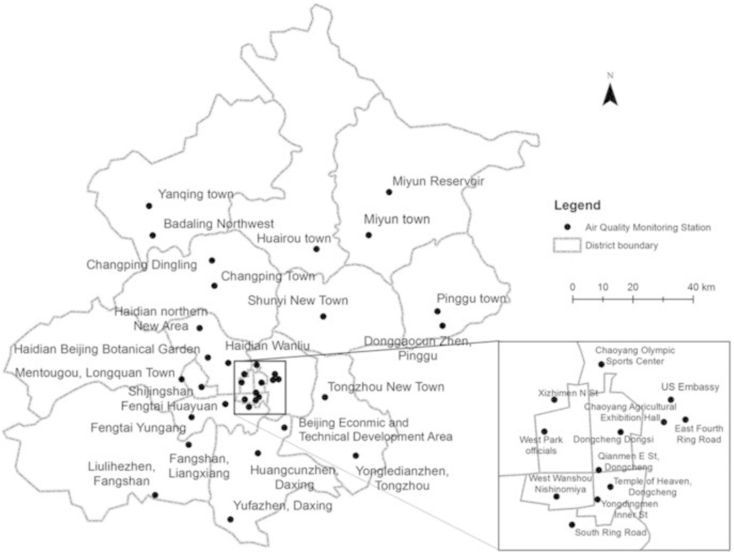

Figure 1: Beijing has a total of 36 Air Quality Inference Stations which are sparse and non-uniformly distributed.

ES 654 Machine Learning, Spring 2022, IIT Gandhinagar

ABSTRACT

Growing air pollution and related environmental and health hazards make the problem of Air Quality Inference of utmost importance. Facilities are set up in several regions to infer the said air quality data. However, they are limited in number over the vast expanses of countries, given the cost limitations in setting them up. Hence, we need to develop a method to predict air quality data of regions where AQI inference stations are not available. To solve this problem, we propose a complex neural approach by combining Transformers and Fully Connected Neural Networks (FNNs). The Transformer layer will capture long-range dependencies among the temporal features. There are two transformers each for capturing temporal dependencies between features of local station and remote stations. Then a FNN to combine the outputs of the transformer and make final prediction of $PM_{2.5}$ values.

KEYWORDS

Air quality, Air Quality Inference, Interpolation, Neural Networks, Transformers

1 INTRODUCTION

Air pollution is a serious environmental health hazard. Almost 9 out of 10 people living in urban areas are affected by air pollution. It can be detrimental to lung development and lead to other respiratory diseases like asthma, emphysema, and chronic obstructive pulmonary disease (COPD).[1]

The air quality index (AQI) is widely used to measure air quality. For a specific air pollutant, its individual air quality index (IAQI) in an area is measured by a monitoring station, reflecting the real-time concentration of the pollutant. AQI is the highest IAQI values among all kinds of air pollutants.

The Air Quality Index (AQI) of several places in India has crossed 400, whereas safe levels are in the range 0-100.[2,3]

The cities, which are heavily polluted, have lesser number of air quality inference stations.[4] These areas have high populations in the range of 0.5 to 2 million which are affected by this perilous pollution.[5]

[1] https://www.niehs.nih.gov/health/topics/agents/air-pollution/index.cfm
[2] https://www.airnow.gov/aqi/aqi-basics/
[3] https://theprint.in/india/delhi-records-worst-air-quality-in-five-years-a-day-after-diwali/761984/
[4] http://cpcbenvis.nic.in/airpollution/monetoring.htm
[5] https://www.census2011.co.in/census/city/41-hisar.html

The deployment of more stations is one visible solution to the problem. However, due to several limitations, there is a plateau in the number of stations that can be set up. Thus, it becomes essential to develop solutions to predict AQI data of regions with sparse AQI inference stations deployment. This will help residents take necessary steps in a timely manner to stay safe if the levels were to exceed safe levels.

2 RELATED WORK

Past approaches include using classical emission models to simulate the flow of air particles based on numerous empirical assumptions and parameters. These works include Gaussian Plume Modeling[1] and Computational Fluid Dynamics[5]. These models utilize many ideal world approximations and assumptions which are not true in the real world. Thus, they give very unsatisfactory results.

Other approaches include Statistical modeling such as mathematical formulations, regression models, and Neural networks. These methods include

- Linear Interpolation, where we take a weighted average of IAQI values from all the stations. The weights assigned to each station are inversely proportional to the distance of the station.
- Gaussian Interpolation. Interpolation based on Gaussian distribution.
- Gaussian Process Regression, This is a non-parametric Bayesian regression model.
- Feedforward Neural Networks, a simple Neural Network with dropout and L2 normalization, is used to predict AQI values. For sequential features, only the latest values are used.
- Support vector regression and Long short-term memory based on Deep Learning are used to classify the AQI values.[3]
- U-Air[7], a classification model which contains two classifiers each for spatial features and temporal features. Then the model combines output from both classifiers to infer AQI. But this separation of features may cause the model to overlook dependencies between spatial and temporal features.
- ADAIN[2], an RNN-FNN hybrid model which contains two groups of input layers each for spatial and temporal features. It also contains an attention layer to learn the importance of different stations automatically over using the k-nearest neighbors method for the same like in some previous papers. This is currently the state-of-the-art model in predicting the AQI.

There are several other works that solve similar problems like predicting the flow of crowd in a city[6] and real-time public transportation-based crowd prediction systems[4]. But these works can not be extended to predicting AQI due to the fine-grained nature of air particles.

3 PROBLEM STATEMENT

Given a set of AQ monitors $\mathcal{S}$, timestamps $\mathcal{T}$, features (latitude, longitude, temperature, humidity, weather, wind speed and wind direction) and corresponding $PM_2.5$ observations, the aim is to predict $PM_2.5$ at a new set of locations $\mathcal{S}^*$ for the same $\mathcal{T}$ timestamps.

4 APPROACH

Link to our GitHub Repo

4.1 Dataset Creation

We used the data given in Zheng et al [8], collected as a part of Urban Air Project 2012, Microsoft research team. It is spanned over a year and was consisting of five parts: City Data (43 x 6), District Data (380 x 4), Air Quality Monitoring Station Data (437 x 6), Air quality data (2891393 x 8), and Meteorological data (1898453 x 8).

Since we needed to look at data pertaining to the city Beijing, we picked the first of the two city clusters where Beijing lied. From the district data, we picked only the districts inside the city Beijing. We then extracted all the stations lying in all the districts in Beijing city. We obtained a total of 36 stations lying in Beijing.

We then performed several data preparing steps: stored the CSVs in separate Pandas DataFrames, dropped the chinese and English names from the dataset since they were of no importance while predicting, joined the station and air quality data using station id, and joined the station and district data by assigning district meteorological data to the stations located in them. We then set time and station id both combined as the index of the DataFrame.

Now, since our dataset was missing some timestamps, we created an empty DataFrame with all the time values corresponding to all the stations and then filled the available values we had from the previous DataFrame into this new DataFrame leaving the unavailable values as NaN. This ensured that we have all the time values in our DataFrame even though their corresponding row entries would be NaN. Since we needed some values to be there in order to make our predictions in place of the NaN values, we tried different interpolation methods on the dataset of which Quadratic Interpolation gave the best results on the baselines. So, we finally interpolated our data using Quadratic Interpolation. Same way, we also added latitude and longitude values from the previous DataFrame into the final DataFrame using stored values from dictionaries of latitude and longitude made from the previous DataFrame.

We then one-hot encoded the categorical data in the dataset and divided the data into three folds, each having 12 test stations and 24 train stations. Then, for every fold, we perform the following dataset pre-processing: Keep one of the 24 train stations as a local station and the remaining 23 stations as remote stations. Repeat this for all the 24 train stations as local stations once. During this, we take a bucket of 24 hours as a single sequence that we will feed to our transformers. Finally, feed the data from the local stations into the local transformer and that from the remote stations into the remote transformer.

4.2 Baselines

We compare our method to the baselines that have previously been used as key techniques in the literature. We had also created a baseline for SVM.SVR (Support Vector Regressor) but did not include it as the results were poorer in that case.

- K - Nearest Neighbors (KNN) regression:
  KNN assumes that the new case and existing cases are comparable, and then assigns the new case to the category that is most similar to the existing categories. It stores all of the available data and classifies a new data point based on its resemblance to the existing data.
- Random Forest Regressor:
  Random Forest Classifier is based on ensemble learning, which is a method of integrating several classifiers to solve a complex problem and increase the model's performance. In exact terms it uses a number of decision trees on different subsets of a dataset and averages the results to enhance the dataset's predicting accuracy.
- XGBoost:
  XGBoost or Extreme Gradient Boosting is a technique developed by University of Washington academics. It uses a gradient boosting framework. It iteratively combines results from weak estimators.
- Inverse Distance Weighting:
  Interpolation using Inverse Distance Weighting (IDW) is mathematical (deterministic), presuming that closer values are more connected to the function than further values that is in this method, the assigned values to unknown points are calculated with a weighted average of the values available at the known points.
- MLP Regressor:
  MLP Regressor trains iteratively and computes the partial derivatives of the loss function with respect to the model parameters to update the parameters. In the output layer of this MLP regressor, there is no activation function. It can be used for both single and multiple target value regression.
- Linear Regression:
  Linear Regression is a statistical technique for performing predictive analysis. Linear regression makes predictions for continuous/real or numeric variables. Linear Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

4.3 Transformers: A brief explanation

Transformer is a two-part architecture for transforming one sequence into another, although it varies from previously existing sequence-to-sequence models in that it does not utilize Recurrent Networks.The two parts of the Transformer are the encoder and the decoder. Both Encoder and Decoder are made up of modules with Multi-Head Attention and Feed Forward layers that can be layered on top of one another several times.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Q is a matrix containing the query (vector representation of one word in the sequence), K all the keys (vector representations of all the words in the sequence), and V all the values (vector representations of all the words in the sequence). V is the same word sequence as Q for the encoder and decoder, multi-head attention modules. V, on the other hand, differs from the sequence represented by Q in the attention module, which takes into consideration the encoder and decoder sequences. This indicates that the weights 'a' are determined by how each word in the sequence (represented by Q) is influenced by the rest of the sequence's words (represented by K). The SoftMax function is also applied to the weights 'a', resulting in a distribution between 0 and 1.

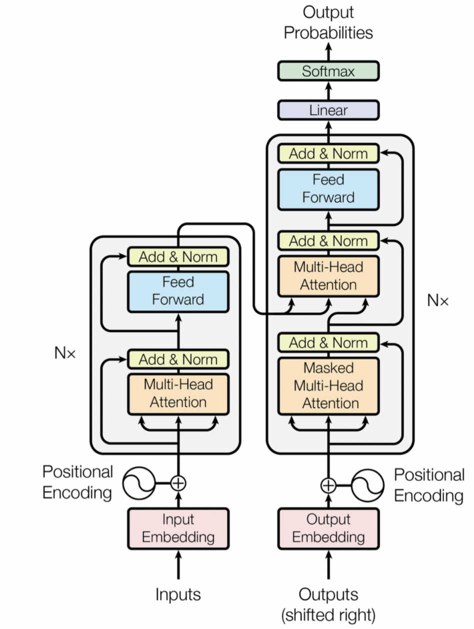

Figure 2: Model architecture of transformer [6]

With linear projections of Q, K, and V, the attention mechanism is repeated several times. This allows the system to learn from a variety of Q, K, and V representations, which is useful to the model. These linear representations are achieved by multiplying Q, K, and V by weight matrices W learnt during training. The encoder and decoder are connected by a multi-head attention module, which ensures that the encoder input sequence is taken into account along with the decoder input sequence up to a certain point. A pointwise feed-forward layer follows the multi-attention heads in both the encoder and decoder. This small feed-forward network has identical parameters at each position, and each element from the given sequence may be characterized as a distinct, identical linear transformation [7], [8], [9].

4.4 Model Architecture

Link to our GitHub Repo

We predict the PM 2.5 value for a station based on data of the last 24 hours. This time frame of 24 hours is referred to as time_window. We have 7 meteorological features, which after one hot encoding become 30. This is referred to as met_features.

For each local station in training, we have 23 remote stations. This

[7] https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04
[8] https://machinelearningmastery.com/the-transformer-model/
[9] https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/

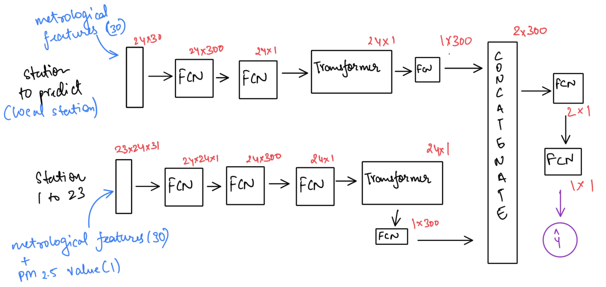

Figure 3: Our Model Architecture

Table 1: Comparing performances of various baselines amidst themselves and with our model (Test Loss)

Model	RMSE loss
KNN	55.041
RF	29.590
LR	29.590
IDW	55.041
XGBoost	32.179
MLP	44.761
Transformer model implemented by us	4.367

is referred to as remote_stations.
For Local Stations, input is (batch_size, time_window, met_features)

- First, we apply a series of FNNs on the meteorological features of input to learn the dependencies among them. Thereby converting the size of input to (batch_size, time_window)
- Then pass this into a transformer to learn the temporal dependencies. The size of the output will be (batch_size, time_window)
- After this, an FNN layer will convert this into (batch_size, 300)

For Remote Station input is (batch_size, remote_stations, time_window, met_features)

- First, we apply a series of FNNs on meteorological features of input to learn the dependencies between them. Thereby converting the size of input to (batch_size, remote_stations, time_window)
- Then apply a FNN on remote stations to learn dependencies between them. Thereby converting the size to (batch_size, time_window).
- After this, pass this into the transformer to learn temporal dependencies. The size of output is (batch_size, time_window)
- Finally, a FNN layer will convert this into (batch_size, 300)

Now, we concatenate the outputs of the transformers. Then apply a series of FNNs to get the final prediction for $PM2.5value$.

5 RESULTS AND DISCUSSION

Final train_loss after 10 epochs is: 4.4050

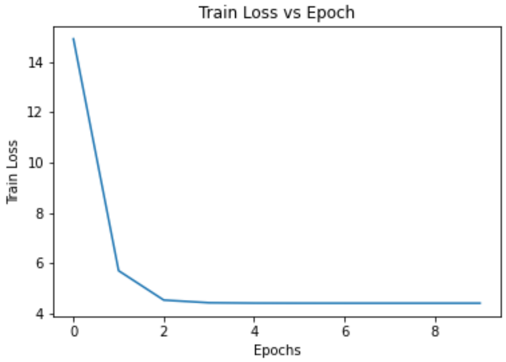

Figure 4: Plot of train_loss as the model is trained over 10 epochs

5.1 Inferences and observations

- As seen from the observation table, the Transformer model implemented by us outperforms all the baselines mentioned above this happens due to its significantly high capability of learning temporal dependencies in the data as compared to the baselines.
- We experimented with different types of scalings, and found out that StandardScaling and RobustScaling does not work with our model because they scale data into ranges whose lower limit is less than -1. This is because we have an Embedding layer in our transformer which cannot accept inputs less than -1. Thus we needed to use MinMaxScaling, which scales the data to $[-1, 1]$.

6 LIMITATIONS AND FUTURE WORK

- Due to computational limitations, we could run our train model for the data of only one month.
- In the current work, only those types of scalings work which do not scale the values below -1, due to the presence of an Embedding layer in transformer which can not accept inputs lesser than -1.
- In the current work, we haven't included distance between the stations. We can add a FNN for combining the transformer output and distances before making the final prediction, thus learning the spatial dependencies in the data.
- A GNN layer could be added to capture the spatial dependencies in the data instead of just using the distance as spatial dependencies.

7 ACKNOWLEDGEMENTS

We are grateful to Prof. Nipun Batra for giving us an opportunity to pursue this project under the course ES 654 in IIT Gandhinagar. We would like to thank our mentors, Harsh Patel and Palak Purohit, for guiding us in the project and their continuous help and support throughout the semester.

REFERENCES

[1] MM ABDELWAHAB, Khaled SM Essa, HM Elsman, A Sh Soliman, SM Elgmmal, and AA Wheida. 2014. Derivation of the Gaussian plume model in three

dimensions. *MAUSAM* 65, 1 (2014), 83–92.

[2] Weiyu Cheng, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. 2018. A neural attention model for urban air quality inference: Learning the weights of monitoring stations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[3] R Janarthanan, P Partheeban, K Somasundaram, and P Navin Elamparithi. 2021. A deep learning approach for prediction of air quality index in a metropolitan city. *Sustainable Cities and Society* 67 (2021), 102720.

[4] Victor C Liang, Richard TB Ma, Wee Siong Ng, Li Wang, Marianne Winslett, Huayu Wu, Shanshan Ying, and Zhenjie Zhang. 2016. Mercury: Metro density prediction with recurrent neural network on streaming CDR data. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 1374–1377.

[5] H Scaar, T Teodorov, T Ziegler, and J Mellmann. 2012. Computational fluid dynamics (CFD) analysis of air flow uniformity in a fixed-bed dryer for medicinal plants. In *Ist International Symposium on CFD Applications in Agriculture 1008*. 119–126.

[6] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-first AAAI conference on artificial intelligence*.

[7] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. 2013. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1436–1444.

[8] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. 2015. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2267–2276.