

ES 654 Project :

Exploring Transformers for Urban Air Quality Inference

Hitarth Gandhi	19110087
Ishan Prayagi	19110194
Somesh Singh	19110206
Vishal Soni	19110207
Xhitij Choudhary	19110044

Mentors: Harsh Patel | Palak Purohit

[Our GitHub Repository](#)



Contents

1. Quick Recap from phase 1 and 2
2. Preparing Dataset For Transformer based Model
3. Our Model Architecture
4. Results
5. Discussion
6. Future Work and Limitations

[Our GitHub Repository](#)



Quick Recap of phase 1 and 2





Problem statement





Problem statement

Given a set of AQ monitors S , timestamps T , features (latitude, longitude, temperature, humidity, weather, wind speed and wind direction) and corresponding $\text{PM}_{2.5}$ observations, the aim is to predict $\text{PM}_{2.5}$ at a new set of locations S^* for the same T timestamps.



Related work





Past approaches to the problem

Past approaches to the problem statement are classified into two categories:

1. Classical emission models that simulate the flow of air particles based on empirical assumptions and parameters.
2. Statistical modeling: mathematical formulations, regression models, neural networks

Former approach of Classical emission models takes various assumptions and often overlooks many real world scenarios giving unsatisfactory results.



Data Description





Data Description

1. We are using the data given in [Zheng et al.](#), collected as a part of Urban Air Project 2012, Microsoft research team.
2. The data we are using is collected over a span of one year and consists of six parts:
 - a. City data (43 x 6)
 - b. District data (380 x 4)
 - c. Air quality monitoring station data (437 x 6)
 - d. Air quality data (2891393X8)
 - e. Meteorological data (1898453X8)
 - f. Weather forecast data: We do not use this for our work.



Data Preparation and Preprocessing





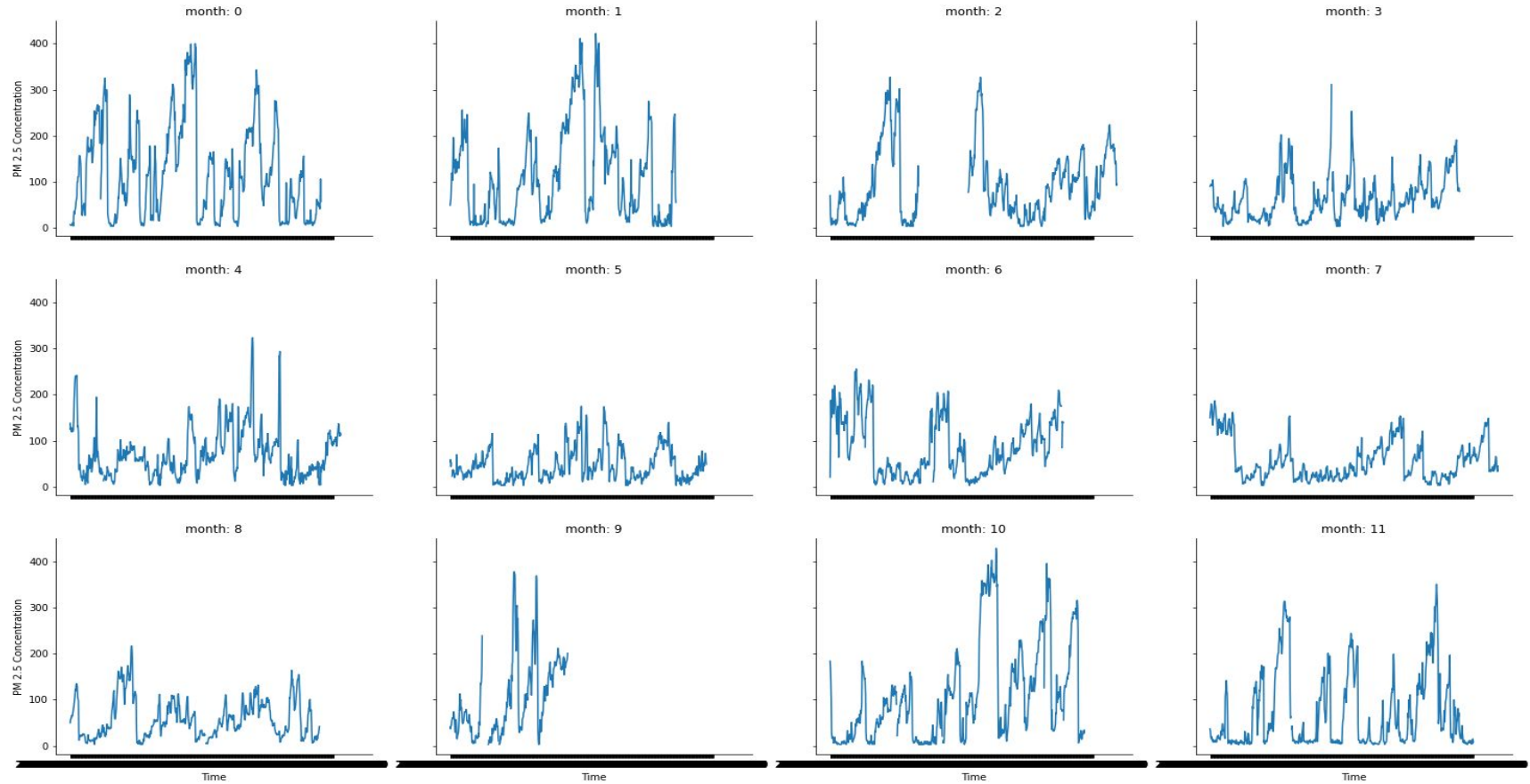
Prepared Dataset

	station_id	time	PM25_Concentration	latitude	longitude	district_id	weather	temperature	pressure	humidity	wind_speed	wind_direction
0	1001	2014-05-01 00:00:00	138.0	40.090679	116.173553	101	0.0	20.0	1004.0	56.0	7.92	13.0
1	1001	2014-05-01 01:00:00	124.0	40.090679	116.173553	101	0.0	18.0	1004.0	64.0	7.56	13.0
2	1001	2014-05-01 02:00:00	127.0	40.090679	116.173553	101	0.0	18.0	1004.0	70.0	5.76	13.0
3	1001	2014-05-01 03:00:00	129.0	40.090679	116.173553	101	0.0	17.0	1004.0	74.0	6.12	13.0
4	1001	2014-05-01 04:00:00	119.0	40.090679	116.173553	101	0.0	17.0	1004.0	75.0	4.68	1.0

Data Visualization



PM25 vs time plots for station 1001 for all months





Baselines





Baselines Implementation results

- Quadratic Interpolation

	KNN	RF	LR	IDW	XGBoost	MLP
Unscaled	53.21939056	29.61174825	38.46993201	53.21939056	32.00670839	36.10281415
Min-Max	55.04081392	29.59019946	29.59019946	55.04081392	32.17881524	44.76072357
Standard	55.04081392	29.58144378	29.58144378	55.04081392	31.72306638	44.68022807



Preparing Dataset For Transformer based Model





Preparing Dataset For Transformer based Model

- Since our dataset was missing some timestamps, we created another dataframe having all the time values and filled it with all the available data from the previous dataframe and leaving the unavailable data as NaN values.
- Now, we tried different interpolation methods on this dataset and again found out that quadratic interpolation worked best on the data upon testing on the baselines so we went forth with quadratic interpolated data.
- We then one-hot encoded the categorical data and divided the dataset into three folds, each having 12 test stations and 24 train stations.
- For every fold, we performed the following pre-processing:



Final Pre-processing to feed the transformers

- Iterate over the 24 train stations, every time, keeping one of them aside as a local station and the remaining ones as the remote stations.
- All the data in these two groups of local and remote stations will be in buckets of 24 hours as a single sequence that we will feed to our transformers.
- Feed the data from the local stations to the local transformer and that from the remote stations to the remote transformer.

Dataset Snippets

```
1 pd.DataFrame(station_metaq[0][0])
```

	0	1	2	3	4	5	6	7	8	9	...	21	22	23	24	25	26	27	28	29	30
0	0.4	0.410869	0.613001	0.752038	0.608949	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1	0.4	0.410869	0.613001	0.752038	0.667113	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.4	0.410869	0.612719	0.770125	0.594268	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.2	0.352174	0.612734	0.818357	0.549658	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	0.2	0.352174	0.608970	0.824386	0.546391	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
5	0.2	0.352174	0.613218	0.848502	0.563379	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.3	0.381521	0.612735	0.842473	0.572120	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
7	0.3	0.381521	0.617837	0.818357	0.525571	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
8	0.4	0.410869	0.616283	0.806299	0.525638	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
9	0.5	0.440217	0.616282	0.788212	0.598983	1.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.4	0.410869	0.616282	0.776154	0.581870	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.4	0.410869	0.616282	0.818357	0.592822	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
12	0.7	0.498913	0.612517	0.739980	0.549382	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
13	0.9	0.557608	0.606671	0.661603	0.548117	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
14	0.9	0.557608	0.596310	0.631458	0.571116	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
15	1.0	0.586958	0.468025	0.619400	0.492535	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
16	0.9	0.557608	0.586433	0.631458	0.569905	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
17	0.9	0.557608	0.586658	0.613371	0.626994	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.9	0.557608	0.586941	0.643516	0.618081	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.8	0.528260	0.431617	0.685719	0.635194	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
20	0.7	0.498913	0.593110	0.715864	0.611620	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
21	0.7	0.498913	0.602857	0.727922	0.624107	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
22	0.6	0.469565	0.603139	0.752038	0.628346	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
23	0.5	0.440217	0.603147	0.794241	0.628499	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

24 rows x 31 columns

Remote Data

```
1 pd.DataFrame(local_met[0])
```

	0	1	2	3	4	5	6	7	8	9	...	20	21	22	23	24	25	26	27	28	29
0	0.373683	0.616349	0.814995	0.502644	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1	0.373683	0.616349	0.814995	0.508240	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.373683	0.626492	0.842279	0.529537	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.311402	0.625926	0.915036	0.497716	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
4	0.311402	0.623372	0.924131	0.499217	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
5	0.311402	0.603363	0.960509	0.423397	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.342543	0.625926	0.951415	0.393762	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
7	0.342543	0.617612	0.915036	0.479189	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
8	0.373683	0.641430	0.896847	0.406238	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
9	0.404823	0.641475	0.869563	0.411019	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.373683	0.641475	0.851374	0.484214	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.373683	0.641475	0.915036	0.552994	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
12	0.467104	0.638967	0.796806	0.567906	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
13	0.529384	0.617413	0.678575	0.471384	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
14	0.529384	0.625453	0.633102	0.490463	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
15	0.560524	0.553443	0.614913	0.549242	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
16	0.529384	0.610931	0.633102	0.590493	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
17	0.529384	0.597625	0.605818	0.495740	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.529384	0.587482	0.651291	0.495927	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.538928	0.541570	0.600605	0.491332	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.467104	0.845238	0.760427	0.400992	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
21	0.467104	0.611403	0.778616	0.495606	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
22	0.435963	0.601306	0.814995	0.529548	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
23	0.404823	0.600995	0.878658	0.530148	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

24 rows x 30 columns

Local Data



Our Model Architecture



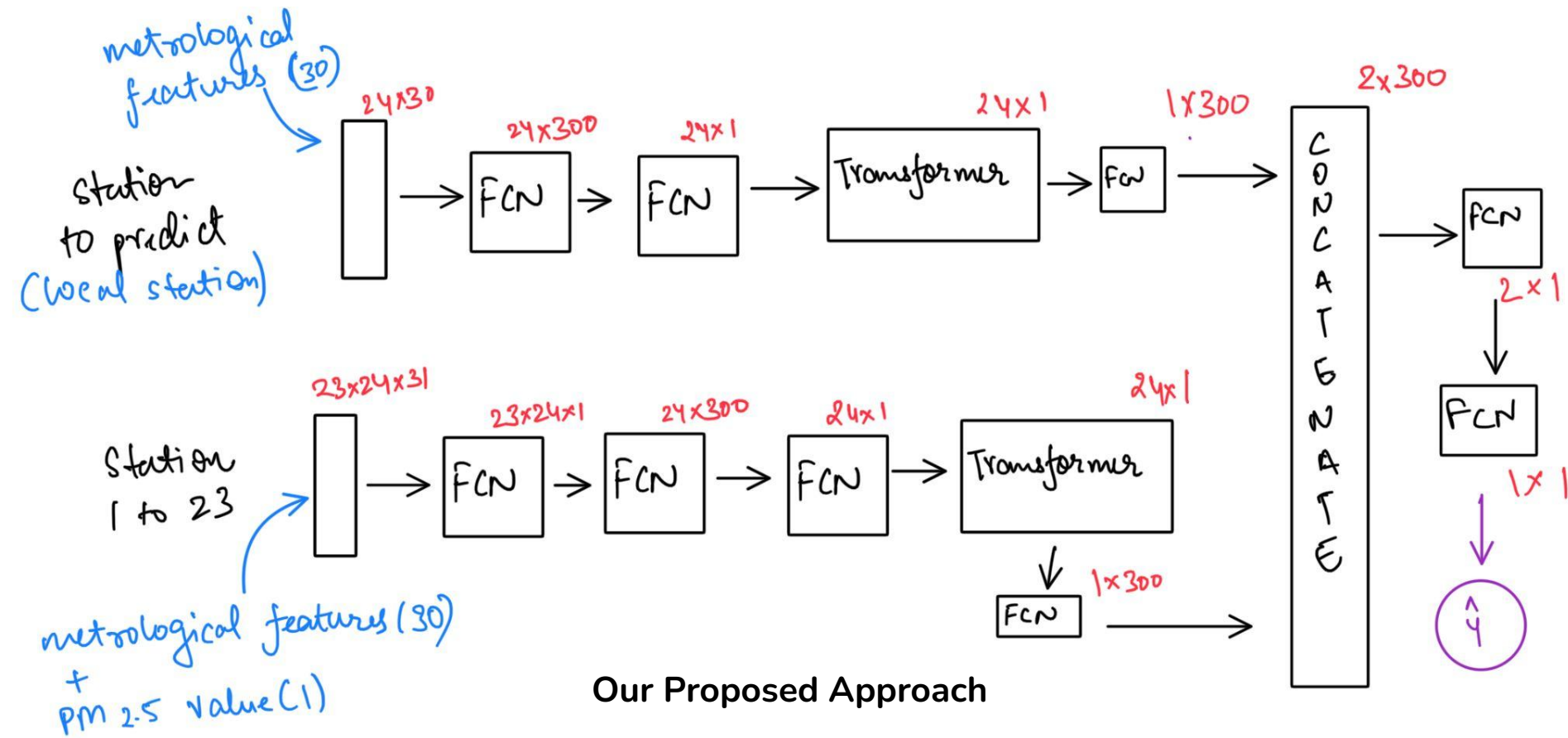
Transformer

- The existing methods have a weak ability to capture long-term dependencies and complex relationships from time-series data.
- Recent studies have proven that Transformers with attention mechanisms are far better learners of long-range dependencies than LSTMs in machine translation tasks in NLP, object detections, and classification.
- Thus we propose to use Transformers for capturing dependencies in time series data.



Model Architecture

- Predicts PM 2.5 value for a station based on data of the last 24 hours. Originally, 7 meteorological features. After one hot encoding 30 features. 23 remote stations per local station.
- For Local Stations input is (batch_size, time_window, met_features)
 - Apply series of FNNs on the meteorological features to learn dependencies → Pass into a transformer to learn temporal dependencies → An FNN layer to convert this into (batch_size, 300)
- For Remote Station input is (batch_size, remote_stations, time_window, met_features)
 - Apply series of FNNs on meteorological features to learn the dependencies → Apply FNN on remote stations to learn dependencies → Pass this into the transformer to learn temporal dependencies → A FNN layer will convert this into (batch_size, 300)
- Concatenate the outputs of the transformers → Apply a series of FNNs to get the final prediction for PM 2.5 value.



[Our GitHub Repository](#)

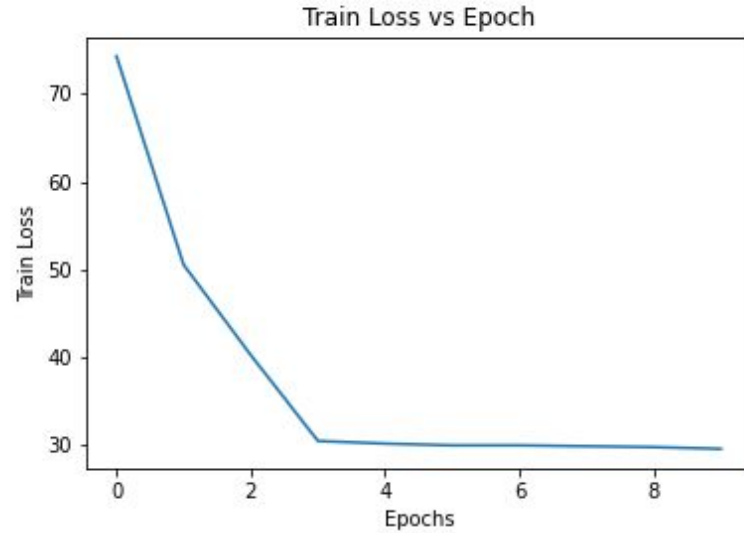


Results





Plot of train loss Vs Epochs



Final train_loss after 10 epochs is: 29.5




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	29.9

Final test_loss is: 29.9



Discussion





Inferences and Observations

- As seen from the observation table, the Transformer model implemented by us is almost same as the best baseline, that is, the Random Forest. This is because we have a very simple transformer architecture, and very low number of epochs (only 10). To further improve our model, we can increase the number of epochs, change the size of embeddings and add more layers.
- 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 $[-1, 1]$.



Future Work and Limitations





Future Work and Limitations

- Due to computational limitations, we could run our model only for a single 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 model we haven't incorporated the distance between stations. We can do that by adding a FNN to combine the transformer output with the distances to get a final output prediction which will have learnt the spatial dependencies in the data.
- We can add a GNN layer to capture the spatial dependencies in the data instead of just using the distance as spatial dependencies



References

- [1] MM ABDELWAHAB, Khaled SM Essa, HM Elsmam, 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.