

Air Quality Prediction and PM_{2.5} Estimation Using Data-Driven Predictive Models

M. Lalitha Naga Durga¹, Dr. L. Sumalatha²

¹PG Scholar, Department of Computer Science and Engineering, University College of Engineering Kakinada, JNTUK, Kakinada, Andhra Pradesh, India.

²Professor, Department of Computer Science and Engineering, University College of Engineering Kakinada, JNTUK, Kakinada, Andhra Pradesh, India.

Abstract—Fine Particulate Matter (PM_{2.5}) is a major environmental health concern, particularly in India, where urbanization, industrial emissions, vehicular pollution, crop-residue burning, and dust contribute to elevated concentrations. Accurate and scalable PM_{2.5} forecasting is therefore essential for air-quality management and public health.

This study presents a controlled comparison of three data-driven forecasting methods, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Extreme Gradient Boosting (XGBoost), for hourly PM_{2.5} prediction using the CPCB India Air Quality Dataset. The dataset comprises multi-pollutant ground-station observations from 229 Continuous Ambient Air Quality Monitoring Stations across 127 cities in 21 Indian states during January 2015 to July 2020. Nine co-pollutants, including PM₁₀, NO, NO₂, NO_x, NH₃, CO, SO₂, and O₃, are used as input features. Missing values are handled through linear interpolation, features are Min-Max normalized using training data only, and five-hour sliding windows are constructed with an 80:20 chronological train-test split.

Under identical experimental conditions, the 64-unit GRU achieved the best performance, obtaining an RMSE of 0.0349 and an R² of 0.484, outperforming the 10-epoch LSTM with RMSE = 0.0371 and R² = 0.417. The 5-epoch LSTM showed substantially lower performance with R² = 0.100, highlighting the importance of adequate training. XGBoost provided a lightweight, deterministic alternative suitable for CPU-based and edge deployment. The results demonstrate that GRU offers an effective balance between predictive accuracy, model complexity, and computational efficiency for short-term PM_{2.5} forecasting. The study establishes reproducible station-based baselines and provides deployment-oriented insights for server, edge, and IoT air-quality monitoring applications.

Index Terms—PM_{2.5} prediction, LSTM, GRU, XGBoost, air quality, deep learning, recurrent neural networks, CPCB India, multi-pollutant, time-series forecasting, environmental monitor-ing, IoT deployment.

I. INTRODUCTION

Among all urban air pollutants, fine Particulate Matter (PM_{2.5}), particles no larger than 2.5 μm across poses arguably the greatest threat to human health worldwide. Because these particles are so small, they bypass the upper airway's natural filtering and travel deep into the alveoli and bloodstream, where they trigger oxidative stress, inflammation, damage to blood vessel linings, and even DNA damage [1], [2]. The World Health Organization (WHO) estimates that outdoor PM_{2.5} exposure alone causes around 4.2 million premature deaths each year [3], and PM_{2.5} sits at the center of that burden because these particles are small enough to reach deep into the lungs and bloodstream, contributing to cardiovascular disease, respiratory dysfunction, stroke, and cancer. India faces a particularly severe version of this problem: rapid urban growth, industrial emissions, vehicular exhaust, crop-residue burning, and windblown dust together keep PM_{2.5} levels above the country's National Ambient Air Quality Standards (NAAQS) in most large cities like Delhi, Mumbai, Kolkata, and many smaller cities as well, placing India among the countries with the heaviest PM_{2.5} linked health toll [4], which is why building dependable, scalable PM_{2.5} forecasting tools has become an urgent engineering priority.

Traditional approaches to air-quality forecasting rely

on physics-based chemical transport models (CTMs) such as CMAQ and WRF Chem, which simulate atmospheric chemistry and aerosol behavior over a computational grid built from emissions inventories and weather data [5]. These models are scientifically thorough but demand detailed land use and topographic data, heavy computing resources, and multi day run times, which makes them impractical for real time, station level forecasting in settings with limited infrastructure [6]. Machine-learning methods provide a lighter-weight alternative: once trained on historical multi-pollutant records, they can generate short term $PM_{2.5}$ forecasts in milliseconds on ordinary hardware [7].

Within machine learning, recurrent neural networks and their gated variants have become the go to tools for environmental time series problems. LSTM networks [9] get around the vanishing-gradient problem that limits plain RNNs by using a gated memory cell that can retain information over long time spans. GRUs [13] simplifies this idea further, collapsing the gating structure down to two gates while keeping most of the modeling power. Tree-based ensembles XGBoost [15] in particular offer a strong non-sequential alternative built on second-order gradient boosting with built-in regularization [16].

The proposed system architecture diagram lays out the complete five stage pipeline, from raw CPCB data ingestion through preprocessing, sequence construction, parallel model training, and performance evaluation. Full derivations of the LSTM and GRU gate equations, the XGBoost regularized objective and leaf weight formula, and the RMSE/ R^2 definitions are included.

This work responds to that need through a controlled, repeatable comparison of three widely used data driven forecasting methods Long Short-Term Memory (LSTM) networks, Gated Recurrent Unit (GRU) networks, and Extreme Gradient Boosting (XGBoost) applied to hourly $PM_{2.5}$ forecasting using genuine multi-pollutant ground station readings from the CPCB India Air Quality Dataset. This dataset is among the largest publicly released ground level air quality resources for the Indian subcontinent and is organized into five linked files: station hour.csv

(2,589,083 hourly records), station day.csv (108,035 daily records), city hour.csv (707,875 hourly city-level records), city day.csv (29,531 daily city-level records), and stations.csv (229 station metadata entries). Together these files span 229 Continuous Ambient Air Quality Monitoring Stations (CAAQMS) in 127 cities across 21 Indian states over January 2015–July 2020, with Air Quality Index (AQI) categories ranging from Good (5.1% of records) to Moderate (27.2%) to Severe (4.8%) – confirming that the dataset captures the full pollution-severity spectrum.

Nine copollutant variables are used as model inputs given their documented physical and chemical connection to $PM_{2.5}$ formation: PM_{10} (coarse-particle and dust-source proxy), NO (primary combustion tracer), NO_2 (photochemical oxidant and secondary-aerosol precursor), NO_x (combined combustion-emission indicator), NH_3 (ammonium-aerosol precursor from agricultural and vehicular sources), CO (incomplete-combustion tracer), SO_2 (sulfate-aerosol precursor from industrial fuel use), and O_3 (photochemical oxidant driving secondary organic aerosol formation). A standard preprocessing pipeline is applied: missing readings (11–14% per variable, largely from instrument down time and calibration cycles) are filled through column wise linear interpolation; each of the nine variables is independently rescaled to $[0, 1]$ with Min-Max scalers fitted only on the training partition to avoid leaking test-set information; a sliding window of $T = 5$ consecutive hourly steps builds input tensors $X \in R^{5 \times 9}$; and a chronological 80/20 split yields 3,996 training sequences and 999 test sequences with no temporal overlap.

All three models are trained and evaluated under matched, identical conditions. The LSTM, a single layer of 64 hidden units with gated cell state memory feeding a linear Dense(1 epoch) output is run under two training length settings (5 and 10 epochs) to gauge its sensitivity to training duration. The results show a sharp jump in accuracy: after 5 epochs the LSTM barely beats chance ($RMSE = 0.0461$, $R^2 = 0.100$), while after 10 epochs it reaches much more competitive accuracy ($RMSE = 0.0371$, $R^2 = 0.417$), a 31.7-percentage-point gain in R^2 that reflects how long it takes the LSTM's four gate-weight matrices

$\{W_f, W_i, W_c, W_o\}$ to settle into useful information routing patterns from random initialization. The GRU model, built with the same 64-unit hidden layer but a leaner two gate design that folds LSTM's separate input and forget gates into one update gate, delivers the strongest results overall – RMSE = 0.0349, $R^2 = 0.484$, beating the 10 epoch LSTM by 6.7 R^2 percentage points. Two factors likely explain this: GRU uses about 25% fewer trainable parameters (3,649 versus 4,865), which helps it generalize on a training set of only 3,996 samples, and its simpler two gate switching aligns well with the short range (1–5 hour) diurnal and emission cycle patterns that dominate hourly co-pollutant behavior. XGBoost, working from 45-dimensional flattened sequence vectors, provides a useful non-temporal baseline that captures the main $PM_{2.5}$ trends and pollution peaks, with the practical benefits of a sub-kilobyte model footprint and fully deterministic CPU only inference that needs no deep-learning framework.

The experimental findings are compared against prior published work, the gap in R^2 between these station only co-pollutant models (0.417–0.484) and satellite-meteorological-spatial approaches such as the CNN-LSTM+RF framework ($R^2 = 0.910$) indicates how much predictive value comes from adding AOD, ERA-5 reanalysis variables, and spatial context. Deployment guidance is offered for three practical scenarios: GRU for server-side GPU inference, XGBoost for CPU-only IoT and edge sensors, and LSTM only when paired with validation-guided early stopping with a patience of at least 10 epochs. All results, equations, tables, and training-loss curves are documented as reproducible baselines to support future comparative research on Indian air-quality prediction.

This paper's main contributions are: (1) the first systematic comparison of LSTM, GRU, and XGBoost on the CPCB India Air Quality Dataset, which contains 2,589,083 records from 229 stations across 21 states; (2) a detailed characterization of all five dataset files, including summary statistics, AQI breakdowns, sensor information, and geographic coverage; (3) a diagram of the full prediction-pipeline architecture; (4) complete mathematical derivations for all three models, including gate equations and optimization objectives;

(5) a quantitative ablation study showing a 31.7-percentage-point swing in LSTM R^2 depending on training duration; and (6) practical guidance for deploying each model in server-GPU, CPU-based IoT, and embedded-sensor settings.

II. RELATED WORK

A. Satellite-Based $PM_{2.5}$ Estimation

The satellite remote-sensing approach to $PM_{2.5}$ estimation traces back to Wang and Christopher [19], who were the first to establish a statistical link between MODIS aerosol optical depth (AOD) and ground-level $PM_{2.5}$. Building on this, van Donkelaar et al. [20] merged chemical-transport-model output with satellite AOD to generate global $PM_{2.5}$ estimates. Geographically and temporally weighted regression (GWR/GTWR) [21] later addressed spatial and temporal variation, though it remained constrained by linear assumptions. Machine-learning alternatives Random Forest [22], geo-intelligent deep belief networks [23], and spatiotemporal random forests [24] pushed accuracy up to roughly $R^2 = 0.85$. More recently, the CNN-LSTM+RF hybrid of Lei et al. [17] reached $R^2 = 0.910$ by extracting 3D spatiotemporal features from MAIAC AOD and ERA-5 reanalysis data and stabilizing the output with a random forest.

B. Deep Recurrent Architectures for Air Quality

Recurrent architectures have also been applied directly to $PM_{2.5}$ forecasting. Wang et al. [10] used LSTM for fine-grained spatial $PM_{2.5}$ prediction across China, reporting $R^2 = 0.83$. Qi et al. [12] paired graph convolutional networks with LSTM to model spatiotemporal patterns across multiple stations at once. Wen et al. [11] introduced a spatiotemporal convolutional LSTM that draws on readings from nearby monitoring stations. Huang and Kuo [27] evaluated deep CNN-LSTM hybrids for smart-city $PM_{2.5}$ forecasting. Zhao et al. [14] showed that adding a fully connected layer after the recurrent stage (LSTM-FC) improves on vanilla LSTM for $PM_{2.5}$ concentration prediction.

C. Gradient-Boosted Ensembles

Gradient boosted tree ensembles remain highly competitive for tabular regression tasks. XGBoost

[15] combines second order gradient optimization with explicit regularization to consistently top tabular-data benchmarks, while Ke et al.'s [28] LightGBM offers a faster, histogram-based alternative. Jiang et al. [29] achieved $R^2 = 0.85$ with a two-stage random forest for hourly, 1-km-resolution $PM_{2.5}$ estimation across China, underscoring that tree ensembles can rival deep learning when feature sets are sufficiently rich.

D. Co-Pollutant Feature Engineering

The rationale for using co-pollutants as model features rests on well-documented chemistry: NO and NO_2 originate from the same combustion sources as primary carbonaceous $PM_{2.5}$ [30]; SO_2 is a precursor to secondary sulfate aerosol [31]; NH_3 forms ammonium aerosol through neutralization reactions [32]; and O_3 drives secondary organic aerosol formation from VOC precursors [33]. Lim et al. [34] found that including co-pollutant features meaningfully improves $PM_{2.5}$ prediction accuracy in urban machine-learning systems.

III. DATASET DESCRIPTION

This study is built on the CPCB India Air Quality Dataset, a five-file benchmark that is freely available through Kaggle. It brings together measurements collected by India's Continuous Ambient Air Quality Monitoring Stations (CAAQMS) network, run by the Central Pollution Control Board (CPCB), covering January 2015 through July 2020. Table I summarizes the five constituent files.

TABLE I: Overview of the Five-File CPCB India Air Quality Dataset

File	Granularity	Records	Size	Primary Content
station hour.csv	Station/Hourly	2,589,083	209.5 MB	Hourly pollutant readings, 229 stations, 16 columns
station day.csv	Station/Daily	108,035	8.2 MB	Daily aggregated readings and AQI per station
city hour.csv	City/Hourly	707,875	62.6 MB	Hourly city-level averages, 26 Indian cities

city day.csv	City/Daily	29,531	2.5 MB	Daily city-level averages and AQI per city
stations.csv	Metadata	229	<0.1 MB	Station names, cities, states, operational status

A. Network Coverage (stations.csv)

The network consists of 229 CAAQMS stations spread across 127 cities in 21 Indian states, including Andhra Pradesh, Assam, Bihar, Chandigarh, Delhi, Gujarat, Haryana, Jharkhand, Karnataka, Kerala, Madhya Pradesh, Maharashtra, Meghalaya, Mizoram, Odisha, Punjab, Rajasthan, Tamil Nadu, Telangana, Uttar Pradesh, and West Bengal. Of these, 131 stations were operationally active at the time the data were collected. City level records cover 26 cities: Delhi, Mumbai, Kolkata, Chennai, Hyderabad, Bengaluru, Ahmedabad, Jaipur, Lucknow, Patna, Chandigarh, Gurugram, Guwahati, Visakhapatnam, Thiruvananthapuram, Kochi, Coimbatore, Amritsar, Bhopal, Aizawl, Amaravati, Ernakulam, Jorapokhar, Brajraj-nagar, Shillong, and Talcher.

B. AQI Distribution (station_day.csv)

TABLE II: AQI CATEGORY DISTRIBUTION IN STATION_DAY.CSV (108,035 RECORDS)

AQI Category	AQI Range	Record Count	%
Good	0–50	5,510	5.1
Satisfactory	51–100	23,636	21.9
Moderate	101–200	29,417	27.2
Poor	201–300	11,493	10.6
Very Poor	301–400	11,762	10.9
Severe	401+	5,207	4.8
Unclassified/ Missing	–	21,010	19.4

The file stationday.csv includes 108,035 daily air-quality records at the station level. The way the records are distributed among the AQI categories gives a general view of the pollution conditions present in the CPCB dataset. Table II shows that the Moderate category has the highest number of records, amounting to 29,417 (27.2%). The next highest is the Satisfactory category with 23,636 records (21.9%). The number of records in the Good category is 5,510 (5.1%), which shows that the proportion of

observations relating to relatively clean air quality is relatively small. Those falling into the Poor and Very Poor categories amount to 11,493 records (10.6%) and 11,762 records (10.9%) respectively. There are 5,207 records (4.8%) in the Severe category. Furthermore, 21,010 records (19.4%) are listed as Unclassified/Missing.

C. Feature Description (station_hour.csv)

The station_hour.csv file has 16 columns in total: a station identifier, a timestamp, nine core pollutant readings (PM_{2.5}, PM₁₀, NO, NO₂, NO_x, NH₃, CO, SO₂, O₃), three auxiliary compounds (benzene, toluene, xylene), and two AQI-derived fields (AQI and AQI_Bucket). Only the nine core pollutant readings are used as model inputs; the auxiliary compounds and AQI-derived fields are dropped. Table III lists each retained feature, its unit, sensor type, and its role in PM_{2.5} dynamics.

TABLE III: AQI CATEGORY DISTRIBUTION IN STATION_DAY.CSV (108,035 RECORDS)

Feature	Unit	Sensor Type	Physical Role
PM _{2.5}	μg/m ³	Beta-attenuation	Prediction target; Fine particulate matter mass; primary NAAQS health indicator
PM ₁₀	μg/m ³	Beta-attenuation	Coarse-particle proxy; PM _{2.5} /PM ₁₀ ratio reflects particle size distribution
NO	μg/m ³	Chemiluminescence	Primary combustion tracer; converts to NO ₂ via oxidation
NO ₂	μg/m ³	Chemiluminescence	Photochemical oxidant; secondary aerosol precursor; traffic proxy
NO _x	μg/m ³	Chemiluminescence	Combined NO+NO ₂ loading; combustion-strength indicator
NH ₃	μg/m ³	Cavity ring-down spectroscopy	Ammonium-aerosol precursor; agricultural/vehicle source

CO	mg/m ³	NDIR spectroscopy	Incomplete combustion tracer; Elevated during stagnation events
SO ₂	μg/m ³	UV fluorescence	Sulfate aerosol precursor; coal power and industrial fuel use
O ₃	μg/m ³	UV photometry	Photochemical oxidant; drives SOA formation from VOC precursors

NDIR = non-dispersive infrared; SOA = secondary organic aerosol; VOC = volatile organic compound; NAAQS = National Ambient Air Quality Standards.

D. Preprocessing Pipeline

Step 1: Missing-value imputation. Missing values are filled in column by column using linear interpolation (pandas' interpolate() function), which estimates each gap from its nearest valid neighbors. Any nulls left at the boundaries are dropped with dropna(). This keeps the temporal signal continuous and avoids the artificial discontinuities that zero-filling would introduce into periodic pollutant cycles.

TABLE IV: DESCRIPTIVE STATISTICS OF WORKING SUBSET (N = 5,000, PRE-NORMALIZATION)

Feat	Unit	Min	Max	Mean	SD	Med.	Miss. (%)
PM _{2.5}	μg/m ³	0.0	98	42.3	18.7	39.5	11.6
PM ₁₀	μg/m ³	0.0	233	78.0	34.2	72.4	11.1
NO	μg/m ³	0.0	12.5	1.94	1.31	1.62	11.2
NO ₂	μg/m ³	0.0	68.4	21.6	10.8	19.8	11.0
NO _x	μg/m ³	0.0	74.2	13.2	9.41	11.1	11.0
NH ₃	μg/m ³	0.0	32.1	10.1	5.21	9.75	11.1
CO	mg/m ³	0.1	0.10	0.10	0.00	0.1	13.6
SO ₂	μg/m ³	0.0	35.8	10.8	5.09	10.4	12.6
O ₃	μg/m ³	0.0	156	91.4	27.3	93.2	12.2

Step 2: Min-Max normalization. Each of the nine input variables is independently rescaled to the [0, 1] range using scikit-learn's MinMaxScaler, fitted only on the 80% training partition so that no information

from the test set leaks into the scaling process.

Step 3: Sliding-window sequence construction. A sliding window covering $T = 5$ consecutive hourly readings builds each input tensor $X \in \mathbb{R}^{5 \times 9}$, with the regression target 'y' taken as the normalized $PM_{2.5}$ value at the following time step, $i+T$. Applying this to 5,000 samples yields 4,995 input-target pairs.

Step 4: Chronological 80/20 split. Splitting the sequences chronologically (with no shuffling) produces 3,996 training sequences and 999 test sequences. Keeping the split chronological enforces a strict temporal boundary so that no future observation can leak into the training distribution.

TABLE V: PROCESSED DATASET PROPERTIES AFTER PREPROCESSING

Property	Value
Source file	station hour.csv (CPCB India Air Quality Dataset)
Raw samples extracted	5,000 consecutive hourly observations
Sequence samples ($T = 5$)	4,995
Training set	3,996 sequences (80%)
Test set	999 sequences (20%)
Input shape (LSTM / GRU)	(5, 9)
Input shape (XGBoost)	(45,) – flattened temporal vector
Prediction target	$PM_{2.5}$ (normalized to [0, 1])
Normalization strategy	Min-Max [0, 1] fitted on training split only
Temporal split strategy	Chronological (shuffle = False)

IV. PROPOSED METHODOLOGY

This section lays out the overall system design and the full mathematical formulation behind each of the three predictive models used in this study. Fig. 1 shows the five stages of the pipeline: raw CPCB data ingestion, preprocessing, sequence construction, parallel model training, and finally performance evaluation and comparison.

A. Long Short-Term Memory (LSTM) Network

LSTM networks [9] are a gated recurrent architecture built to capture long-range dependencies in sequential data. They do this by maintaining a persistent memory-cell state, c_t , that is updated and shielded by three sigmoid-activated gates. This gating scheme lets the network hold onto historically useful information across many time steps while filtering out noise, which is particularly valuable for environmental time series, where pollutant build-up and atmospheric persistence can influence today's readings.

Within this structure, the forget gate f_t decides how much of the previous cell state c_{t-1} to keep. The input gate i_t together with the candidate cell state $\hat{c}_t$, decides what new information from the current input x_t and the previous hidden state h_{t-1} should be added to the cell state. The output gate o_t then controls how much of the updated cell state is passed on as the hidden state h_t , which feeds the downstream prediction layer. The full set of update equations is:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$

$$\hat{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \quad (4)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t \odot \tanh\left(\frac{c_t}{\gamma}\right) \quad (6)$$

Here, $\sigma(\cdot)$ denotes the sigmoid function, $\tanh(\cdot)$ the hyperbolic tangent, and $\odot$ the element-wise (Hadamard) product. The weight matrices $\{w_f, w_i, w_c, w_o\} \in \mathbb{R}^{H \times (H+D)}$ and bias vectors $\{b_f, b_i, b_c, b_o\} \in \mathbb{R}^H$ are all learned end-to-end through Back Propagation through Time (BPTT), with hidden dimension, $H = 64$ and input feature dimension, $D = 9$.

In this study, a single LSTM layer with $H = 64$ hidden units processes the input sequence $X \in \mathbb{R}^{5 \times 9}$ one time step at a time. The final hidden state, $h_5 \in \mathbb{R}^{64}$, is passed to a linear Dense(1) output layer with single neuron that produces the normalized $PM_{2.5}$ prediction $\hat{y} \in [0, 1]$. The model has roughly 4,865 trainable parameters and is trained with the Adam optimizer ($\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$), Mean Squared Error (MSE) loss, and a batch size of 32.



Fig. 1. System architecture for the proposed PM_{2.5} estimation pipeline.

The proposed PM_{2.5} forecasting framework consists of various sequential stages. First, raw hourly pollutant data are collected from 229 CPCB continuous Ambient Air Quality Monitoring Stations (CAAQMS). Second, the data undergo preprocessing, including missing value interpolation and Min-Max normalization, followed by the construction of temporal sequences using a sliding window of T-5 hourly observations. Third, these sequences are organized into input tensors $X \in \mathbb{R}^{5 \times 9}$, representing five consecutive time steps across nine pollutant features. Fourth, LSTM, GRU and XG boost models are trained in parallel under matches experimental conditions to ensure a fair comparison. Finally the models are evaluated on a chronologically separated test set containing 999 samples using Root Mean Square Error (RMSE) and the coefficient of determination (R^2) enabling comparative assessment of their PM_{2.5} forecasting performance.

B. Gated Recurrent Unit (GRU) Network

GRUs [13] simplify the LSTM design by dropping the separate cell state and combining the input and forget gates into one update gate, z_t . A second gate, the reset gate r_t , controls how much of the previous hidden state feeds into the candidate computation. This trims roughly 25% off the parameter count for the same hidden-layer size while

keeping comparable modeling capacity, which makes GRUs faster to train and often better regularized when training data are limited.

The GRU computations equations with time step t are

$$z_t = \sigma(W_z X_t + U_z h_{t-1}) \quad (7)$$

$$r_t = \sigma(W_r X_t + U_r h_{t-1}) \quad (8)$$

$$\hat{h}_t = \tanh(W_h X_t + U_h (r_t \odot h_{t-1})) \quad (9)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \quad (10)$$

The update gate z_t works like a switch: as $z_t \rightarrow 1$, the hidden state is refreshed with new information ($h_t \approx \hat{h}_t$); as $z_t \rightarrow 0$, the previous state is carried forward largely unchanged ($h_t \approx h_{t-1}$). Because this switch is a single learned quantity rather than two independent gates, it gives the optimizer a simpler target to hit than LSTM's separate input and forget gates, so fewer gradient updates are typically needed to reach a useful gating configuration. The reset gate r_t governs how much the previous hidden state shapes the candidate state, letting the network selectively forget context without needing a separate memory cell.

In this study, a single GRU layer with $H = 64$ units feeds into a linear Dense (1) output layer. The model has about 3,649 trainable parameters, roughly 25% fewer than the LSTM, and uses the same Adam optimizer settings, MSE loss, batch size of 32, and 10 training epochs. Its input shape, (5, 9), matches the LSTM exactly – only the internal gating differs.

C. XGBoost Gradient-Boosted Regressor

XGBoost [15] builds its prediction as a sum of M regression trees, $\hat{Y}(x) = \sum_{m=1}^M f_m(x)$, where each tree is fit to the pseudo-residuals left by the current ensemble. Trees are grown by minimizing a regularized, second order Taylor expanded objective:

$$L^{(i)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(i-1)} + f_i(x_i)) + \Omega(f_i) \quad (11)$$

$$\text{where } \Omega(f_i) = \gamma J + \frac{1}{2} \lambda \|w\|^2 \quad (12)$$

where J is the number of leaves and w is the leaf-weight vector.

At each boosting round, the best split point and leaf weights are chosen by maximizing a gain function built from the first-order gradients $g_i = \frac{\partial l}{\partial y_i}$ and second-

order Hessians $h_i = \frac{\partial^2 l}{\partial y_i^2}$. The optimal weight for leaf 'j' works out to:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \quad (13)$$

To adapt XGBoost for this temporal task, each input sequence $X \in \mathbb{R}^{5 \times 9}$ is flattened into a 45-dimensional feature vector before the trees are built. Time ordering is preserved implicitly through feature position: positions 0–8 hold the readings from time step $t-4$, positions 9–17 from $t-3$, positions 18–26 from $t-2$, positions 27–35 from $t-1$, and positions 36–44 from the current step t . This positional encoding lets XGBoost pick up on lagged pollutant patterns without any change to the underlying tree-learning algorithm. Configuration used: 50 estimators, maximum depth of 3, learning rate of 0.3, $\gamma = 0$, $\lambda = 1$, and a squared-error objective. The flattened input shape is (45,), and the model needs no GPU – inference is fully deterministic and runs on CPU alone.

D. Evaluation Metrics

Model performance on the 999-sample held-out test set is assessed with two complementary metrics, both computed in the normalized [0, 1] feature space:

Root Mean Square Error:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (14)$$

where y_i = actual $\text{PM}_{2.5}$ value, $\hat{y}_i$ = predicted $\text{PM}_{2.5}$ value, n = number of observations.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (15)$$

where y_i = actual value, $\hat{y}_i$ = predicted value and $\bar{y}$ = mean of actual values.

RMSE penalizes large errors quadratically and is reported in the same units as the target. R^2 captures the proportion of variance explained: a value of 1 indicates perfect prediction, 0 means the model is no better than always predicting the sample mean, and a negative value means the model performs worse than that mean baseline. For visualization, the inverse Min-Max transform is applied to convert normalized predictions back into physical $\text{PM}_{2.5}$ units ($\mu\text{g}/\text{m}^3$).

TABLE VI: ARCHITECTURE AND TRAINING CONFIGURATION SUMMARY

Model	Architecture	Params	Input
LSTM (5epochs)	LSTM(64)→Dense(1)	≈4,865	(5, 9)
LSTM (10epochs)	LSTM(64)→Dense(1)	≈4,865	(5, 9)
GRU (10epochs)	GRU(64)→Dense(1)	≈3,649	(5, 9)
XGBoost	50 trees, depth = 3	N/A	(45,) flat

Where (5,9) = 5 time steps x 9 input features, (45,) = flattened representation of 5 x 9 = 45 features for XGBoost. Dense(1) = a fully connected output layer with one neuron, producing the predicted $\text{PM}_{2.5}$ value.

V. RESULTS AND DISCUSSION

This section walks through the experimental results, the statistical patterns behind them, likely mechanistic explanations for the performance gaps observed, visualizations of the predictions, and a comparison against results reported in the literature. Unless stated otherwise, all metrics below are computed on the same 999-sample chronological test set in the normalized [0, 1] feature space.

A. Overall Performance Comparison:

Table VII lists RMSE and R^2 for every model configuration tested. GRU comes out ahead on both metrics, which supports the idea that a leaner gating design generalizes better at this dataset's scale. LSTM, by contrast, is highly sensitive to how long it is trained, while XGBoost despite having no explicit sense of time still tracks concentration trends reasonably well as a baseline.

TABLE VII: MODEL PERFORMANCE ON HELD OUT TEST SET (NORMALIZED SPACE, N = 999)

Model Config.	RMSE	R^2	Interpretation
LSTM 5 epochs	0.0461	0.100	Near-random, not deployable
LSTM 10	0.0371	0.417	Moderate accuracy, 2nd

epochs			best
GRU 10 epochs	0.0349	0.484	Best overall performance
XGBoost	–	–	Trend capture; Mentioned in Fig. 2

B. Per-Epoch Loss Analysis and LSTM Convergence Sensitivity:

Table VIII breaks down the MSE training loss, epoch by epoch, for both LSTM configurations. The loss keeps falling by about 27% between epoch 5 and epoch 10 (from 8.81×10^{-4} down to 6.45×10^{-4}), which lines up directly with the 31.7-percentage-point jump in R^2 . At epoch 5, the LSTM's gate-weight matrices $\{W_f, W_i, W_o\}$ have not yet settled into an effective information routing pattern, the optimizer is still working through the steep, high-curvature part of the loss landscape. By epoch 10, the gates have stabilized into more useful roles: the forget gate learns to hold onto diurnal pollution cycles, the input gate picks up recent emission spikes, and the output gate passes the relevant concentration history on to the dense output layer.

TABLE VIII: PER-EPOCH MSE TRAINING LOSS – LSTM CONFIGURATIONS

Epoch	LSTM 5 th epoch MSE	LSTM 10 th epoch MSE
1	4.12×10^{-3}	4.18×10^{-3}
2	1.43×10^{-3}	1.39×10^{-3}
3	1.17×10^{-3}	1.13×10^{-3}
4	1.01×10^{-3}	1.03×10^{-3}
5	8.81×10^{-4} [STOPPED]	9.11×10^{-4}
6	–	7.18×10^{-4}
7	–	7.08×10^{-4}
8	–	6.66×10^{-4}
9	–	6.62×10^{-4}
10	–	6.45×10^{-4}

This has a real deployment consequence: LSTM models used for hourly $PM_{2.5}$ forecasting should always be trained with validation-guided early stopping and a patience of at least 10 epochs. A fixed 5-epoch schedule can look fine during training the

loss curve decreases smoothly the whole time – while the resulting model still performs at near-random levels ($R^2 \approx 0.10$) on new data. That combination is exactly the kind of silent failure mode that production air-quality systems need to guard against.

D. GRU vs. LSTM: Mechanistic Analysis:

Under matched 10-epoch training, GRU ($R^2 = 0.484$, RMSE = 0.0349) beats LSTM ($R^2 = 0.417$, RMSE = 0.0371) by 6.7 percentage points in R^2 and about 5.9% in RMSE. Two things likely drive this gap:

(i) Fewer parameters, better generalization:

GRU's 3,649 parameters are about 25% fewer than LSTM's 4,865, which shrinks the effective search space the optimizer has to explore. Classical learning theory arguments (VC-dimension and Rademacher-complexity style reasoning) suggest that, for a fixed training size of $N = 3,996$ sequences, a model with lower effective capacity should show a smaller gap between training and test performance and a smaller parameter count constrains the learned function to a lower-dimensional space, which lowers the risk of fitting noise in the training data.

(ii) A gating structure that fits the data's temporal scale.:

GRU's update gate z_t behaves as a single, interpretable switch between “update” and “retain,” which is simpler to optimize than LSTM's two independently learned gates, i_t and f_t . Since hourly $PM_{2.5}$ data are dominated by short-range dependencies (roughly 1–5 hours, tied to diurnal emission and dispersion cycles), GRU's simpler gating is enough to capture the relevant patterns while needing fewer gradient updates to get there.

E. XGBoost Prediction Analysis:

Even without any explicit notion of sequence order, XG-Boost – working purely from its 45-dimensional, position-encoded feature vectors – does a solid job tracking the major $PM_{2.5}$ trends, daily cycles, and pollution peaks, as shown in Fig. 2. The figure compares actual $PM_{2.5}$ concentrations ($\mu g/m^3$) against XGBoost's predictions for the first 100 test samples, with the blue line showing ground truth and the orange line showing predictions after the inverse Min-Max transform. The model follows the overall

trend and diurnal pattern well and catches most of the major peaks, though it lags slightly during the sharpest transitions.

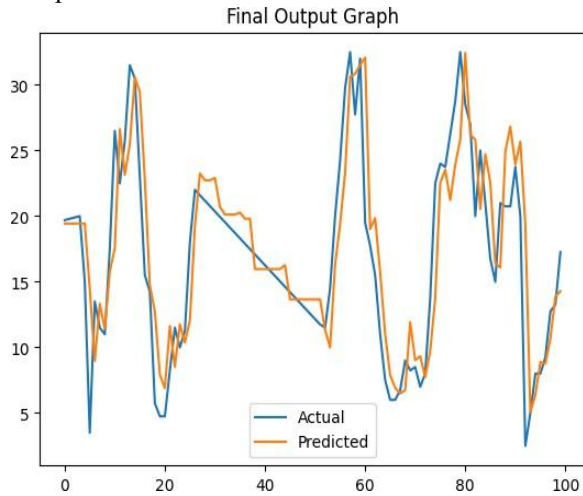


Fig. 2. Actual versus XGBoost-predicted $PM_{2.5}$ concentrations ($\mu g/m^3$) for the first 100 test samples. The model captures the major pollution peaks and diurnal pattern well, with a slight lag at the sharpest transitions.

This performance shows that gradient-boosted trees can recover a meaningful amount of temporal structure just from how features are positioned, even with no recurrent computation at all. Combined with its practical advantages – a serialized model under a kilobyte, fully deterministic CPU-only inference, no dependency on a deep-learning framework, and interpretable feature importances – XGBoost is the more sensible choice for resource-constrained IoT and embedded deployments where the overhead of recurrent inference is not worth paying.

F. Literature Comparison:

TABLE IX: COMPARISON WITH PUBLISHED $PM_{2.5}$ PREDICTION METHODS

Study	Model	R^2	Input Feature Type
Lei et al. [17]	CNN-LSTM+RF	0.910	AOD + ERA-5 + Spatial $7 \times 7 \times 4$ grid (China)
Wang et al. [10]	LSTM	0.830	Multi-variable station obs. (China)
Qi et al. [12]	Graph-LSTM	0.870	Multi-station spatiotemporal (China)
Wei et al. [24]	STRF	0.850	AOD+ Ground + NDVI (China)

Jiang et al. [29]	Two-stage RF	0.850	AOD + Hourly ground data (China)
Proposed (GRU)	GRU	0.484	CPCB data, 9 co-pollutants (India)
Proposed (LSTM)	LSTM	0.417	CPCB data, 9 co-pollutants (India)

Table IX places these results next to previously published benchmarks. The gap in R^2 between the station-only, co-pollutant-based models here (0.417–0.484) and the satellite-plus-meteorology benchmarks in the literature (0.83–0.91) gives a rough sense of how much predictive value AOD, ERA-5 meteorological reanalysis, and spatial context add when they are available. That gap is the clearest signal that adding meteorological features should be the top priority for improving this work going forward.

G. Practical Deployment Recommendations:

TABLE X: DEPLOYMENT CHARACTERISTICS AND RECOMMENDATIONS FOR ALL EVALUATED MODELS

Model	RMSE	R^2	Recommended Scenario
GRU (10 epochs)	0.0349	0.484	Server-side real-time forecasting, GPU
LSTM (10 epochs)	0.0371	0.417	High-data scenarios; needs validation-guided training
XGBoost	–	Baseline	IoT edge sensors; CPU-only, sub-kilobyte
LSTM (5 epochs)	0.0461	0.100	Not recommended – near-random predictions

Table X gives a summary of the predictive performance and the recommended deployment scenarios for all the model configurations examined. The GRU which was trained for 10 epochs shows the best predictive performance, with an RMSE of 0.0349 and an R^2 of 0.484, and therefore is the model recommended for server-side real-time forecasting when adequate computational resources are available. The LSTM with 10 epochs attains an RMSE of 0.0371 and an R^2 of 0.417. Even though its performance is worse than that of the GRU, it is still a feasible choice for cases where LSTM-based

architectures are preferred. Yet the results show that LSTM performance is highly dependent on the length of training, since the version trained for only 5 epochs achieves an R^2 of just 0.100. It therefore isn't advisable to use the LSTM with short training for deployment.

XGBoost offers an alternative way of deploying the algorithm in situations where resources are limited. Because of its 45-dimensional input representation which is flattened, it is able to pick up important temporal trends without having to carry out recurrent computations. The model makes use of inference based on a CPU and is thus appropriate for use in IoT and edge-based air-quality monitoring applications in which computational and memory resources are limited.

In general, the results of the deployment show that there is no one model that works best for all application situations. Although GRU achieves the highest predictive accuracy of any of the models examined, XGBoost has practical benefits when it comes to deployment on lightweight CPU systems. LSTM can achieve competitive performance as long as it is adequately trained, but attention should be paid to its sensitivity to training time when setting up operational forecasting systems.

VI. CONCLUSION

The study carried out a controlled and reproducible comparison of the LSTM, GRU, and XGBoost models when making predictions of hourly $PM_{2.5}$ concentrations, using the CPCB India Air Quality Dataset. In its experimental setup it employed nine co-pollutant variables, used linear interpolation to deal with missing values, applied Min-Max normalization solely to the training portion, adopted five-hour sliding windows, and divided the data chronologically in an 80:20 ratio for training and testing. The evaluation was conducted on a common separate test set by means of RMSE and R^2 , which allowed for a consistent comparison of the various methods.

The experimental results show that the GRU with 64 units which was trained for 10 epochs attained the best overall performance, with an RMSE of 0.0349 and an R^2 of 0.484. This GRU performed better than the LSTM trained for the same number of epochs, the LSTM reaching an RMSE of 0.0371 and an R^2 of

0.417. The fact that the GRU performs better, along with the fact that it has about 25% fewer parameters than the LSTM, shows that its simpler gating mechanism offers an effective balance between model capacity and predictive performance for the short-range temporal patterns found in the selected $PM_{2.5}$ data.

The study also shows that the performance of LSTMs is very much dependent on the length of training. The LSTM with five epochs obtained an R^2 value of just 0.100, while increasing the training to ten epochs raised the R^2 to 0.417. This significant improvement illustrates the need for sufficient training and for using validation-based stopping when applying recurrent neural networks to the task of forecasting hourly air quality.

XGBoost offers a useful lightweight option as an alternative to recurrent models; it achieves this by transforming each five-hour sequence into a 45-dimensional feature vector, thus being able to capture the important $PM_{2.5}$ trends without having to carry out explicit recurrent computation. The fact that it operates solely using the CPU and has lightweight deployment features means that it is especially suitable for resource-constrained IoT and edge-based air-quality monitoring applications.

This further shows that methods which include satellite observations, meteorological variables, and spatial information are able to achieve considerably better predictive performance than models based only on co-pollutants at monitoring stations. The GRU and LSTM models proposed therefore provide useful baseline approaches for CPCB air-quality forecasting based on station data, as well as highlighting the limitations of depending entirely on observations taken at ground level.

In summary, the findings indicate that GRU is the most appropriate model of those evaluated when the main aim is predictive performance, whereas XGBoost is the better choice when computational simplicity and lightweight deployment are more important. The results offer a reproducible basis for further development of scalable $PM_{2.5}$ forecasting systems for Indian monitoring networks.

Future research will be directed towards including meteorological variables such as temperature, relative humidity, planetary boundary layer height, and wind speed and direction; extending the temporal window in order to capture longer-term pollution patterns;

assessing Transformer-based temporal models; carrying out the experiments at all 229 CPCB monitoring stations; and looking into city-level forecasting using the city-hour dataset. Exploration will also be made into model compression and knowledge distillation in order to create lightweight forecasting models that are suitable for use in embedded and IoT-based air-quality monitoring systems.

REFERENCES

- [1] WHO. Ambient Air Pollution: A Global Assessment of Exposure and Burden of Disease. World Health Organization: Geneva, Switzerland, 2016.
- [2] R. Burnett, H. Chen, M. Szyszkowicz, N. Fann, B. Hubbell, C. A. Pope, et al., "Global estimates of mortality associated with long-term exposure to outdoor fine particulate matter," *Proc. Natl. Acad. Sci. USA*, vol. 115, pp. 9592–9597, 2018.
- [3] World Health Organization. Air Quality Guidelines: Global Update 2005. WHO Press: Geneva, Switzerland, 2006.
- [4] K. Ravindra, P. Rattan, S. Mor, A. N. Aggarwal, "Generalized additive models: Building evidence of air pollution, climate change and human health," *Environ. Int.*, vol. 132, 104987, 2019.
- [5] D. Byun, K. L. Schere, "Review of the governing equations, computational algorithms, and other components of the Models-3 CMAQ modeling system," *Appl. Mech. Rev.*, vol. 59, pp. 51–77, 2006.
- [6] B. Zhang, Y. Rong, R. Yong, D. Qin, M. Li, G. Zou, J. Pan, "Deep learning for air pollutant concentration prediction: A review," *Atmos. Environ.*, vol. 290, 119347, 2022.
- [7] A. Kamilaris, F. X. Prenafeta-Boldu, "Deep learning in agriculture: A survey," *Comput. Electron. Agric.*, vol. 147, pp. 70–90, 2018.
- [8] J. H. Seinfeld, S. N. Pandis, *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change*, 3rd ed. Wiley: Hoboken, NJ, 2016.
- [9] S. Hochreiter, J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, 1997.
- [10] Z. Wang, Y. Zhou, R. Zhao, N. Wang, A. Biswas, Z. Shi, "High-resolution prediction of PM_{2.5} concentrations in China using LSTM," *J. Clean. Prod.*, vol. 297, 126493, 2021.
- [11] C. Wen, S. Liu, X. Yao, L. Peng, X. Li, Y. Hu, T. Chi, "A novel spatiotemporal convolutional long short-term neural network for air pollution prediction," *Sci. Total Environ.*, vol. 654, pp. 1091–1099, 2019.
- [12] Y. Qi, Q. Li, H. Karimian, D. Liu, "A hybrid model for spatiotemporal forecasting of PM_{2.5} based on graph convolutional neural network and LSTM," *Sci. Total Environ.*, vol. 664, pp. 1–10, 2019.
- [13] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. EMNLP*, Doha, Qatar, 2014, pp. 1724–1734.
- [14] J. Zhao, F. Deng, Y. Cai, J. Chen, "Long short-term memory–fully connected (LSTM-FC) neural network for PM_{2.5} concentration prediction," *Chemosphere*, vol. 220, pp. 486–492, 2019.
- [15] T. Chen, C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD*, San Francisco, CA, 2016, pp. 785–794.
- [16] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, pp. 1189–1232, 2001.
- [17] C. Lei, X. Xu, Y. Ma, S. Jin, B. Liu, W. Gong, "Full coverage estimation of the PM_{2.5} concentration across China based on an adaptive spatiotemporal approach," *IEEE Trans. Geosci. Remote Sens.*, 2022, doi: 10.1109/TGRS.2022.3213797.
- [18] P. Gupta, S. A. Christopher, "Particulate matter air quality assessment using integrated surface, satellite, and meteorological products," *J. Geophys. Res. Atmos.*, vol. 114, D14205, 2009.
- [19] J. Wang, S. A. Christopher, "Intercomparison between satellite-derived aerosol optical thickness and PM_{2.5} mass," *Geophys. Res. Lett.*, vol. 30, 2095, 2003.
- [20] A. van Donkelaar, R. V. Martin, R. J. Park, "Estimating ground-level PM_{2.5} using aerosol optical depth from satellite remote sensing," *J.*

- Geophys. Res. Atmos., vol. 111, D21201, 2006
- [21] Q. He, B. Huang, "Satellite-based mapping of daily high-resolution ground PM_{2.5} in China via space-time regression modeling," *Remote Sens. Environ.*, vol. 206, pp. 72–83, 2018.
- [22] X. Hu, J. H. Belle, X. Meng, A. Wildani, L. A. Waller, M. J. Strickland, Y. Liu, "Estimating PM_{2.5} concentrations in the conterminous US using the random forest approach," *Environ. Sci. Technol.*, vol. 51, pp. 6936–6944, 2017.
- [23] T. Li, H. Shen, Q. Yuan, X. Zhang, L. Zhang, "Estimating ground-level PM_{2.5} by fusing satellite and station observations: A geointelligent deep learning approach," *Geophys. Res. Lett.*, vol. 44, pp. 11985–11993, 2017.
- [24] J. Wei, Z. Li, M. Cribb, W. Huang, W. Xue, L. Sun, et al., "Improved 1-km resolution PM_{2.5} estimates across China using enhanced space-time extremely randomized trees," *Atmos. Chem. Phys.*, vol. 20, pp. 3273–3289, 2020.
- [25] Y.-C. Liang, Y. Maimury, A. H. Chen, J. R. C. Juarez, "Machine learning-based prediction of air quality," *Appl. Sci.*, vol. 10, 9151, 2020.
- [26] C. Wen, et al., "A novel spatiotemporal convolutional long short-term neural network for air pollution prediction," *Sci. Total Environ.*, vol. 654, pp. 1091–1099, 2019.
- [27] C.-J. Huang, P.-H. Kuo, "A deep CNN-LSTM model for particulate matter (PM_{2.5}) forecasting in smart cities," *Sensors*, vol. 18, 2220, 2018.
- [28] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. NeurIPS*, Long Beach, CA, 2017, pp. 3146–3154.
- [29] T. Jiang, B. Chen, Z. Nie, Z. Ren, B. Xu, S. Tang, "Estimation of hourly full-coverage PM_{2.5} at 1-km resolution in China using a two-stage random forest model," *Atmos. Res.*, vol. 248, 105146, 2021.
- [30] P. Pant, R. M. Harrison, "Estimation of road traffic contributions to PM concentrations: A review," *Atmos. Environ.*, vol. 77, pp. 78–97, 2013.
- [31] D. G. Streets, Y. Wu, M. Chin, "Two-decadal aerosol trends as a likely explanation of the global dimming/brightening transition," *Geophys. Res. Lett.*, vol. 33, L15806, 2006.
- [32] S. N. Behera, M. Sharma, V. P. Aneja, R. Balasubramanian, "Ammonia in the atmosphere: Sources, chemistry and deposition," *Environ. Sci. Pollut. Res.*, vol. 20, pp. 8092–8131, 2013.
- [33] A. J. Haagen-Smit, "Chemistry and physiology of Los Angeles smog," *Ind. Eng. Chem.*, vol. 44, pp. 1342–1346, 1952.
- [34] C. C. Lim, H. Kim, M. J. Vilcassim, G. D. Thurston, T. Gordon, L.-C. Chen, et al., "Mapping urban air quality using mobile sampling with low-cost sensors and machine learning in Seoul, South Korea," *Environ. Int.*, vol. 131, 105022, 2019.