Web Of Sensors: A Novel Vehicle Mounted Sensor Network Dataset and Analyses For Scalable Air Pollution Monitoring

Anonymous Author(s)

ABSTRACT

Air pollution is one of the biggest concerns faced by developing countries like India and the world at large. The capital of India, Delhi and the National Capital Region (NCR), sees life threatening air pollution levels. This paper presents a new Particulate Matter (PM) dataset for Delhi-NCR, which contains PM data recorded over three months from November 2020 to January 2021. The dataset contains more than 12.5 million sensor data points and spans an area of 559 square Kms. The data has been collected using vehicle-mounted mobile sensors in collaboration with the Delhi Integrated Multi-Modal Transit System (DIMTS) buses. The 13-bus dataset has been compared with the data over the same period obtained from the pre-existing static sensors, which the buses pass by. Several Machine Learning (ML) problems have been outlined, that can be studied using this dataset, two of which, spatio-temporal interpolation and recommending locations for incremental static sensor deployment are detailed in this paper. The dataset is public at *{hidden URL for author anonymity}* along with appropriate documentation.

ACM Reference Format:
Anonymous Author(s). 2018. Web Of Sensors: A Novel Vehicle Mounted Sensor Network Dataset and Analyses For Scalable Air Pollution Monitoring. In *The WebConf '22: The ACM Web Conference, April 25–29, 2022, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION AND RELATED WORK

Air pollution is a bane of modern civilization, specifically in the big cities of developing countries. *Particulate Matter (PM)* is especially dangerous, as our breathing cannot filter out the ultra fine particles. Air pollution has reached life-threatening levels in Delhi-NCR, one of the largest and most densely populated urban centers in the world. There have been studies analyzing factors affecting PM [1–5], but such PM measurement infrastructure are highly expensive (thousands of US Dollars per instrument). This cost-scalability trade-off is very evident from the fact that the *Central Pollution Control Board (CPCB)* and *Delhi Pollution Control Committee (DPCC)* have only 37 air pollution measurement centers in Delhi-NCR, which are thoroughly inadequate to cover the vast geography of 55, 000 square Kms. On the positive side, off-the-shelf *laser scattering* based PM sensors are now available at few tens of US Dollars (USD), along with promising work on low cost sensor calibration [6–11] against reference grade instruments like EBAM or TSI DustTrak.

The WebConf '22, April 25–29, 2022, The WebConf, Lyon, France

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

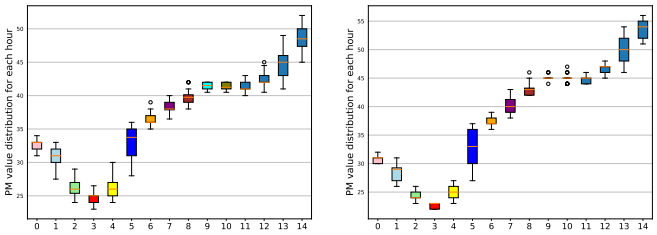


Figure 1: PM values measured by our low-cost PM sensor (USD 30) on the left vs. TSI DustTrak (USD 9500) on the right on a sample day Jul 24, 2021. The values are almost identical.

Fig 1 plots a low cost PM sensor (cost USD 30) on the left, and an industry grade reference instrument TSI DustTrak (cost USD 9500) on the right, showing hours of the day along x-axis and sensed PM 2.5 values along y-axis, for a sample day July 24, 2021. While the cost gap between the instruments is huge, the gap between their sensed PM 2.5 values, as seen in this graph, is negligible. This pattern has been observed consistently by us and other researchers [6–11]. Building on this positive observation of low cost PM sensing, this paper presents a novel PM dataset for Delhi, collected using IoT platforms comprising such laser based low cost PM sensors.

Our IoT platforms have been mounted on 13 public buses in Delhi for 3 months (Nov 1^{st}, 2020 to Jan 31^{st}, 2021), in collaboration with Delhi Integrated Multimodal Transport System, after rigorous tests for automotive safety certification and appropriate permissions and letters of support from the Delhi Ministry of Transport and Delhi Pollution Control Committee. The inside of our custom-made instrument comprising (a) PM sensor measuring PM 2.5, PM 10 and PM1, (b) GPS sensor to locate the bus, (c) 4G radio to communicate data from bus to server, (d) SD card for locally storing data when 4G signal is low, (e) BME sensor to record temperature and relative humidity and (f) micro-controller to orchestrate the sense-store-communicate software, is shown in Fig 2(a). The mounting location in the bus driver's cabin, next to two open windows to allow enough air-flow, is shown in Fig 2(b)-(c). Each bus commutes for 16-20 hours per day, and our instruments collect data at a fine granularity of 20 samples per minute. In Fig. 2(d), we show the government deployed static sensors installed in and around our bus trajectories, as location icons. We overlay a heatmap of PM data collection by the buses over the three months, in the same figure. Overall, the bus trajectories cover 559 sq Km, along the main arterial roads in North-West, North, North-East and South-East Delhi.

The dataset has been cleaned and made available at *{hidden URL for author anonymity}* with proper documentation. This dataset complements the static sensor data available from the 37 government deployed instruments in important ways. The static sensors are located at the top of high towers to get precise recordings of ambient pollution values, not affected by local sources. Our mobile sensors, on the other hand, are installed in the bus driver's cabin to measures

(a) Measurement device (b) Mounting location (c) Mounted device (d) Bus trajectories

Figure 2: (a) Shows the inside of our PM measuring IoT unit. (b) Shows the mounting location in bus driver's cabin in non air-conditioned public bus (below the existing white box). (c) Shows a mounted IoT unit in the bus (below the existing white box). (d) Heatmap of bus locations on 1^{st} January 2021, with static sensors shown as landmark icons.

the ground level pollution that daily commuters breathe in. Environmentalists can therefore use these complementary datasets to understand PM at tower heights using static sensor data, as well as near ground using our mobile sensor data. We compare the values across our mobile and the government installed static sensors in terms of quality and quantity in supplementary section.

Pollution researchers and environmental think tanks can use the released dataset to study pollution trends over a given time-period. For example, PM *cluster analysis* would reveal spatial and temporal pollution hot-spots. *Correlation analysis* of PM with other factors such as green-cover, population density, wind speed, humidity, etc. may be studied by environmentalists to mine factors that influence pollution. This auxiliary data may be collected from Google satellite images for green cover detection, Google Places for neighborhood characterization such as residential areas vs. commercial, and weather data repositories for humidity, wind speed, etc. The dataset would also facilitate designing of public-awareness programs such as mobile apps to disseminate pollution levels at a much finer granularity than currently possible, recommendation of pollution-free routes, identification of green-zones in a city, etc. Finally, the dataset would be invaluable for policy-decisions on deciding the optimal balance between low-cost IoT based mobile monitoring methods with high-cost sparse static sensing in developing countries. This is therefore a very important dataset from environmental sustainability stand-point, a pilot dataset which can make cost-effective scalable mobile PM monitoring a viable alternative in budget-constrained areas.

In addition to be of value to environmental researchers as discussed above, modeling the characteristic spatial and temporal patterns in this dataset is also of interest to Machine Learning (ML) researchers. We demonstrate a sample modeling exercise for spatio-temporal interpolation with several ML baselines on our dataset in Section 2. A second important research question we explore in this paper is recommending optimal locations for static senor placement under budget constraints (Section 3). Currently, government agencies randomly select locations for expensive static sensor installation. We demonstrate this random placement to have much higher pollution prediction errors than carefully selected locations using Graph Convolutional Neural Networks (GCN) and Reinforcement Learning, using our bus-collected dataset. When a new city or urban area needs to be instrumented with expensive static sensors, a pilot deployment of vehicle mounted sensors as ours, along with GCN-RL based location recommendation, can make the static sensor deployment more optimal and effective. Finally, the GPS part of the dataset can be interesting for other ML problems like automated detection of traffic congestion in non-laned traffic, as Delhi is also notorious for traffic congestion, and our buses cover the same routes daily in peak and non-peak hours.

To summarize, our key contributions are as follows:

- **Dataset:** We release the largest PM2.5 dataset from one of the most polluted regions in the world. The dataset is unique in terms of both scale and statistical characteristics.
- **Documentation of pilot study:** We not only release the dataset, but also document the mechanism used to collect this dataset, which will be useful to replicate a similar data collection exercise in other parts of the world.
- **Demonstrated ML applications:** We showcase two clear applications on the released dataset: **(1)** spatio-temporal interpolation of PM values to predict pollution levels in regions that do not receive direct sensor measurements and **(2)** using low-cost mobile sensors to locate budget-constrained strategic locations for installation of more expensive high-accuracy static sensors using a novel active learning based algorithm.

Released Dataset: Our collected and cleaned dataset can be found at *{hidden URL for author anonymity}*. The data is organized filewise for each day, each file containing values for up to 13 sensors deployed in buses. Not all 13 buses are active on all days of the 3 month deployment period (Nov 1^{st}, 2020 to Jan 31^{st}, 2021). DeviceID, GPS location, time-stamp, PM10, PM2.5 and PM1 values are listed in each row of a file. We compare the quantity and quality of our mobile monitoring data with static sensor data in the supplementary section. This exercise allows us to quantify our dataset's utility for environmentalists, who already have access to the static sensor dataset[1].

2 SPATIO-TEMPORAL INTERPOLATION

Spatio-temporal Interpolation is a widely explored ML research problem [12–21]. Since our mobile sensors record PM data only on a subspace of the entire spatio-temporal space, the problem of PM interpolation on the uncovered space is a natural consequence. Specifically, we study the following problem.

[1]https://openaq.org

Problem 1. *Given a training set of tuples of the form* $\langle date, time, latitude, longitude, pm2.5\rangle$, *learn a model to predict the PM2.5 value on an unseen spatio-temporal point* $p = (date, time, latitude, longitude)$.

Since we focus on the interpolation problem, we assume the unseen point $p \in \mathcal{B}$, where $\mathcal{B}$ is the bounding box over all spatio-temporal points seen in the train set.

2.1 Interpolation Methods

The baseline methods we use for interpolation on our dataset are described below, along with data pre-processing details for each modeling technique.

[1] GPR: The Gaussian Process (GP) regression model is defined by specifying the mean and the covariance function the GP prior [22]. While the mean function largely models the trend of the mean values in the model, the covariance function decides how changes in the features affect the values of the target variable. The covariance function is further specified using a kernel function [23] that learns the covariance between the different target values based on the feature values. The most common kernels used in a GPR model are squared exponential, Matern (MA), rational quadratic and periodic kernels [24]. The kernel function has free hyperparameters that are learnt while optimising the GPR model, for instance, variance and lengthscale of the kernel along with the variance of the Gaussian noise in the model.

Firstly, a downsized train dataset is created by taking the mean of the PM2.5 variable for all observations which are at a spatial distance of approximately 150m to each other and at a temporal distance of 15 minutes, making this averaged point one single point. This downsized dataset is further normalized, in order to make the mean zero and standard deviation 1 for the input and output features. While the mean function was specified as a constant function, the kernel function was the product of a Periodic and MA-1/2 kernel. This was done in order to give more importance to the points which were both spatially and temporally close while also allowing us to model the datetime feature using periodicity. We used ARD, automatic relevance determination (ARD) [22] to allow different values of hyperparamters to be fitted for each feature. For both our Gaussian process-based models, we have used the python library GPyTorch [25] to code them.

[2] Variational GPR: The Variational GPR allows us to use inducing points to learn a more easily tractable final posterior distribution, by computing the variational lower bound (or ELBO) rather than the GP marginal log likelihood, as compared to the above mentioned GPR model. This allows us to extend the GPR framework to big data. The preprocessing for this model is similar to the preprocessing for GPR. The mean function is kept as a constant function again though the kernel function here is just a MA-1/2 function with ARD. Further, the number of inducing points here is limited to 2000 points which helps us strike a good balance between computational requirements and accuracy of the model.

[3] ANN: Extensive testing was performed with various multilayer perceptron models with varied hyperparameters to find the best architecture. We had three input features latitude, longitude and datetime and PM 2.5 as target variable. All the hyperparamters were tuned based on the train and validation datasets. Input aggregation of data 15 mins apart, normalization for zero mean and standard deviation 1 were done. For the model, a 6-layered architecture was found to be the best (one input layer, one output layer and 4 hidden layers). All the hidden layers had 200 densely connected hidden units each and weight initialization of [26] was used. Further, Rectified linear unit (ReLU) was used as the activation function for all layers. For optimization, we used Adam [27] optimizer with an exponentially decaying learning rate. Finally, the loss function used was Mean Squared Error (MSE) loss between predicted and actual PM2.5 values. Model implementation was done using open source PyTorch library [28].

[4] GraphSAGE: We aim at learning universal weights, similar to GraphSAGE [29], which will signify the importance of a neighbour based on some known node values and edge weights. Here we define node values as the value of the pollutant PM2.5 while the edges are created using latitude, longitude and datetime features. Firstly, a graph is created from the train dataset, aggregating all inputs within 150m and 15 minutes of each other into a single node. An edge is created between two nodes if they lie within 2Km and 2 hours of each other. Weights of the edges are inversely dependent on the product of a term consisting of the squaring of the haversine distance between those nodes and the temporal distance between them (which was taken as 15 minutes here). The graph then goes through two graph-based layers to learn the required weights where embeddings are learnt using the max and mean aggregation layers, followed by 3 fully connected neural network layers to predict the final pollutant value. Finally, instead of just using one graph to train our model, we construct 50 such smaller subgraphs by dividing the main graph randomly into smaller subsets. This makes our training and validation process more robust as we perform a random walk which protects us against bias in our model. The loss function is the common Mean Squared Error loss between the predicted value of pollutant of all the nodes. For both this model and the Meaner, we have used PyTorch Geometric[30] to code them.

[5] Meaner: Meaner simply uses the graph described in our GraphSAGE model and averages the values of the known neighbours of any given unknown node to predict its value. Thus, a weighted mean based on the value of the edges and the nodes neighbouring the unknown node is performed where the weights are the values of the edges.

2.2 Experimental Setup

To evaluate each method, we take five common randomly chosen days from our data and separate out the data hour-wise from 9:00AM to 9:00PM, split the data randomly into train and test using a ratio of 80:20 and further split the train data into train and validation sets using the same ratio. Train root mean square error (TRAIN_RMSE) and the test root mean square error (TEST_RMSE) [31] are computed using ground truth PM values. We also measure the time taken to get the final model (TRAIN_TIME) (in seconds) and provide information on if the method outputs variance values (VAR). In addition to training till convergence (denoted by $_FULL$), we also report values for 100 epochs training (denoted by writing $_100$), to check whether models can achieve a decent error rate with streaming data. We use a server machine with 96 GB RAM. The CPU is 2x Intel Xeon G-6148 (20 cores 2.4 GHz) "Skylake" and the GPU is 2x NVIDIA V100 (32GB GPU memory with 5120 CUDA cores).

Table 1: Spatio-temporal Interpolation Baselines

Models	Metrics						
	TRAIN_RMSE_{100}	TEST_RMSE_{100}	TRAIN_TIME_{100}	TRAIN_RMSE_{FULL}	TEST_RMSE_{FULL}	TRAIN_TIME_{FULL}	VAR
GPR	29.76	30.41	1176.45	29.13	29.74	3433.04	✓
Variational GPR	32.42	33.05	267.89	29.89	30.63	1053.88	✓
ANN	47.04	47.51	51.78	31.61	32.49	141.42	×
GraphSAGE	×	35.45	4654.64	×	34.15	4894.81	×
Meaner	×	×	×	×	33.84	1578.64	×

2.3 Evaluation Results

The results are documented in Table 1. We observe from TEST_RMSE_{100} and TRAIN_TIME_{100} that most of these models can be used easily in a streaming fashion, especially Variational GPR which has a good trade-off balance between TEST_RMSE_{100} and TRAIN_TIME_{100}. ANN on the other hand take the least time and thus can be quickly used to produce interpolation maps for deployment purposes. GPR is a fast method too but has a limitation on the size of the dataset it can process. Thus, when using this on larger datasets, it can run into memory-based problems.

GPR, Variational GPR and ANN take less time to train for 100 epochs and get a decent RMSE, as compared to their convergence RMSEs. GraphSage and Meaner, however, take significant time to achieve low RMSEs. This is caused by the time taken for graph building, as creating edges for each node consumes a lot of time. Since the model is fixed for Meaner and getting the predictions on a given graph doesn't take much time, its TRAIN_TIME_{FULL} indicates mainly the time taken to preprocess and build the graph on our dataset.

In terms of TRAIN_RMSE_{FULL} and TEST_RMSE_{FULL}, GPR methods is the best. GPR methods also output a variance value with each prediction, unlike other models. This variance value shows how confident the model is about its prediction and is thus extremely useful for a number of purposes including model-based predictive control [32], comparing space-filling designs [33] and active learning [34].

Both static and mobile sensor datasets will have spatial or temporal gaps in measurement, making interpolation a necessity. In this section, we have examined several baseline algorithms for interpolation, which give comparable RMSE, but vary in terms of processing time. Based on the real time processing requirements with streaming PM data vs. historical analyses of air pollution information with no real time requirements, an appropriate choice of algorithm needs to be made by urban environmental agencies.

3 STATIC SENSOR LOCATION RECOMMENDATION

The next problem we explore with our collected PM dataset is how to recommend locations for deploying expensive static sensors. Static sensors are industry grade, with excellent calibration for pollutant values, standardized APIs to compute Air Quality Index (AQI) aggregating different pollutants, among other advantages. The only drawback is their cost, which makes them difficult to deploy at high spatial density, especially in developing countries, where budget is a constraint. Therefore, given a restricted budget for static sensors, low cost vehicle mounted sensor data might be useful to recommend the optimal locations to deploy the static sensors, so that some metric (e.g. spatio-temporal interpolation errors) are minimized. We examine this research question of optimal static sensor location recommendation in this section, using graph convolutional neural network (GCN) based policy learning [35].

Problem 2. *Suppose we have a pollution prediction model $\mathcal{M}$ that takes recordings of all existing static sensor readings $\mathcal{S}$ and an input query spatio-temporal location $q = \langle time, latitude, longitude \rangle$, and outputs the predicted PM2.5 recording at q. Let $\mathcal{E}(\mathcal{M}(\mathcal{S}, q))$ denote the expected error of $\mathcal{M}$ with $\mathcal{S}$ for any given location q. Given a set of candidate locations C where static sensors may be installed, and a budget b, we would like to add b static sensors such that the marginal decrease in expected error is maximized. Mathematically,*

$$\underset{\mathcal{A} \subseteq C, |\mathcal{A}|=b}{\text{argmax}} \; \{\mathcal{E}(\mathcal{M}(\mathcal{S}, q)) - \mathcal{E}(\mathcal{M}(\mathcal{S} \cup \mathcal{A}, q))\} \tag{1}$$

The error on a spatio-temporal point q is defined in terms of RMSE. Mathematically,

$$\mathcal{E}(\mathcal{M}, q) = \sqrt{\mathcal{M}(\mathcal{S}, q)^2 - PM2.5(q)^2} \tag{2}$$

Here, $PM2.5(q)$ denotes the true pollution recording at q. To obtain the ground-truth, we utilize our dataset of PM2.5 values collected through mobile sensors. Specifically, let $Q = \{q_1, \ldots, q_n\}$ be the set of all PM2.5 recordings from mobile sensors. The error of a model $\mathcal{M}$ is therefore

$$\mathcal{E}(\mathcal{M}) = \frac{1}{|Q|} \sum_{q \in Q} \mathcal{E}(\mathcal{M}, q) \tag{3}$$

3.1 GCN based Location Recommendation

We use two connected graph convolution based modules for recommending static sensor node locations. The *Predictor module* $\mathcal{M}$ predicts $PM2.5$ values at certain locations and times as a regression

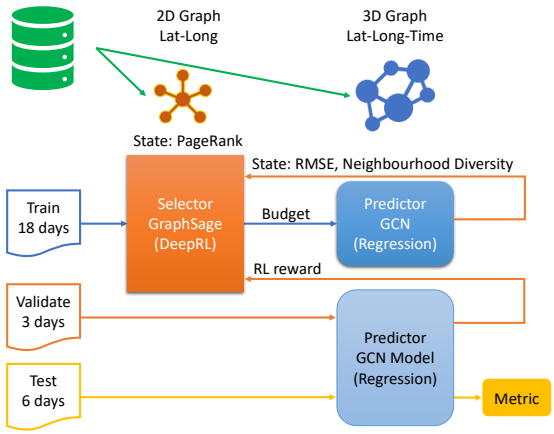


Figure 3: GCN based Recommender Architecture

task. The *Selector module* π incrementally recommends new static sensor locations until the sensor budget is hit. The Selector module tries to minimize the RMSE of PM predictions by the Predictor module, based on the validation dataset, using the GCN architecture, states and rewards shown in Fig 3.

3.2 3D and 2D Graphs and Dataset Details

Similar to the approach in Section 2, the collected mobile datasets are transformed to a 3D graph (for Lat-Long-Time). We average the spatio-temporal locations within 150m and 15 minutes, and consider the aggregated data as one node. For a given day, the nodes within 1 km spatial distance and 1 hour temporal distance are connected by an edge. For adjacent days, the nodes within 1 km spatial distance and same time (i.e. no temporal distance) are connected by an edge. The weight of each edge is $\frac{1}{1+d}$, where d is the spatial distance between the connected nodes ($0 \leq d \leq 1km$). This 3D graph is used for the PM values predictor module described above, as Pollution is both space and time dependent.

For generating the 2D graph, we remove the temporal dimension from the 3D graph, leaving the nodes based on spatial locations (Latitude and Longitude). Two nodes in the 2D graph are connected if there is at least one edge between any pair of nodes at this spatial locations in the 3D graph. As the edge weight is dependent only on the spatial distance, the edge weight in 2D graph corresponds to that of the 3D graph. This 2D graph is suitable for the selector module described earlier, as we intend to select/recommend spatial locations for the static sensors, which would function at that location throughout the day. There is also a (one to many) mapping between the nodes of the 2D and 3D graphs to facilitate the selection and prediction modules work together.

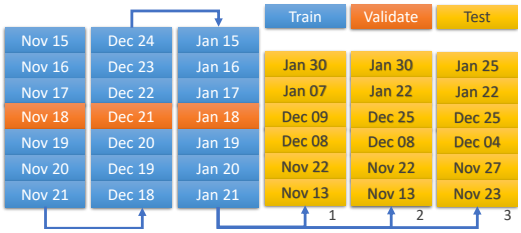


Figure 4: Train, validation and test datasets

As shown in Fig 4, we take 7 continuous days from each of the months (Nov, Dec, Jan) as training and validation data. The middle day from the 7 days is used for validation and others for training. Any day next in sequence is the adjacent day, and hence edges exist across nodes from such days. We take 3 sets of test data, with 2 days from each month. Separately appending these test datasets to the above described training/validation datasets, gives three graph datasets for our evaluations. Dataset 1 and 3 have mutually exclusive test days. Dataset 2 is a mix of the two.

3.3 GCN based Predictor Module ($\mathcal{M}$)

This comprises of two Graph Convolution layers followed by an MLP layer. It predicts the PM value of nodes in a 3D graph, based on the one-hot encoded Lat-Long-Time features corresponding to the nodes. With the Adam optimizer, we utilize Mean Square Error as the loss function over the training nodes corresponding to the selector module's recommendation. This Predictor module is used for all our models and baselines alike, while the Selector module varies across our models and baselines. Per iteration of the Selector, Predictor trains for budget+75 epochs in order to achieve a suitable convergence for PM value prediction.

3.4 GraphSAGE based Selector Module (π)

This is a GraphSAGE [29] based Graph Neural Network, comprising of one or two (Mean/Max) Aggregation layers followed by two MLP layers, giving non-normalized probabilities. These probabilities are then normalized and used to select the optimal nodes upto given budget from the 2D graph (only lat-lon locations for static sensor deployment, and not time, as the static sensors will be active at all times). The optimal nodes in the 2D graph have corresponding nodes in the 3D graph, and are chosen by the selector module to give best marginal improvement in predicting the PM values of the validation nodes in the 3D graph. This selection is performed sequentially upto the given budget on per batch basis, for a multi-batch training, from the remaining pool of training nodes available for selection. The RMSE over the validation set is fed as a Reinforcement Learning reward to the selector, using Adam as the optimizer. For the state features, we calculate PageRank from the 2D graph, and extract Node absolute RMSE and Neighbourhood RMSE difference from the output of the Predictor.

In the training phase, given graph G, our goal is to maximize the sum of expected rewards obtained from following policy π over G. The objective function with respect to the selector network parameters θ is

$$\mathcal{J}(\theta) = \mathrm{E}_{P(V^b_{budget};\theta)}\mathcal{R}(V^b_{budget}) \tag{4}$$

where b is the query budget and $\mathcal{R}$ is the trajectory reward. We utilize REINFORCE [36], a classical policy gradient method, to train the selector network.

After training, we take the trained Selector model π^* with lowest validation RMSE. This model recommends the best 2D budget nodes. Using the corresponding 3D nodes, the Predictor $\mathcal{M}$ aims to minimize the RMSE metric over the 3D test nodes. We report these RMSE over the PM values of the nodes from the test sets as metric in our evaluations.

3.5 Selector's Active Learning as MDP

We consider a graph denoted as $G = (V; E)$, where V is a set of nodes and E is a set of edges. Each node $v \in V$ is associated with a feature vector $x_v \in \mathcal{X} \subseteq \mathbb{R}^d$, and it ground-truth label $PM2.5(v) \in \mathcal{Y} \subseteq \mathbb{R}$. The node set is divided into three subsets as V_{train}, V_{valid}, and V_{test}. In conventional semi-supervised or budget-limited node regression, the labels of a subset $V_{budget} \subseteq V_{train}$ are given. The task is to learn a regression network $\mathcal{M}$ (formulated as a GNN) with the graph G and V_{budget} to predict the nodes in V_{test}.

For active learning on graphs, the selected training subset is initialized as an empty set, $V^0_{budget} = \phi$. A query budget b is given, which allows us to sequentially acquire the labels of b nodes from V_{train}, where $b << V_{train}$. At each step t, we select an available node v_t from $V_{train} \backslash V^{t-1}_{budget}$ based on an active learning policy and query the label of v_t. Next, we update the selected node set as $V^t_{budget} = V^{t-1}_{budget} \cup \{v_t\}$. The prediction GNN $\mathcal{M}$ is then trained

with the updated V^t_{budget} for one more epoch. When the budget b is used up, we stop the query process and continue training the prediction GNN $\mathcal{M}$ with V^b_{budget} until convergence.

As different query sequences will be tried during policy training, we require full label information on the training and validation set.

3.6 MDP State

For a Graph G, we denote the state at step t as a matrix S^G_t, where each row s^v_t is the state representation of node v. Based on several commonly used heuristic criteria, we define the state representation of each node in active learning as follows.

i. PageRank: PageRank [37] is a traditional algorithm used by Google for search queries. It represents the centrality and importance of a node v in a connected data structure G. The PageRank of a node v, in relation to all connected nodes, is defined as

$$W_v = (1-d) + d \sum_{i=1, i\neq v}^{N} I_{iv} \frac{W_i}{n_i} \implies s^t_v(1) = W_v / \max_{i=1}^{N} W_i \quad (5)$$

where d is the damping factor, which is 0.85, and n_i is the count of connections at node i. We normalize the PageRank of all nodes to form that as a part of Selector state, which remains constant throughout the training process.

ii. RMSE from the Predictor: We calculate the RMSE loss $L(x;t)$ for all 3D nodes using the Predictor's output for the 3D graph. The RMSE loss is then averaged over the set of nodes corresponding to the 2D node v.

$$s^t_v(2) = \frac{1}{|n_v|} \sum_{x \in n_v} L(x;t) \quad (6)$$

iii. Neighbourhood RMSE Diversity: Using the $s^t_v(2)$ for node v, we calculate mean RMSE difference between node v and all its neighbours u.

$$s^t_v(3) = |L(v;t) - \frac{1}{|N_v|} \sum_{u \in N_v} L(u;t)| \quad (7)$$

An indicator to specify whether a node v is picked under budget for Predictor's RMSE loss calculation, or not.

$$s^t_v(4) = \mathbf{1}\{v \in V^{t-1}_{budget}\} \quad (8)$$

We concatenate all above features for the state representation of each node s^t_v to create a graph state matrix S^t_G, which is passed to the selector to learn more complex representations.

3.7 MDP Action and Reward

At step t, the Selector has to select a node v_t from $V_{train} \backslash V^{t-1}_{budget}$ based on the action probability given by the Selector GraphSAGE network in the same step t.

We use the RMSE loss calculated over the validation set after Predictor's convergence as the trajectory reward. Given a sequence of budget selected nodes $V_{budget} = (v_1, \ldots, v_b)$, we define the reward as

$$\mathcal{R}(V_{budget}) = \mathcal{E}(\mathcal{M}(\mathcal{S}, V_{valid}), PM2.5(V_{valid})) \quad (9)$$

where $\mathcal{M}$ is the Predictor GNN model trained using the graph G and $PM2.5$ values of V_{budget}. V_{valid} and $PM2.5(V_{valid})$ are the nodes and labels of the validation set. $\mathcal{E}$ is the Evaluation metric (RMSE loss).

3.8 MDP Framework

At each query step t, a new node v_t is added to V^t_{budget} to update the Predictor $\mathcal{M}$, the graph state thus transits from S^t_G to S^{t+1}_G. The selection indicator of the selected node v_t is changed from 0 to 1. Updating the Predictor $\mathcal{M}$ can influence the predicted $PM2.5$ values, thus the *RMSE* and *Neighbourhood diversity* in the state vector of each node will change accordingly. Since it is hard to directly model the transition dynamics $p(S^{t+1}_G | S^t_G; v_t)$, we learn the optimal policy in a model-free approach.

At query step t, we first update the current graph state S^t_G with the graph G and the outputs of $\mathcal{M}_{t-1}$. The Selector network π takes S^t_G as input and produces a probability distribution over actions as p^t_G, which represents the probability of selecting each node in the candidate pool $V_{train} \backslash V^{t-1}_{budget}$. Next, we sample a node v_t based on p^t_G and add it to the budgeted subset to get V^t_{budget}. $\mathcal{M}_{t-1}$ is trained for one more epoch with V^t_{budget} to get $\mathcal{M}_t$ (or simply $\mathcal{M}$), which is then used to generate the graph state S^{t+1}_G for the next step. When $t = b$, we stop the query phase and train $\mathcal{M}$ until convergence (75 more epochs for our implementation). Finally, we evaluate $\mathcal{M}$ on the validation set V_{valid}, and the performance score is used as the trajectory reward $\mathcal{R}$ to update the selector network π.

3.9 Existing Static Sensors Locations

Static sensor location recommendation and deployment will typically happen in an incremental fashion in any city. The next best location recommendation will be queried from a system like ours, given the current static sensor deployment. Therefore for analyzing the performance of our location recommender system, we extract the locations of the existing static sensors in the region of our mobile pollution data collection. We identify 10 static sensors lying on the routes of our mobile sensors. They are shown in Green colour in Fig 6, and are used in our performance analyses. The next set of 10 nearby sensors, shown in Blue color, are also utilized for a baseline method.

3.10 Baselines and Selector Transformations

To analyze the suitability of our methods, we need to compare them with suitable baselines and also among suitable transformations in our own model. With the predictor module common throughout the analysis, below are the baselines we selected for the performance comparison (replacing our DeepRL based selector module).

i **Random:** We pick the budget nodes uniformly and randomly from the selection pool of available training nodes in a sequence for each of the training batches.

ii **Centrality:** We pick the budget nodes with the highest PageRank value from the selection pool of available training nodes.

iii **Static:** We utilize the locations of the installed static sensors, as discussed in previous section 3.9. For a budget of 20, we use the green and blue locations as shown in Fig 6. During training, for each batch pool, we sequentially pick the nodes nearest to the location of the static sensors.

There is also an state-of-the-art method **Modulo** [38], which greedily suggests some routes giving the best spatio-temporal (3D) coverage, by favouring the routes with sample points in separate 3D cells. Though Modulo uses the actual data samples and does not

accumulate them and does not have any graph formation, the random baseline listed above is doing the same thing as modulo would do when translated to our problem. Hence Modulo is inherent in the Random baseline. As the baselines select either fixed or random nodes, we perform the training process for 10 iterations and take an average over the same, for 5 batches, to factor out the randomness in Predictor GCN's initial weights.

For our Selector GraphSAGE module, we employ various combinations of the Mean and Max Aggregation, and have the following models: **i. Max:** A Max Layer, **ii. Mean:** A Mean Layer, **iii. MaxMax:** A Max Layer followed by another Max Layer, **iv. MeanMax:** A Mean Layer followed by Max Layer, **v. MaxMean:** A Max Layer followed by Mean Layer. **vi. MeanMean:** A Mean Layer followed by another Mean Layer. All these models have two MLP Layers in the end.

3.11 Performance Analysis

We start with analysing the performance of our location recommender system, over a reasonable budget of 20 recommended locations. The RMSE metrics for the dataset 3 are shown in Fig 5, where lower indicates better. Above each bar, the number represents the Std deviation across the 5 batches, lower being better. We observe that our Selector module recommends better locations as compared to the random and static baselines, with significantly lower RMSE over the unseen test set. Among the Selector transformations, we find *Max*, *Mean*, *MeanMean* perform a bit lower than the others. We observe *MaxMax* performing better, possibly due to the consistency of Max aggregation in selecting the dominant features over two layers.

In Table 2, we present the RMSE separately for different days in the three test datasets. We also split the PM values into 10 classes and present the F1Score. We again observe our system performing better than baselines across individual days, both for RMSE and F1Score. Out of the pair of the numbers presented in the table, the first is the mean RMSE across the 5 batches, and the second (in brackets) is the Std Deviation across the same. We find that our Selector not only improves mean RMSE, but also performs consistently across the 5 batches (with low Std Deviation).

We also look into the effect of increasing sensor budget on RMSE in Table 3, comparing the performance of MaxMax Selector against Random and Centrality baselines. We observe that the benefits of using our method over random decrease, as we keep on increasing the sensor budget. This is intuitive, as with very high budgets, even randomly selecting sensor locations would do fine. Careful choice, using a recommender system as ours, makes sense when budget is restricted, as will be the case in most practical deployments especially in developing countries.

Table 3: RMSE for increasing budgets.

Dataset	Method	Budgets				
		1	5	10	20	50
1	Random	141.4 (22.9)	105.3 (16.5)	100.3 (16.5)	95.0 (13.1)	84.8 (7.1)
	Centrality	108.2 (1.2)	106.6 (1.1)	100.2 (2.0)	119.0 (3.5)	147.6 (5.2)
	MaxMax	99.6 (0.5)	83.5 (0.4)	94.8 (1.8)	71.0 (0.7)	73.7 (1.3)
2	Random	151.5 (26.3)	103.4 (14.5)	96.9 (16.2)	86.4 (12.0)	77.0 (6.9)
	Centrality	115.3 (0.8)	114.0 (1.0)	99.4 (1.8)	117.8 (3.5)	151.0 (5.5)
	MaxMax	105.3 (0.7)	82.5 (0.5)	67.4 (0.4)	64.0 (0.6)	63.5 (0.6)
3	Random	155.7 (25.0)	120.2 (18.3)	109.4 (15.4)	100.3 (12.3)	93.4 (8.0)
	Centrality	126.3 (0.8)	125.9 (0.8)	111.4 (1.4)	129.6 (2.7)	137.1 (4.2)
	MaxMax	184.2 (0.3)	99.2 (1.3)	82.4 (1.7)	79.0 (0.4)	80.4 (0.8)

We further analyze the performance of our Selector in recommending static sensor locations for a given budget, starting with some pre-existing static sensors. For a budget of 20, we take the 10 Static Sensor locations marked in Green in the Fig 6. On every iteration, we first pre-train the Predictor with the mobile sensors adjacent/near to the 10 Static Sensors, then let the Selector choose the remaining 10 locations from the pool of the remaining mobile sensor locations. It was encouraging to find that our Selector still improves the performance by carefully selecting the remaining sensors from the mobile pool. This analysis is presented in Table 4. When no static sensors are used, our method is better than Random and Centrality

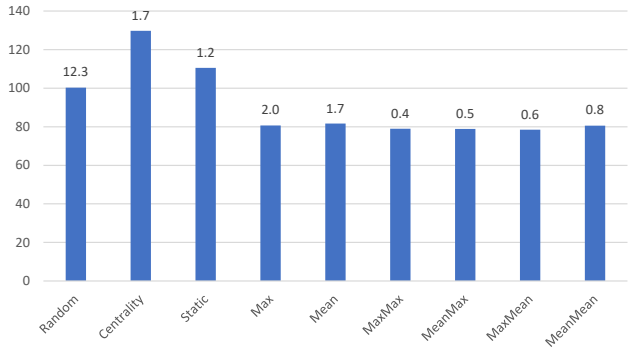


Figure 5: RMSE (as bar) and Std Deviation (above bar)

Table 2: TestDay-wise RMSE for 20 budgets (Std Deviation across the 5 batches in brackets), with F1Score

Dataset	Method	RMSE							F1Score
		TestDay1	TestDay2	TestDay3	TestDay4	TestDay5	TestDay6	Overall	
1	Random	89.7 (9.7)	88.0 (15.7)	80.1 (10.9)	89.4 (11.9)	129.1 (15.0)	94.0 (15.0)	95.0 (13.1)	41.2
	Centrality	113.1 (2.9)	125.8 (5.0)	93.5 (2.3)	106.9 (2.6)	153.7 (4.5)	116.3 (3.6)	118.2 (3.5)	36.8
	Static	102.7 (1.7)	97.7 (2.0)	90.2 (1.6)	96.2 (1.4)	148.4 (2.1)	112.1 (1.8)	107.9 (1.8)	41.9
	MaxMax	71.8 (0.3)	56.4 (1.1)	57.1 (0.4)	69.6 (0.5)	105.3 (1.0)	66.1 (1.0)	71.0 (0.7)	49.2
2	Random	89.0 (10.4)	85.8 (15.2)	79.6 (10.3)	80.4 (12.1)	92.5 (9.6)	91.1 (14.4)	86.4 (12.0)	44.9
	Centrality	117.6 (2.7)	132.8 (4.2)	97.4 (2.2)	111.9 (3.2)	130.4 (2.1)	122.1 (3.2)	118.7 (2.9)	36.1
	Static	102.3 (1.4)	96.5 (1.7)	89.1 (1.2)	100.3 (1.5)	106.1 (1.0)	111.4 (1.5)	100.9 (1.4)	45.0
	MaxMax	72.3 (0.2)	57.0 (1.1)	58.7 (0.3)	53.9 (0.6)	76.3 (0.5)	65.9 (0.9)	64.0 (0.6)	53.9
3	Random	89.2 (12.7)	125.4 (18.2)	116.2 (11.7)	80.5 (10.6)	91.5 (9.5)	99.2 (11.1)	100.3 (12.3)	39.2
	Centrality	122.5 (2.1)	140.9 (2.3)	155.9 (1.4)	109.5 (1.8)	135.7 (1.4)	113.7 (1.6)	129.7 (1.7)	31.9
	Static	89.6 (1.0)	136.1 (1.4)	111.0 (0.8)	100.6 (1.6)	105.5 (1.1)	120.4 (1.6)	110.5 (1.2)	38.0
	MaxMax	71.2 (0.2)	88.6 (0.5)	109.6 (0.5)	55.4 (0.2)	78.5 (0.4)	71.0 (0.6)	79.0 (0.4)	45.1

Table 4: RMSE for 20 budgets for varying Static Sensors (Std Deviation across the 5 batches in brackets)

Dataset	Method	No Static	10 Static	20 Static
1	Random	95.0 (13.1)	87.0 (8.7)	
	Centrality	118.2 (3.5)	89.2 (0.7)	107.9 (1.8)
	MaxMax	71.0 (0.7)	74.2 (0.5)	
2	Random	86.4 (12.0)	77.8 (6.6)	
	Centrality	118.7 (2.9)	77.9 (0.9)	100.9 (1.4)
	MaxMax	64.0 (0.6)	64.2 (0.6)	
3	Random	100.3 (12.3)	90.8 (7.3)	
	Centrality	129.7 (1.7)	90.1 (1.0)	110.5 (1.2)
	MaxMax	79.0 (0.4)	80.9 (0.5)	

baselines (as already seen in earlier sections). When 10 pre-existing static sensors are present and the next 10 are recommended, our system again does better than the baselines. Most interesting is the last column showing the RMSE with the existing 20 static sensors (blue and green locations in Fig 6). The RMSE values are higher than our MaxMax Selector's RMSE in the earlier columns. This shows that a recommender system like ours to plan sensor deployment can greatly improve pollution prediction, compared to the current static sensor infrastructure in Delhi city.

3.12 Are Our Recommendations Stable?

To analyze the consistency of our recommendations, we use bipartite matching analysis, across different sets of recommendations. We form a complete bipartite graph using two sets of recommended nodes. The edge weights are the haversine distance between nodes. Performing a minimum weight bipartite matching allows us to map nodes in the vicinity. The mean of the matched edges represent the consistency of the matching, and hence the stability of the recommendations. The lower the mean value, the better is the consistency. For matching across different batches from the same dataset, we take the Mean (and Std Deviation) across B x $(B-1)$ bipartite matching, where B is the batch-size. For matching across different datasets, we take the Mean (and Std Deviation) across B x B bipartite matching. These values are presented in Table 5.

Table 5: Mean Distance and Std Deviation (in km) for Bipartite matching over recommended nodes for MaxMax

Dataset	1	2	3
1	0.474 (1.990)	0.993 (2.791)	1.885 (3.113)
2	0.993 (2.791)	0.104 (0.465)	1.560 (2.930)
3	1.885 (3.113)	1.560 (2.930)	0.008 (0.034)

3.13 Visualizing Recommended Locations

We observe the recommendations from MaxMax Selector module for prospective installation locations of the static sensors, and checked them alongside the existing static sensor locations. Figure 6 shows the 20 existing static sensors in Green and Blue colour and 20 fresh recommendations from the Selector in Red colour. An incremental recommendation of 10 sensors over the existing 10 Green. The routes of the mobile sensors are shown in pink colour. We also observe that Selector recommendations are mostly consistent across the batches, which is also evident from our bipartite matching based analysis. There are some co-locations between the full and incremental recommendations as well.

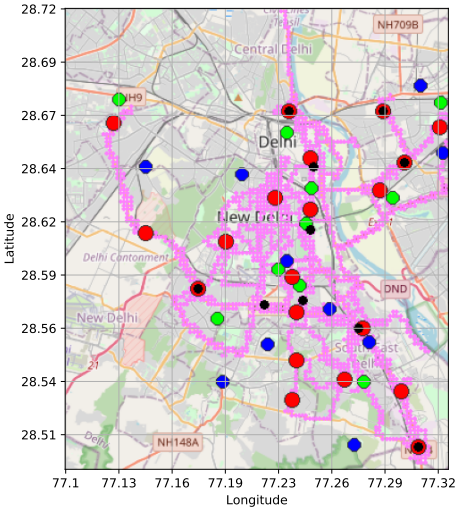


Figure 6: Green+Blue: Existing 20 static sensors, Red: Recommended 20 sensors, Black: Recommended 10 sensors over existing 10 static Green.

4 CONCLUSION AND FUTURE WORK

We release a PM dataset for Delhi in this paper, collected between Nov 1^{st}, 2020 and Jan 31^{st}, 2021 using IoT devices deployed in the driver's cabin in 13 public buses. The data has been cleaned and correlated with the government deployed static sensors, whenever our bus mounted unit has come in the vicinity of a static sensor. Environmental researchers can use this complementary dataset, to understand PM exposure at ground levels, as well as PM temporal variations over days, weeks and spatial variations across locations. We also outline several ML research problems that can use this dataset, and discuss in depth two of these. For the *spatio-temporal interpolation problem*, we present several baseline models' performance on the dataset. For recommending static sensor deployment locations, we use GCN based policy learning [35], and demonstrate our recommended locations give better pollution predictions than the current static sensors deployed in Delhi-NCR, which are similar to random sensor placements.

We are currently extending the sensor deployment to more buses and and other vehicle fleets (shared cabs, delivery agencies' two-wheelers etc.), and augmenting our IoT unit with gas sensors. We will keep augmenting the dataset (with comparisons with static sensors, as done here) when more data gets collected. We are also collecting and analyzing auxiliary datasets that affect PM: road traffic congestion data (from this dataset's bus GPS traces, Google Traffic APIs), green cover vs. built area information (from satellite image datasets), residential areas vs. commercial areas (from Google Places and OpenStreetMap POI datasets), so that our measured PM values can be correlated at fine-grained spatio-temporal scales with the auxiliary data for factor analysis. Finally our MSE loss metric is outlier sensitive. As our dataset contains outliers, a more fair baseline to compare would be to use likelihood estimation methods and Bayesian neural networks [39–41], that tolerate outliers better. We will explore these interpolation methods in future.

REFERENCES

[1] Apte JS, Kirchstetter TW, Reich AH, Deshpande SJ, Kaushik G, Chel A, Marshall JD, and Nazaroff WW. Concentrations of fine, ultrafine, and black carbon particles in auto-rickshaws in new delhi, india. *Atmospheric Environment*, 45, 2011.

[2] Google. Mapping the invisible: Street view cars add air pollution sensors, 2014.

[3] Messier KP, Chambliss SE, Gani S, Alvarez RA, Brauer M, Choi JJ, Hamburg SP, Kerckhoffs J, LaFranchi B, Lunden MM, Marshall JD, Portier CJ, Roy A, Szpiro AA, Vermeulen RCH, and Apte JS. Mapping air pollution with google street view cars: Efficient approaches with mobile monitoring and land use regression. *Environmental Science & Technology*, 52, 2018.

[4] Apte JS, Messier KP, Gani S, Brauer M, Kirchstetter TW, Lunden MM, Marshall JD, Portier CJ, Vermeulen RCH, and Hamburg SP. High resolution air pollution mapping with google street view cars: Exploiting big data. *Environmental Science & Technology*, 51, 2018.

[5] Alexeef SE, Roy A, Shan J, Liu X, Messier K, Apte JS, Portier C, Sidney S, and van den Eeden SK. High-resolution mapping of traffic related air pollution with google street view cars and incidence of cardiovascular events within neighborhoods in oakland. *Environmental Health*, 2018.

[6] Tongshu Zheng, Michael H. Bergin, Karoline K. Johnson, Sachchida N. Tripathi, Shilpa Shirodkar, Matthew S. Landis, Ronak Sutaria, and David E. Carlson. Field evaluation of low-cost particulate matter sensors in high and low concentration environments. *Atmospheric Measurement Techniques*, 2018.

[7] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, Yilong Li, Ji Jia, and Xiaofan Jiang. Aircloud: A cloud-based air-quality monitoring system for everyone. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, 2014.

[8] Meiling Gao, Junji Cao, and Edmund Seto. A distributed network of low-cost continuous reading sensors to measure spatiotemporal variations of pm2. 5 in xi'an, china. *Environmental pollution*, 199:56–65, 2015.

[9] Aakash C Rai, Prashant Kumar, Francesco Pilla, Andreas N Skouloudis, Silvana Di Sabatino, Carlo Ratti, Ansar Yasar, and David Rickerby. End-user perspective of low-cost sensors for outdoor air pollution monitoring. *Science of The Total Environment*, 607:691–705, 2017.

[10] Wan Jiao, Gayle Hagler, Ronald Williams, Robert Sharpe, Ryan Brown, Daniel Garver, Robert Judge, Motria Caudill, Joshua Rickard, Michael Davis, et al. Community air sensor network (cairsense) project: evaluation of low-cost sensor performance in a suburban environment in the southeastern united states. *Atmospheric Measurement Techniques*, 9(11), 2016.

[11] T. Zheng, M. H. Bergin, R. Sutaria, S. N. Tripathi, R. Caldow, and D. E. Carlson. Gaussian process regression model for dynamically calibrating and surveilling a wireless low-cost particulate matter sensor network in delhi. *Atmospheric Measurement Techniques*, 12(9):5161–5181, 2019.

[12] Bernardo S Beckerman, Michael Jerrett, Marc Serre, Randall V Martin, Seung-Jae Lee, Aaron Van Donkelaar, Zev Ross, Jason Su, and Richard T Burnett. A hybrid approach to estimating national scale spatiotemporal variability of pm2. 5 in the contiguous united states. *Environmental science & technology*, 47(13):7233–7241, 2013.

[13] Yanlin Qi, Qi Li, Hamed Karimian, and Di Liu. A hybrid model for spatiotemporal forecasting of pm2. 5 based on graph convolutional neural network and long short-term memory. *Science of the Total Environment*, 664:1–10, 2019.

[14] Zoubeida Kebaili Bargaoui and Afef Chebbi. Comparison of two kriging interpolation methods applied to spatiotemporal rainfall. *Journal of Hydrology*, 365(1-2):56–73, 2009.

[15] Rui Ma, Ning Liu, Xiangxiang Xu, Yue Wang, Hae Young Noh, Pei Zhang, and Lin Zhang. Fine-grained air pollution inference with mobile sensing systems: A weather-related deep autoencoder model. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(2), June 2020.

[16] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, Yilong Li, Ji Jia, and Xiaofan Jiang. Aircloud: A cloud-based air-quality monitoring system for everyone. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, page 251–265, New York, NY, USA, 2014. Association for Computing Machinery.

[17] Lixin Li, Travis Losser, Charles Yorke, and Reinhard Piltner. Fast inverse distance weighting-based spatiotemporal interpolation: a web-based application of interpolating daily fine particulate matter pm2. 5 in the contiguous us using parallel programming and kd tree. *International journal of environmental research and public health*, 11(9):9101–9141, 2014.

[18] P Banerjee, S Ranu, and S Raghavan. Inferring uncertain trajectories from partial observations. In *IEEE International Conference on Data Mining*, pages 30–39, 2014.

[19] Xiucheng Li, Gao Cong, and Yun Cheng. Spatial transition learning on road networks with deep probabilistic models. In *ICDE*, pages 349–360, 2020.

[20] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. Modeling trajectories with recurrent neural networks. In *IJCAI*, 2017.

[21] Hao Wu, Jiangyun Mao, Weiwei Sun, Baihua Zheng, Hanyuan Zhang, Ziyang Chen, and Wei Wang. Probabilistic robust route recovery with spatio-temporal dynamics. In *SIGKDD*, page 1915–1924, 2016.

[22] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[23] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

[24] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning*, pages 1067–1075. PMLR, 2013.

[25] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[29] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

[30] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[31] Alexei Botchkarev. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*, 2018.

[32] J. Kocijan, R. Murray-Smith, C.E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *Proceedings of the 2004 American Control Conference*, volume 3, pages 2214–2219 vol.3, 2004.

[33] Rachel T. Silvestrini, Douglas C. Montgomery, and Bradley Jones. Comparing computer experiments for the gaussian process model using integrated prediction variance. *Quality Engineering*, 25(2):164–174, 2013.

[34] Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian process regression: Active data selection and test point rejection. In *Mustererkennung 2000*, pages 27–34. Springer, 2000.

[35] Austin Xu and Mark Davenport. Simultaneous preference and metric learning from paired comparisons. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 454–465. Curran Associates, Inc., 2020.

[36] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[37] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *COMPUTER NETWORKS AND ISDN SYSTEMS*, pages 107–117, 1998.

[38] Dhruv Agarwal, Srinivasan Iyengar, Manohar Swaminathan, Eash Sharma, Ashish Raj, and Aadithya Hatwar. Modulo: Drive-by sensing at city-scale on the cheap. In *Proceedings of the 3rd ACM SIGCAS Conference on Computing and Sustainable Societies*, 2020.

[39] Alex Kendall and Yarin Gal. What uncertainties do we need in computer vision. In *NeurIPS*, 2017.

[40] Abhinav Kumar, Tim K. Marks, Wenxuan Mou, Ye Wang, Michael Jones, Anoop Cherian, Toshiaki Koike-Akino, Xiaoming Liu, and Chen Feng. Luvli face alignment: Estimating landmarks' location, uncertainty, and visibility likelihood. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[41] Abhinav Kumar, Tim K Marks, Wenxuan Mou, Chen Feng, and Xiaoming Liu. Uglli face alignment: Estimating uncertainty with gaussian log-likelihood loss. In *ICCV Workshops on Statistical Deep Learning in Computer Vision*, 2019.

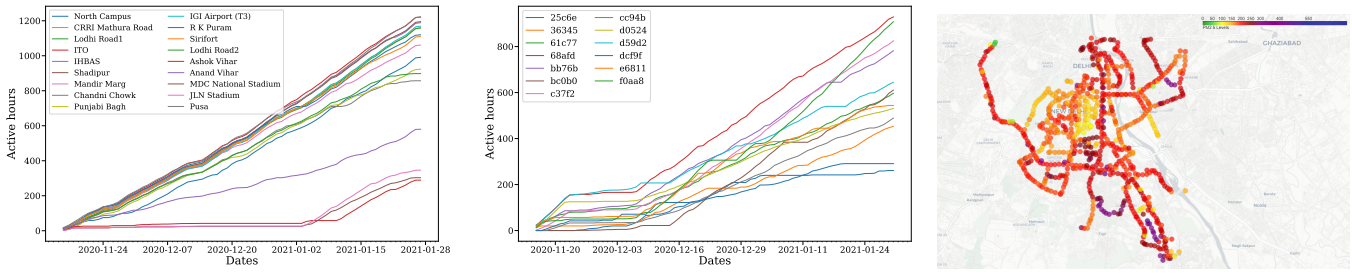
[42] Hongjie Guo, Guojun Dai, Jin Fan, Yifan Wu, Fangyao Shen, and Yidan Hu. A mobile sensing system for urban monitoring with adaptive resolution. *Journal of Sensors*, 2016, 2016.

[43] Matthew D. Adams and Denis Corr. A mobile air pollution monitoring data set. *Data*, 4(1), 2019.

[44] Jason Jingshi Li, Boi Faltings, Olga Saukh, David Hasenfratz, and Jan Beutel. Sensing the air we breathe: The opensense zurich dataset. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, page 323–325. AAAI Press, 2012.

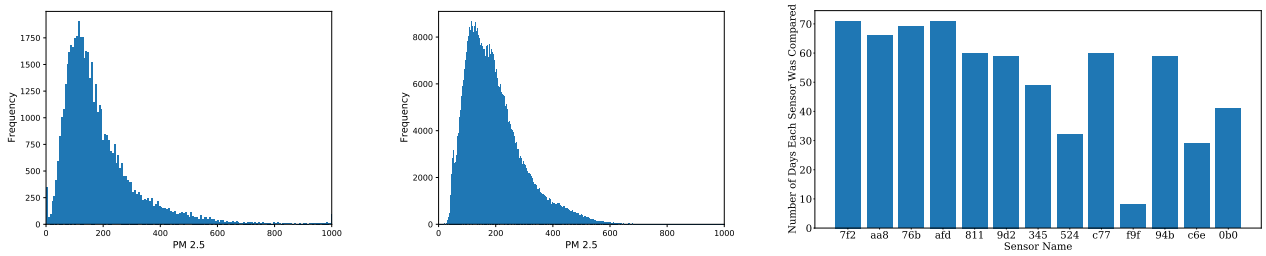
[45] Joshua S. Apte and Pallavi Pant. Toward cleaner air for a billion indians. In *Proceedings of the National Academy of Sciences*, 2019.

[46] N. Ojha, A. Sharma, M. Kumar, I. Girach, T. U. Ansari, S. K. Sharma, and S. S. Gunthe. On the widespread enhancement in fine particulate matter across the indo-gangetic plain towards winter. In *Scientific reports*, 2020.



(a) Static Sensors (b) Mobile sensors (c) Mean PM 2.5 in Jan

Figure 7: Cumulative active hours for (a) static and (b) mobile sensors between 16^{th} Nov to 30^{th} Jan. (c) Average PM 2.5 in Jan.



(a) Static PM 2.5 histogram (b) Mobile PM 2.5 histogram (c) Comparison with static sensor

Figure 8: Histograms of PM 2.5 collected by (a) Static and (b) Mobile instruments. The distributions are similar across the two sets of instruments. (c) Number of days each mobile sensor comes close to and is compared against at least one static sensor.

5 SUPPLEMENTARY MATERIAL

5.1 Related Work

Vehicle mounted air pollution sensing has been tried in scientific literature before [1–5, 42–44]. Of these, only two studies have made their datasets publicly available, namely [43] and [44], which collected data in Ontario, Canada and Zurich, Switzerland respectively. Among these two, the Zurich dataset does not include PM values. Compared to the Canada dataset, our dataset is 1000 times larger and has a significantly different distribution of PM values. This is understandable as Delhi, the area under our study is an air pollution hotspot, whereas Zurich and Ontario have negligible PM levels. We perform a detailed statistical comparison of our dataset vs. the Canada PM dataset, as shown in Table 6 and Table 7, which show higher spatial and temporal variations in PM, as well as much higher mean PM values. Regarding generalizability of the dataset, Delhi-NCR pollution is likely to correlate well with other regions in the Indo-Gangetic plain, and portions of Pakistan [45, 46], which is one of the most densely densely populated regions of the world.

5.2 Data Volume Compared to Static Sensors

Our mobile sensors spatially cover more area compared to static sensors. As shown in Fig 2(d), the static sensors do *point* measurements where the location icons are, while the buses cover the entire region shown as location heatmaps. However, the static sensors get values for the point location at all times, whereas mobile sensors collect some values at a location and move. Thus, we compare the overall temporal volume of data collected by the two sensor types to compare their data quantities.

Table 6: Delhi and Canada datasets [43] details

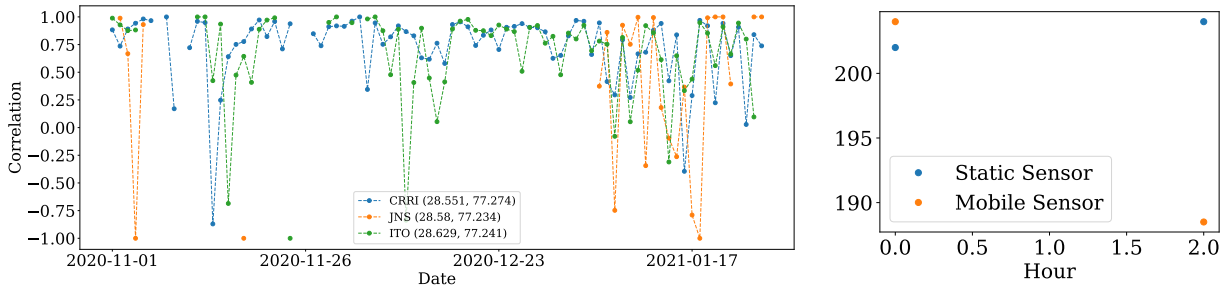
Metric	Delhi	Canada
Total area	559 sq. km	1138 sq. km
Total samples	12,542,183	46080
Samples with PM2.5	12542183	12154
Pollutants covered	PM1, PM2.5 and PM10	CO,NO,NO_2,SO_2,O_3, PM1, PM2.5 and PM10
Vehicles used	Public bus	Commercial van
Monitoring days	91	114

Table 7: Statistical comparison of PM values of Delhi and Canada datasets [43]

Metric	Delhi			Canada		
	PM1	PM2.5	PM10	PM1	PM2.5	PM10
Mean	120.35	207.92	226.11	46.45	15.08	12.15
Std-dev	57.27	114.36	123.86	97.36	12.87	9.02
Missing %	0	0	0	72.24	73.62	71.71

For static sensors, an active hour must have at least one value reported, as their sampling frequency is low. For mobile sensors, a device is said to be active in an hour if it sends at least 800 data points in that hour. This indicates the mobile device being active for at least 40 minutes in the hour with ideal sampling rate of 20 data points in a minute. Figure 7(a)-(b) show cumulative active hours per device from 16^{th} November 2020 to 30^{th} January 2021 for static and mobile sensors. Each curve belongs to a different instrument. When a curve is flatter and more parallel to x-axis, then the number of active hours for that instrument is less on that particular day. Similarly, if the curve is steeper and is more parallel w.r.t y-axis, then there are more active hours that day.

From the plots, we observe that most of the static sensors are active for most days though they have much less values per hour (1 to 5 readings). The mobile sensors are anomalous initially, when the mobile network was being setup, until around 10^{th} of December as indicated by the lower slopes of the curves before that date. After Dec 10^{th}, the mobile sensors become ideal for the remaining days, with more active hours per day and also with many more



(a) (b)

Figure 9: (a) Correlation between hourly averages of static and mobile sensors from 1^{st} November 2020 to 30^{th} January 2021. The three line plots correspond to three different locations where we consistently found the mobile sensors to be passing by the static sensor. We see that most of the instances have a positive correlation indicating reliability of our low-cost sensors. (b) Shows a common example of negative correlation between the static and mobile PM values. The correlation here is -1. The PM values recorded by both the sensors are extremely close in magnitude and thus the negative correlation can be ignored.

samples per hour (800 to 1200 readings) compared to static sensors. Thus, the mobile dataset adds significant quantity of data for the environmentalists to analyze, once the mobile network stabilizes. Figure 7(c) shows the average PM 2.5 values for Jan 2021, after the mobile network stabilizes.

5.3 Measured PM Compared to Static Sensors

We additionally compare the quality of our mobile PM measurements with static sensor measurements in nearby locations. Fig. 8(a)-(b) show the distributions in PM 2.5 as measured by static and mobile sensors, both sets showing similar PM distributions, in spite of the difference in heights they have been installed at, the difference in the amount of exposure of the sensor to direct smoke and dirt, and the difference in PM measurement technique.

We next correlate static and mobile sensor readings more directly. We first locate the mobile sensors that were close (less than or equal to a distance of 150m), to any static sensor. We found three static sensors satisfying this criteria, which were installed at CRRI Mathura Road, Delhi, Jawaharlal Nehru Stadium, Delhi and ITO, Delhi respectively (referred to as CRRI, JNS and ITO respectively in Fig 9). Fig. 8(c) shows the number of days that a mobile sensor is calibrated against one of the static sensors.

We compute the correlation between the hourly mean of all PM2.5 values recorded by a static sensor and its nearby mobile sensors. Since the PM values should roughly move from one hour to the other hour in the same direction (increase or decrease), we expect to see a high positive correlation between both the hourly averages. Fig. 9 (a) shows the daily correlation values of all three locations from 1^{st} November 2020 to 30^{th} January 2021. Region corresponding to JNS lacks some values, as static sensor data was unavailable for comparison in that period. We observe a high correlation across most days. We also found 15 instances where the correlation was found to be negative. Fig 9 (b) shows a common example of the type of instances we found to give out negative correlation. Correlation does not take into account how close in magnitude the recorded PM values are. These were cases where PM values were extremely close in magnitude and thus, their trend became trivial. Overall, the correlation with static sensors is remarkable, given the positioning and measurement technique differences. We had 177 total correlation values out of which 103 were found to be significant at $\alpha = 0.05$. The mean correlation of this subset was 0.85. Only one significant negative correlation was found.

5.4 Additional Experiments for Location Recommendation

We next perform some ablation studies.

Predictor GCN Layers: For the presented analysis, we took two layers of Graph Convolution followed by an MLP layer. We modified the number of Graph Convolution layers to observe its impact on the RMSE convergence. Table 8 shows the RMSE and Std deviation for dataset 2 and MaxMax Selector for 1, 2 and 5 Graph Convolution layers. We find that the Predictor is performing good with 2 such layers.

Table 8: Impact of Graph Convolution layers in Predictor

NumLayers	RMSE (Std deviation)
1	70.7 (0.2)
2	64.0 (0.6)
5	64.5 (1.7)

Feature Ablation in Selector: We also experimented to analyze the impact of the various Selector state-features on the RMSE convergence over various feature combinations.

Table 9 shows the RMSE performance. We observe that PageRank is the dominating feature among the three features, and involving all features gives the best performance (with minimum Std Deviation).

Table 9: Impact of State Feature Ablation

Features			Performance
PageRank	RMSE	Neighbour	RMSE (Std deviation)
✓	✓	✓	79.0 (0.4)
✓	✓	×	80.7 (2.5)
✓	×	✓	80.4 (1.9)
×	✓	✓	83.5 (3.7)
✓	×	×	80.7 (2.1)
×	✓	×	87.3 (5.7)
×	×	✓	84.9 (3.8)