A Supplementary Material

A.1 Dataset Documentation

We provide statistics and plots describing our dataset to give a high level overview of the properties and distribution of the variables. Our tables and plots have been inspired by the format in [42]. In tables 3, 4 and 5 we provide basic statistics on the different nominal, ordinal and continuous variables present in our dataset.

Table 3: Description of Nominal Variables

Name	Type	Count	Unique entries	Most frequent	Least frequent	Missing
deviceId	string	12542183	13	00000000c37f0aa8 (1393279)	000000008f525c6e (427436)	0%
uid	string	12542183	12542183	multiple detected	multiple detected	0%

Table 4: Description of Ordinal Variables

Name	Type	Count	Unique entries	Most frequent	Least frequent	Missing
dateTime	string	12542183	5725406	2021-01-13 20:42:05 (12)	multiple detected	0%

Table 5: Description of Continuous Variables

Name	Type	Count	Min	Median	Max	Mean	Std	Missing	Zeros
lat	number	12542183	28.48655	28.57932	28.72000	28.57855	0.050018	0%	0%
long	number	12542183	77.10014	77.24073	77.32076	77.25250	0.03945	0%	0%
pm1_0	number	12542183	1	110	1730.5	120.34796	57.27231	0%	0%
pm2_5	number	12542183	1	183	1792	207.92479	114.36323	0%	0%
pm10	number	12542183	1	199	1903	226.11056	123.8647	0%	0%

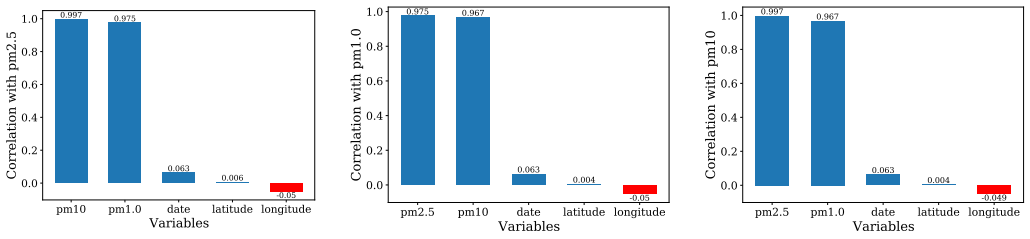


Figure 8: Correlation of different variables with PM2.5, PM1 & PM10 values across the dataset

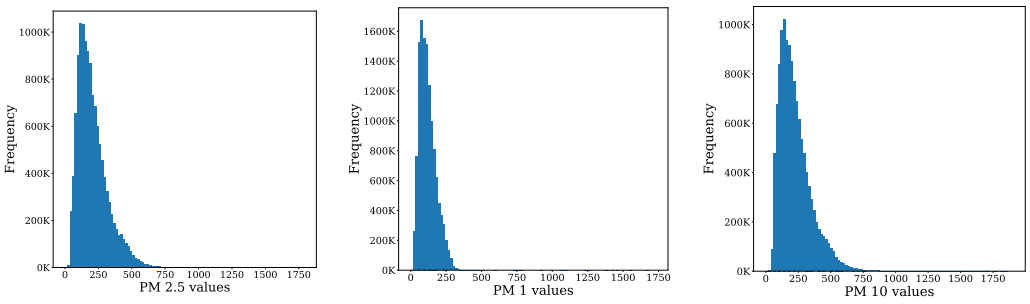


Figure 9: Frequency distribution of PM2.5, PM1 & PM10 values across the entire dataset

Fig 8 shows how the different variables in our dataset correlate with our PM variables while 9 shows the distribution of the PM variables. We see that the three PM sizes of 2.5, 1.0 and 10 are highly correlated in Fig 8, while in absolute values, PM 2.5 and PM 10 see higher magnitudes than PM 1.0 in 9.

Fig 10 (a) shows the distribution of our devices over the entire dataset. It is represented by a bar graph over the different devices. 'f0aa8' is the most frequent device whereas '25c6e' is the least

frequent device. Fig 10 (b)-(d) show our average PM2.5[4] plots for November, December and January separately. We average out all the PM2.5 values for each month, after rounding the latitude and longitudes till the 3rd decimal place, and plot the rounded points in our trajectories. We observe two things here. The first, and most immediate, is that the trajectories differ for all the three months. Since our devices are attached to public buses, the routes often change and this is observed in the plots. The second is that the PM2.5 distribution varies from one month to the next. (b) shows that November has relatively lesser pollution levels across the city with only a small number of regions showing heavy pollution levels. The circles plotted are light in color except around some areas, including the center of our area of study, where the Kushak Nallah bus depot is located. (c) shows December with an increase in the pollution levels with a cluster of really dark points in the center of our area of study. The trajectories change and include a larger area now. Finally, (d) shows that January also has fairly high PM, with it's own regions of densely located dark points. Both December and January indicate the winter season peak and thus, higher average pollution levels are expected.

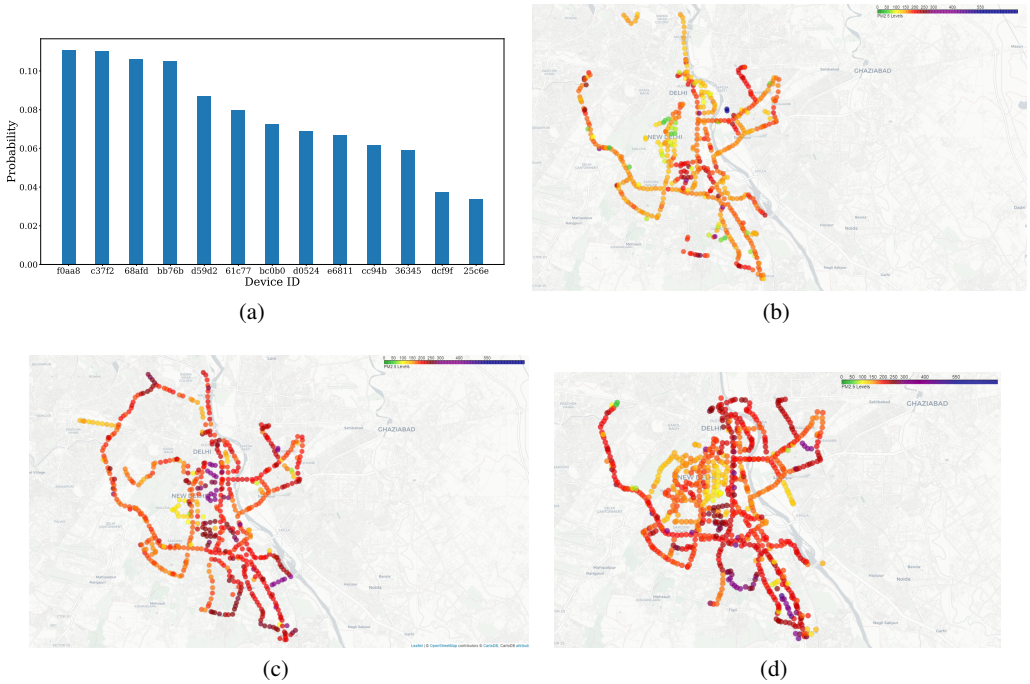


Figure 10: (a) Contribution of individual devices in the entire dataset, in terms of number of records collected.'f0aa8' is the most frequent device whereas '25c6e' is the least frequent device. (b)-(d) Average PM2.5 maps for each month in our data. These maps show the average PM2.5 values in each month for the points in our trajectories. Lighter colored points indicate lesser pollution levels while darker colored points indicate mode pollution levels. (b) November has relatively lesser pollution levels across the city with only a small number of regions showing heavy pollution levels. (c) December shows an increase in the pollution levels with a cluster of really dark points in the center of our area of study. (d) shows that January still has higher pollution levels.

A.2 GPS data cleaning details

As discussed in Section 2, our GPS entries are sometimes 0, when our IoT unit cannot lock the satellites through the metallic body of the bus, especially when the bus moves under a bridge or flyover, or passes an underground tunnel. To deal with these, we formulate a problem as follow: suppose we know the location tuples (x_1, y_1) and (x_2, y_2) for a bus at timestamps t_1 and t_2, we wish to find its possible location at time $t \in (t_1, t_2)$. We solve this using Map Matching or Linear Interpolation, both of which give good accuracy, with the latter having lower runtimes. We detail the two methods here.

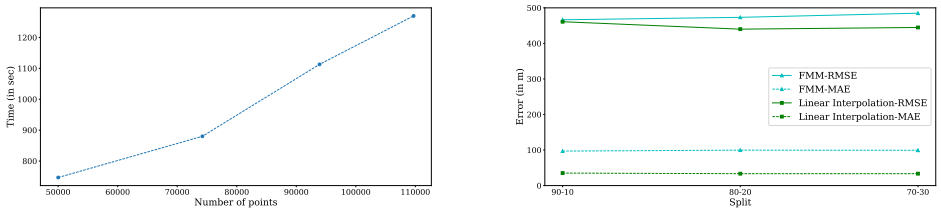
[4]PM values are measured in μg/m^3

Map Matching: Map matching is a commonly performed process to infer a path on a road network from a noisy GPS trajectory [18]. A map-matching algorithm takes recorded serial location points and relates them to edges in an existing street graph (network) in order to represent the trajectory of a user or vehicle. We used Fast Map Matching[5] (fmm), an open source map matching framework in C++ and Python based on [19]. It provides detailed matching information and scales to millions of GPS points and road edges. This helps us achieve two objectives. The first being filling the missing points and the second being correcting any non-missing points in case the recorded values are shifted from the predicted route. After implementing this however, we faced a number of challenges which included the breaking of the algorithm if any two consecutive points were too far, entire trajectories being unmatched if algorithm failed at any point, extensive tuning of the algorithm's parameters and the algorithm taking a lot of time to compute. Fig. 11 (a) shows the time taken on the data of four different days along with the length of trajectory. For a given graph size, the time taken by map-matching is directly proportional to the number of points in the GPS trajectory to match.

Linear Inteporlation: Linear interpolation could be easily used in our case because of the high average sampling rate of devices (one ping per 3 sec). For such small distances, linear interpolation serves as a good enough approximation. Also, as linear interpolation consists of simple calculations, it is an extremely fast method (around 1 second; constant in input size) to apply which makes it easy to process the datasets. We fixed a threshold distance, Δ, up to which linear interpolation is considered to be valid. To fill any missing point P recorded at time t, we did the following:

Identify non-missing points just before and after t, say $P_1 : (x_1, y_1, t_1)$ and $P_2 : (x_2, y_2, t_2)$ respectively. Calculate distance d between P_1 and P_2. If d is greater than Δ, leave P as it is, otherwise, fill P as $\left(x_1 + \frac{t-t_1}{t_2-t_1}.d.(x_2 - x_1)\right), \left(y_1 + \frac{t-t_1}{t_2-t_1}.d.(y_2 - y_1)\right)$

Evaluation: For our data, with Δ as 200m, more than 95% of the missing points, on an average, could be filled. We designed an experiment to compare map-matching and linear interpolation. From the list of non-missing points, 10-30% GPS points were randomly sampled and kept as test points. Using the remaining 70-90% known points, these test points were predicted using the two methods and the root mean square error (RMSE) and the mean absolute error (MAE) [35] of the Haversine distance between the predicted coordinates and the true coordinates were computed. The results of the experiment are shown in Fig. 11 (b). It can be seen that linear interpolation performs better than fmm. Hence, we decide to move ahead and fill the incorrect points with linear interpolation.



(a) (b)

Figure 11: Results of the two interpolation mechanisms. (a) shows the time taken by map-matching on four different days. Map-matching takes a lot of time and thus isn't useful for where time is a constraint. (b) shows the distribution of errors (RMSE and MAE) for different filled-missing points splits for both the methods. (Threshold Δ = 300m for linear interpolation)

A.3 Spatio-temporal Interpolation Baselines

The different spatio-temporal interpolation baseline models have been described below. The description includes the preprocessing, parameter and hyperparameter settings.

Prepocessing: The preprocessing was kept model specific where it could include normalization of the features and target variables, downsizing of the train dataset and finally, creating a graph using the dataset. These were experimented upon using the datasets from different arbitrary days and the best settings were kept for the benchmarking process. The preprocessing details for each model have been mentioned as a part of the model details below.

[5] https://github.com/cyang-kth/fmm

GPR: The Gaussian Process (GP) regression model is defined by specifying the mean and the covariance function the GP prior [30]. 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 [43] that learns the covariance between the different target values based on the feature values. The covariance or kernel function thus captures the relationship between the PM values and how they vary with changes in our spatial and temporal features. The most common kernels used in a GPR model are squared exponential, Matern (MA), rational quadratic and periodic kernels [44]. The kernel function has free hyperparameters that are learnt while optimising the GPR model. The standard hyperparameters to be learnt are the variance and lengthscale of the kernel along with the variance of the Gaussian noise in the model.

Firstly, a downsized version of the train dataset is created to facilitate tractable training of the model. The 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) [30] 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 [45] to code them.

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.

GraphSAGE: This model has been heavily inspired by GraphSAGE [32]. We aim at learning universal weights, similar to GraphSAGE, 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. It is important to note that our model doesn't use feature values for a point as part of our node definition. Rather, only the pollutant value is defined as the node value and the correlation in pollutant values based on our changes in feature values helps us in predicting our final node value for unknown nodes. Firstly, a graph is created from a downsized version of the train dataset. The 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 and making this one point represent one single node.

An edge is then created between two different nodes only if both the nodes lie within 2 hours of each other temporally and around 2km away from each other spatially. The weight function for aggregation of edge was a polynomial encoding an approximate form of dependence of distance and time between any two nodes with the power of the terms being considered as hyperparameters. Weights of the edges were 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). During edge formations, to resolve the issue of node isolation in cases of extremely distant nodes, we set the minimum edges to 4 so as to prevent edge scarcity.

Once formed, 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. These are similar to how they are defined in the original GraphSAGE model. These include the mean aggregator and max aggregator too, as seen in the original GraphSAGE model. After the two layers, there are 3 fully connected neural network layers to predict the final pollutant value. The parameters of these layers are then learnt when this model is trained using the graph created earlier. 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[46] to code them.

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.

ANN: Extensive testing was performed with various multilayer perceptron models with varied hyperparameters to find the best architecture. We had three input features corresponding to latitude, longitude and datetime. Target variable was PM2.5 value. All the hyperparamters were tuned based on the train and validation datasets.

For preprocessing, we found the mean of PM2.5 values for all observations which were temporally 15 minutes apart. Then we performed normalization of input and output features in order to make their mean zero and standard deviation 1. 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 [33] was used. Further, Rectified linear unit (ReLU) was used as the activation function for all layers. For optimization, we used Adam [34] 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 [47].

A.4 Anomaly metrics computation details

As discussed in Section 5, we compute six anomaly metrics on our dataset. Each of these metrics needs several thresholds, which we find by running some heuristics to compute the relevant statistics from the dataset. The heuristic to compute the first anomaly metric of **samples recorded per minute** is given in Algorithm 1. This metric checks for faulty devices which might be sampling more or less than expected rate. Fig. 5a in the paper shows ideal samples collected per minute should be around 20. If it deviates too much, that device is anomalous. The amount of deviation allowed is calculated statistically by observing the distributions for several days. Our algorithm finds the upper bounds and lower bounds of the median (Θ^L_{50}, Θ^U_{50}), 25^{th} percentile (Θ^L_{25}, Θ^U_{25}) and 75^{th} percentile (Θ^L_{75}, Θ^U_{75}) of the expected distribution.

Algorithm 1 Finding Metric 1 parameters

Step 1 : Define a set of days(n) based on which we want to determine the thresholds (X). These days can be manually picked by observing the box plots as in figure 5a.
Step 2: For each day and each device we collect the number of readings from active minutes.
Step 3: From the set of distributions(n), we collect median, 25^{th} percentile and 75^{th} percentile of each distribution. Lets denote it by D_{50},D_{25},D_{75} respectively.
Step 4: To find a considerable range of these parameters we choose 50 percent of central data ruling out first and last 25 percent of data as outliers.
Step 5: Θ^L_{50} = 25^{th} percentile of D_{50} Θ^U_{50} = 75^{th} percentile of D_{50}
Θ^L_{25} = 25^{th} percentile of D_{25} Θ^U_{25} = 75^{th} percentile of D_{25}
Θ^L_{75} = 25^{th} percentile of D_{75} Θ^U_{75} = 75^{th} percentile of D_{75}

Other metrics use similar heuristics. Fig. 12 shows the summary of anomalies detected based on these heuristics and the thresholds listed in Table 6. Fig. 12(a) shows the summary for metrics 1 and 2, which compute **samples recorded per minute** and **number of minutes active in an hour** respectively. Both these metrics show anomalies in the early phase of the mobile sensor deployment, and the anomalies become rarer after Dec 10^{th}, 2020. This is similar to the flat part parallel to x-axis in Fig. 6(b) in the paper, which showed **number active hours in a day** i.e. metric 3. The early phase of IoT deployment saw issues in 4G radio signals based on the placement and packaging of the unit, which was changed until data collection became smooth and anomalies for these metrics became rare. The remaining few anomalous points for the rest of Dec, 2020 and Jan 2021, resulted from occasional 4G signal losses, which is common in urban tunnels in Delhi as the bus travels. 1-2 units consistently performed poorly, which have been debugged to have power supply issues from the bus.

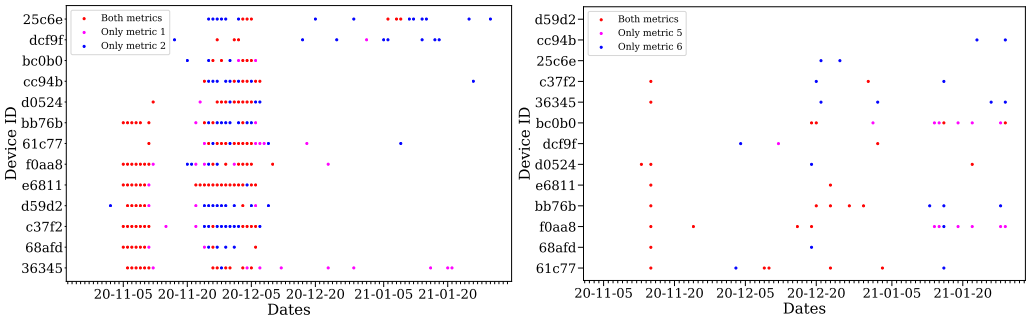


Figure 12: Anomalies detected by our heuristics on the whole dataset (Nov 1^{st}, 2020 to Jan 31^{st}, 2021). Presence of marker represents anomalous device for corresponding date. (a) shows combined anomalies for metrics 1 and 2. (b) shows combined anomalies for metrics for metrics 5 and 6.

Fig. 12(b) shows the summary anomalies for metrics 5 and 6, which compute **inter-sensor** and **intra-sensor** differences in recorded PM values. We again used the detected anomalies to check the devices and the deployment partner (bus company). In most cases, a deviation of a sensor from other sensors happened when there was local electrical maintenance work like soldering ongoing in the bus, a common phenomena when the buses are parked in the depot at night. 1-2 units had some persistent problem towards the end of month Jan, which on checking were found to accumulate some dust near the fan that suck in air. Some cleaning of the units is therefore recommended, which can be done during regular maintenance of the buses in the depot.

Based on our experience of running a live IoT network on public buses in a developing country, such anomaly detection is vital for quick debugging and fixing of issues. These plotted anomalies are therefore included as csv files in our dataset, so that ML researchers interested in automated anomaly detection methods for IoT networks, can use these as reference ground truth. The heuristic codes are also open-sourced, so that the thresholds can be adjusted based on the ML researchers requirements of more or less conservative anomaly detection.

The summary anomalies are represented in one-hot encoded format in csv files. In all the files (except for metric 4), the first column contains the dates for three months and the first row contains all the 13 device Ids. Presence of 1 indicates anomaly for corresponding date and device. In the file of metric 4, the first row represents the 16 regions in consideration rather than the device Ids.

[6]Please refer Algorithm 1 for detailed meanings of D_{25}, D_{50}, D_{75}. The parameter values are found based on date Jan 16 to Jan 31 2021

[7]Please refer section 5, Anomaly metric 4 for detailed description of the symbol and calculation of its value.

[8]Please refer section 5, Anomaly metric 5 & 6 for detailed description of the symbols and calculation of their values.

Table 6: Thresholds used in computation of the six anomaly metrics

Symbol	Metric	Value	Description
Θ^{L}_{25}	1 2	19 20.0	Lower bound of sampling rate, 25^{th} percentile of D_{25} [6]
Θ^{U}_{25}	1 2	22 38.5	Upper bound of sampling rate, 75^{th} percentile of D_{25} [6]
Θ^{L}_{50}	1 2	20 38.0	Lower bound of sampling rate, 25^{th} percentile of D_{50} [6]
Θ^{U}_{50}	1 2	23 59.0	Upper bound of sampling rate, 75^{th} percentile of D_{50} [6]
Θ^{L}_{75}	1 2	21 57.6	Lower bound of sampling rate, 25^{th} percentile of D_{75} [6]
Θ^{U}_{75}	1 2	24 60.0	Upper bound of sampling rate, 75^{th} percentile of D_{75} [6]
γ	3	1(static sensor) 800(mobile sensor)	Minimum data points in an hour to be termed as Active hour
τ	3	variable	Minimum number of active hours for which a sensor must report data to be termed as ideal for the day
δ	4	41	Maximum allowed percentage difference of a region with its moving average [7]
Θ_{25}	5 6	variable	25^{th} percentile of distribution of PM values of a device during an hour
Θ_{75}	5 6	variable	75^{th} percentile of distribution of PM values of a device during an hour
max_IQR	5 6	39.77	Maximum allowed IQR, 90^{th} percentile of distribution of all IQRs in training set [8]
buffer	5 6	105.45 46.36	Used for calculating the majority PM range [8]

A.5 Comparison of Static and Mobile Sensors

The three static sensors against which we compare mobile sensor readings are strategically selected to ensure that the accuracy of *all* mobile sensors can be evaluated. To elaborate, the three static sensors are located in three diverse but highly busy intersections. Hence, a large portion of the mobile sensors get spatio-temporally close to at least one of the static sensors. To give some concrete statistics, on average 78% of the mobile sensors get calibrated against once of the three static sensors per day. Across the entire time-duration, *all* mobile sensors have been calibrated at least over 8 days against one of the static sensors.

Fig.13 (a) shows that if we compute the proportion of mobile sensors we get to compare with the three static sensors, we find that almost all our sensors are being included in the comparison. This experiment was meant to take the static sensor values as ground truth and compare our values with theirs so as to make sure that our data is reliable. Since we are taking almost all our sensors into account when performing the comparison, we claim that the comparison has been done in a fair manner without compromising on its quality.

Finally, Fig.13 (b) shows the number of days each sensors was included in the comparison. The sensors names have been shortened to their last three characters here. Nine of our sensors have been included on around 50 days while only one sensor has been included for less than 10 days.

We next discuss why the specific three static sensors were chosen for accuracy calibration of mobile sensors. We had two constraints for selecting the static sensor observations that could be compared with our mobile sensor observations:

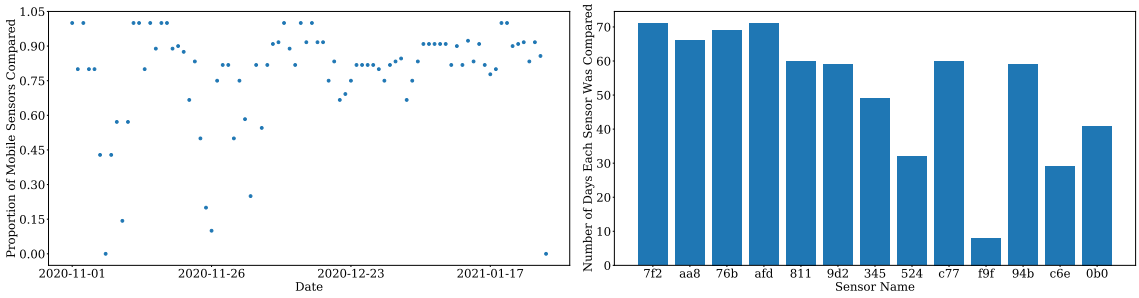
1. Both the mobile and static sensor have to be spatially close i.e our mobile sensors can't be more than approximately 150m away from the static sensor if we want to compare their PM recordings.

2. The time at which they recorded their observations while they were spatially close also has to be temporally close i.e. they should be spatially close in the same hour in which they recorded their values.

To explain the above better, consider the following scenarios:

1. If none of our buses ever pass by a static sensor (pass by meaning coming as close as 150m to a static sensor), then we wouldn't be able to compare any PM values.
2. If one of our bus passes by a static sensor at 4:30 PM and the static sensor doesn't record its PM value during 4-5 PM, then we won't be considering these values for comparison.
3. If one of our bus passes by a static sensor at 12:00 PM and the static sensor records a value at 12:45PM, then we will be considering these values for comparison.

Thus, the above process really thins out the number of static sensors we can compare our values reliably with. If we lessen the severity of the temporal or spatial constraints then we end up decreasing the reliability of our comparison as we are increasing the number of fluctuations that could potentially occur in the PM values being compared. As an example, for 2^{nd} January 2021, considering a maximum distance of around 1500m allows us to compare our mobile sensors with 7 different static sensors, but this affects the reliability negatively since ideally you would want to compare the PM values being measured by both the type of devices at the exact same place. This trade-off between the number of sensors and the reliability is exacerbated by the irregularity of the static sensor's PM value recordings. Their recordings aren't as dense as ours and are too sparsely recorded. Owing to the above constraints, we choose the three static sensors that are spatio-temporally close to a large volume of mobile sensors consistently over the entire time duration.



(a) (b)

Figure 13: (a) Proportion of mobile sensors compared with the 3 static sensors throughout the deployment period. We observe that the values from most of our mobile sensors are included in our comparison analysis as different buses happen to cross the static sensors at different times throughout the day. (b) The number of days each sensor was included in the comparison.

A.6 Variations in Experiments

To report our baselines in a fair manner, we report them by running each model on three different random splits of the five randomly chosen dates. Thus all the results in Table7 have been obtained by averaging the results across all the fifteen runs for each model. Additionally, we also report the standard deviation across all runs. The format below is $Mean \pm StandardDeviation$. Finally, please keep in mind that since the resources on the free version of Colab are distributed dynamically, the training times below are merely indicative and can should be used to get a rough estimate of which method takes more or less time.

Table 7: Interpolation Baselines with Variations

Models	Metrics					
	TRAIN_RMSE$_{100}$	TEST_RMSE$_{100}$	TRAIN_TIME$_{100}$	TRAIN_RMSE$_{FULL}$	TEST_RMSE$_{FULL}$	TRAIN_TIME$_{FULL}$
GPR	**31.70 ± 2.35**	**32.86 ± 2.77**	63.08 ± 6.43	**30.94 ± 2.33**	**32.09 ± 2.57**	403.25 ± 441.94
Variational GPR	34.05 ± 1.85	35.14 ± 2.25	570.32 ± 22.29	31.76 ± 2.15	32.92 ± 2.55	2143.58 ± 521.9
GraphSAGE	×	36.25 ± 2.57	5874.08 ± 502.74	×	35.77 ± 2.32	6000.35 ± 541.78
Meaner	×	×	×	×	35.58 ± 2.5	2145.14 ± 205.6
ANN	40.86 ± 5.12	41.88 ± 4.92	**2.70 ± 0.13**	32.90 ± 2.01	34.07 ± 2.51	**48.59 ± 22.87**

A.7 Comparison with other public datasets

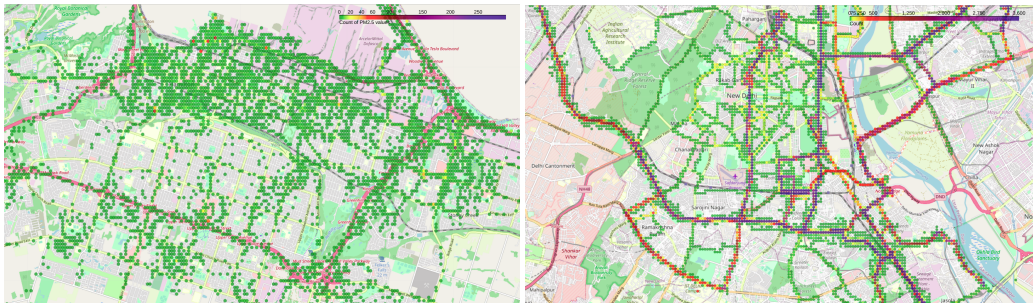
The dataset in [14] contains data about Ultrafine particles and Ozone concentration but doesn't have any PM data. Hence, it is not possible to establish a fair comparison between [14] and our dataset. But [13] has all types of PM data that is available in our dataset. Hence, we present a comparison of [13] with our own dataset based on various parameters which can help establish the significance of our work compared to previously published work in this field.

The dataset in [13] belongs to the city Hamilton in Ontario, Canada. The data was collected during 114 days of air pollution monitoring between November 2005 and to November 2016, covering a span of 11 years. There was also no standard location revisit approach involved in the data collection; hence most of the locations visited across the city have only one sample.

The dataset contains a total of 46080 records where each record has 16 different parameters. Each record in the dataset represents a one-minute integrated sample and is prepared as a line segment, i.e., the start to end points of the one-minute integrated sample. Out of the 46080 records, 13,048 records contain either PM2.5, PM1.0 or PM10 data and only 12,154 records contain PM2.5 data; other records don't have data about any of the PM. In the following sections, we provide a detailed analysis comparing the dataset in [13] with our dataset based on different parameters. We divide Delhi and Hamilton into regions by rounding off the latitude and longitude of each record to 3 decimal places and then grouping all the records belonging to each region. The data belonging to these regions has been used for comparison.

A.7.1 Comparison based on total number of data points collected across all regions

The first parameter used for comparison is total number of records collected in every region in the city across the whole duration of the data. Figure 14 shows this over the cities of Hamilton and Delhi. The color of each circle in the maps represent total number of points collected in every region.



(a) Hamilton (b) Delhi

Figure 14: Total number of data points across all regions

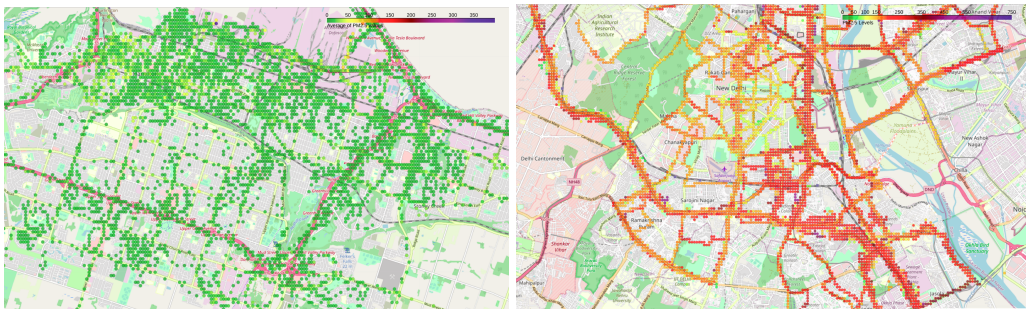
As can be seen from the maps, the circles over most of the regions in Hamilton are of green color which corresponds to 0-20 points over a region. But there is a large variation in the color of circles over Delhi which correspond to more than 500 data points over most regions. This indicates that the number of data points collected across each region in Delhi is much higher than the number of points collected across each region in Hamilton.

A.7.2 Comparison based on average value of PM2.5 level across all regions

Figure 15 shows the average of PM2.5 values recorded in every region throughout the dataset over the cities of Hamilton and Delhi. The color of each circle in the maps represent average PM2.5 in every region.

We can observe that the average PM value recorded in most of the regions in Hamilton is in 0-50 range and there are only a few regions that average of greater than 50. Almost none of the regions record an average greater than 100 across the whole duration of data.

On the other hand, in Delhi we can see that average PM for most regions is greater than 250 which is much higher than any of regions in Hamilton.

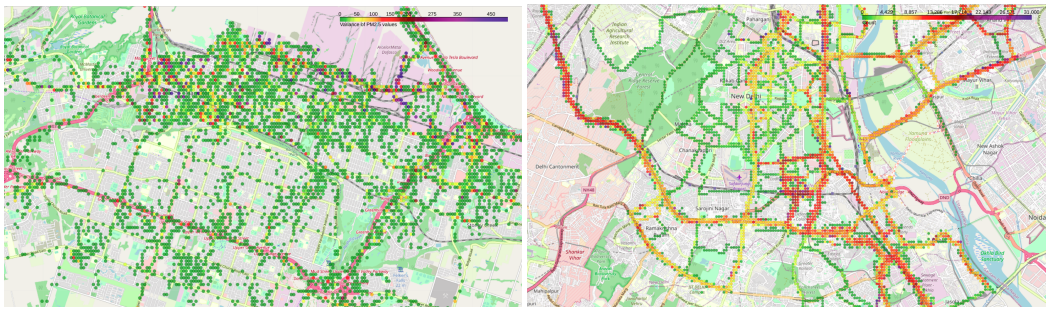


(a) Hamilton (b) Delhi

Figure 15: Average of PM2.5 values across all regions

A.7.3 Comparison based on variance of PM2.5 level across all regions

Next we compare the variance in PM2.5 level in all regions across the whole duration of data. Figure 16 shows this over the cities of Hamilton and Delhi. The color of each circle in the maps represent PM2.5 variance in every region.



(a) Hamilton (b) Delhi

Figure 16: Variance of PM2.5 level across all regions

We can see that for Hamilton, PM2.5 level varies in a very small region of 0-50 across most of the regions and there are a very few points where the variance is greater than 150. We should also note that this variance is observed in a period spanning 11 years.

On the other hand, we see very very high variance in PM2.5 levels recorded across almost all the regions in Delhi and this variance has been observed over just 3 months.

A.7.4 Comparison based on frequency of PM values

Figure 9 & 17 shows the frequency of PM values in Delhi and Canada datasets respectively. Most of the PM2.5 values lie in the range of 0 to 60 for the Canada dataset while it is in the range 0 to 750 in case of Delhi. Not only the range of PM values is high in our dataset but the frequency of each PM value is also high. Most frequent PM2.5 value in case of Hamilton is around 10 and in case of Delhi is 150. The above analysis also holds in case of PM1 and PM10 values.

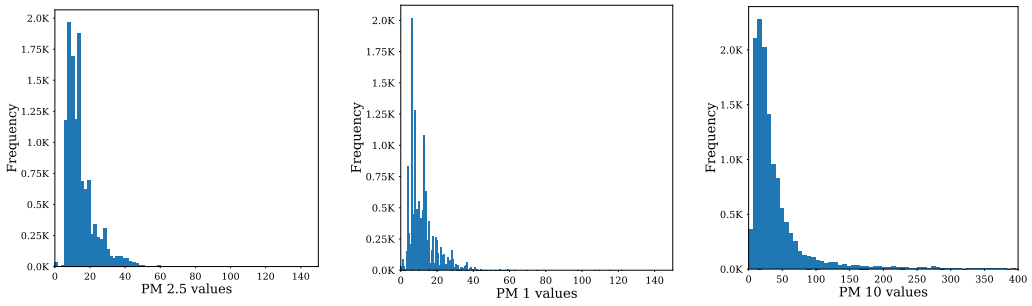
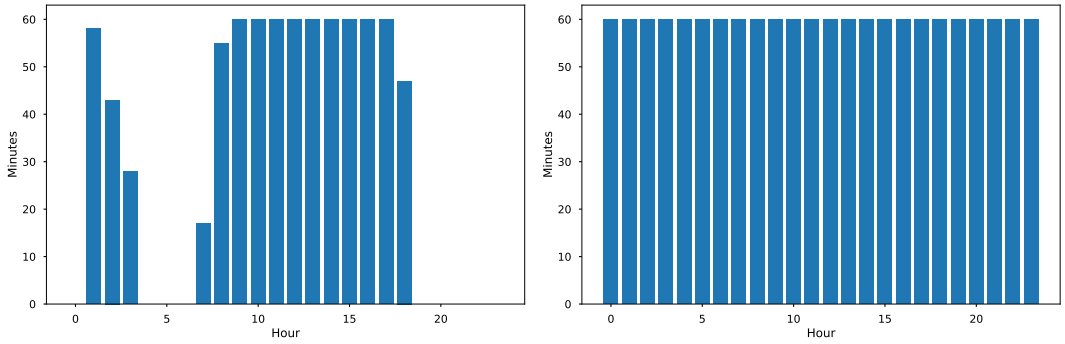


Figure 17: Frequency distribution of PM2.5, PM1 & PM10 values across the entire Canada dataset

A.7.5 Comparison based on number of hours covered across entire dataset

It is found that in Canada dataset, there are some hours where there are no samples at all which is shown in figure 18. For each hour in a day, we count the total number of minutes which have at least one sample across the whole dataset. We observe that in our dataset we have samples for each minute of each hour. Whereas in Canada dataset there are atleast 9 empty hours and most of them in the night time.



(a) Hamilton (b) Delhi

Figure 18: Number of minutes covered across each hour

Metric	Delhi dataset	Canada dataset
Total area covered	559 sq. km	1138 sq. km
Total number of samples	12542183	46080
Samples with PM2.5 value	12542183	12154
Samples with at least one PM value	12542183	13048
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
Number of monitoring days	91	114

Table 8: Comparison of Delhi and Canada datasets over different metrics

Metric	Delhi			Canada		
	PM1	PM2.5	PM10	PM1	PM2.5	PM10
Min	1	1	1	0	0	0
Max	1730.5	1792	1903	2640	731	291
Mean	120.35	207.92	226.11	46.45	15.08	12.15
Std	57.27	114.36	123.86	97.36	12.87	9.02
5th percentile	45	72	79	4	6	8
95th percentile	233	435	471	28	32	138
Missing %	0	0	0	72.24	73.62	71.71

Table 9: Statistical comparison of PM values of Delhi and Canada datasets

A.7.6 Comparison based on performance of ANN with other datasets

Finally, we compare the results of spatiotemporal interpolation for both the datasets based on the performance of our ANN baseline. We do this experiment to understand if our dataset poses a novel and interesting modelling challenge or not. It has been shown above that our dataset is much more varied than the Canada dataset but now we check how tough it is to model our dataset.

Experimental Setup
The method used for finding results on our dataset has already been described in Section 4. The Canada dataset is much smaller in size (Table 8) as compared to our dataset and thus, the entire dataset was taken for getting the interpolation results in each run as compared to five randomly chosen days' data in case of our Delhi dataset. Thus, for each of the three runs, the entire dataset was taken (106 days containing PM2.5 values) and was randomly split into train, validation and test in the ratio of 64:16:20. Separating out the Canada days into different days and then training our model on different days won't have made sense because of the sparsity of data available per day. On an average, only 114 points were available for each day. Finally, there was no need to downsize the coordinates because of the above reasons. The rest of the method used for training and finding the results was same for both the datasets.

Results
Table 10 compares the results obtained for both the datasets. As the entire dataset is taken in every run for Canada as compared to different days' data for each run in case of Delhi, we expected the variability in the results to be much more in the case of Canada. As the Canada dataset is spread out over a number of days, there should be much more variation in terms of seasonal patterns and other fluctuations across days and so, it should have been tougher to model that dataset. But we observe from the table that the RMSEs are significantly lower for the Canada dataset compared to those for Delhi. Hence, It validates our hypothesis that the amount of variation in the Delhi dataset is much higher as compared to Canada dataset and that this variation makes it tougher to model the Delhi dataset as compared to the Canada dataset. This presents an exciting opportunity for both environmentalists and the machine learning community.

Metric	Average		Standard Deviation	
	Delhi	Canada	Delhi	Canada
Total training time(in seconds)	48.59	8.75	22.87	0.56
Training RMSE	32.90	10.79	2.00	0.49
Test RMSE	34.07	11.94	2.51	0.13

Table 10: Comparison of Delhi and Canada datasets based on performance of ANN based interpolation