CIVIL & ENVIRONMENTAL ENGINEERING | RESEARCH ARTICLE

A comparative analysis of LSTM and ARIMA for enhanced real-time air pollutant levels forecasting using sensor fusion with ground station data

Geetha Mani¹* and Rohit Volety²

ABSTRACT: One of the most basic needs of any human being to survive is air. Unfortunately, this basic need is being polluted by many natural factors like volcanic eruptions, forest fires, and man-induced factors like transportation emission.

ABOUT THE AUTHOR

Dr. Geetha Mani received B.E., in Electronic and Instrumentation Engineering from Madurai Kamaraj University and M.E., in Instrumentation Engineering from MIT Campus, Anna University, India. She was awarded Doctor of Philosophy from Anna University, Chennai in the field of Advanced Process Control. She has been teaching for 14 years at reputed institutions. Presently, she is an Associate Professor in the School of Electrical Engineering at Vellore Institute of Technology, Vellore, India. She was a recipient of the IEI Young Engineers Award to recognize her contribution in the field of Electrical Engineering in 2016 from the Institution of Engineers, India. Her research interests include Process Control, Soft sensing, Industrial Automation, and the Internet of Things. She also authored and published several research articles in peer-reviewed international journals and book chapters. Her student, Rohit Volety, is currently pursuing third year of B.Tech in Electronics and Communication Engineering with a specialization in biomedical engineering at VIT University. Both of the authors have contributed to the domain of machine learning in terms of research. The key research activity worth mentioning is the work on the sensor fusion technology which led to this research paper. Apart from this, both the authors are involved in several research activities in the domain of machine learning. The research reported in this paper can be related to wider projects. The concept of sensor fusion can be used for other applications like temperature forecasting, rainfall forecasting etc. The comparison of performance of ARIMA and LSTM can also help other authors choose between these models for their own projects.

PUBLIC INTEREST STATEMENT

One of the most basic needs of any human being to survive is air. Unfortunately, this basic need is being polluted by many natural factors like volcanic eruptions, forest fires, and man-induced factors like transportation emission, mining operations, etc. All of these factors have deprived us and our future generations of the meaning of the word ‘Pure Air’. Unpolluted air is now an ideal environment that can never be achieved. This research paper aims to forecast the air pollutant levels by comparing futuristic machine learning models that are Long Short-Term Memory (LSTM) and Auto-Regressive Integrated Moving Average (ARIMA) performed on fused data obtained from ground station data collected from the CPCB (Central Pollution Control Board) and the data collected from a low-cost hardware setup of sensors. Furthermore, the concept of sensor fusion, which is the main technical innovation, and the novelty of the paper have been discussed. The LSTM model and ARIMA model have been used to forecast the air pollutant levels in the future.
Unpolluted air is now an ideal environment that can never be achieved. So, the pollution levels should be monitored continuously. However, monitoring the levels of pollution will not fix the environment. Forecasting these pollution levels can make the society more aware of the environment and help prepare safety measures. This research paper aims to forecast the air pollutant levels by comparing futuristic machine learning models, which are Long Short-Term Memory (LSTM) and Auto-Regressive Integrated Moving Average (ARIMA) performed on the ground station data from Central Pollution Control Board (CPCB), the data collected from a low-cost IoT hardware setup, and the fused data. The LSTM and ARIMA models have been used to forecast the air pollutant levels in the future. Further, the main novelty of this research is to show that the concept of sensor fusion increases the accuracy of the dataset. The outputs obtained after implementing LSTM and ARIMA models show more accurate results when compared with ground station data and the IoT data from sensors.

Subjects: Environmental Change & Pollution; Machine Learning - Design; Environmental

KEYWORDS: Air pollutant levels; Machine Learning; forecasting; ARIMA; LSTM; sensor fusion; IoT

1. Introduction

Air is one of the most important elements for all living life on earth. The quality of the air we breathe in determines the quality of the life we live and how long we live it. Several cases of ischemic heart disease, stroke, chronic obstructive pulmonary disease (COPD-progressive lung diseases), and acute lower respiratory infections have been reported in people and children, who breathe in polluted air. Apart from the effect on human beings, polluted air has also been shown to contribute to global warming (Sankar Ganesh et al., 2017). Air pollution is a major contribution to some of the deadliest diseases, which are being analyzed every day. Continuous exposure to air pollutants has affected the health of all human beings and has now quickly led to mortality and morbidity (World Health Organization, 2016). All these reasons make the monitoring of air pollution a very crucial task.

The main motivation behind this research work comes from the fact that air pollution can be easily accounted for as one of the biggest disasters in the 21st century. It has caused great harm to not just human beings but also the entire flora and fauna. Moreover, air pollution has always been a side product of the greatest inventions of human beings. Hence it is also our responsibility to rid the earth of this disaster. Air pollution has been on the radar of several stations for a very long time. Hence, a lot of data have been generated and stored, but most of these data come from stations that have been poorly maintained and hence are not reliable. With all of these data at our disposal, it is our objective to pick the best working machine-learning algorithm to help us forecast air pollution levels. However, this forecast is of no use if the data that we base our research on is not reliable, so the main novelty of this research work is that we introduced sensor fusion of the ground station data and the IoT data from the sensors to get more accurate data to which we can apply machine learning models.

In the literature, there has been a significant amount of progress in forecasting the level of air pollutants using several machine learning algorithms. The researchers in (D. Wang et al., 2017) proposed a hybrid CEEMD-VMD-DE-ELM model that depends on a two-stage decomposition method and ELM model upgraded by the DE calculation to boost the figure accuracy of everyday Air Quality Index (AQI) series. To boost the demonstrating precision and reasonability of displaying, a Hampel filter has been utilized in (Li et al., 2019) to diminish outliers, and a feature
A determination approach that depends on CEEMD and BCCSA was talked about to decide the best information variable structure. An altered LSSVM, which depends on the MOMVO calculation, has been discussed at the same time for acquiring high accuracy and solid stability. The authors in (Peng et al., 2017) showed us the potential of using nonlinear machine learning methods to improve air quality forecasts. An innovative hybrid-Garch model is proposed in (P. Wang et al., 2017) to get rid of the issue of heteroscedasticity. An ARIMA model is compared with an ARIMA-XGBoost hybrid model, and it is concluded that declaring the ARIMA-XGBoost hybrid models got better forecasting as they better accuracy (Li & Zhang, 2018). Moreover, VMD-SE-LSTM model is proposed to display a superior capacity for AQI forecasting by capturing the characteristics of the original AQI very comprehensively (Wu & Lin, 2019). The ARIMA model is utilized in (Kulkarni et al., 2018) to forecast the air pollution levels in Nanded city, Maharashtra, India. The academicians in (Ha et al., 2020) proposed the utilization of a Kalman filter with a fractional order to predict indoor air pollutants by taking into account the nonlinear and stochastic nature and hence improving accuracy.

This paper discusses the implementation of LSTM and ARIMA models and compares both these techniques to forecast the air pollutant level. Both the techniques are applied on the ground station dataset, the IoT dataset from sensors, and the fused dataset obtained after fusing the IoT and ground station data. The main contribution of this research is to show that the concept of sensor fusion increases the accuracy of the dataset. The outputs obtained after implementing LSTM and ARIMA models show more accurate results when compared with ground station data and the IoT data from sensors.

The research manuscript is divided into five sections. The first section walks us through the literature study and discusses the importance of the forecasting of air pollutant levels. The second section describes the methodology that was followed in this work. The major topics that were dealt with are time series analysis and their background information, performance indices used in time series analysis, a hardware description of the circuit that was made for collecting data, and background information of the sensor fusion. In the fourth section, experimental results have been shown in the form of actual versus predicted graphs and also by computing the RMSE. The results of sensor fusion and its effect on the forecasted values have also been discussed in this section. Finally, the paper is concluded indicating the main focus of work, results, and also the future work that can be done to make this research even more productive.

2. Proposed methodology
This section explains the proposed method as shown in Figure 1. In this work, ARIMA and LSTM models have been implemented on the data collected from the ground station, sensor output, and
the fused data. Furthermore, the sensor fusion technique has been claimed to be better for forecasting.

The whole methodology of our research paper can be divided into four parts:

(a) Time series analysis
(b) Performance indices
(c) Hardware description
(d) Sensor fusion

2.1. Time series analysis
A time series can be explained as a sequence of observations, which have been taken sequentially in time. We can see countless batches of data that appear as time series like daily rainfall amounts, heights of ocean tides, monitoring of a person’s heart rate, etc. Therefore, it can be seen as a method of forecasting to predict the future depending on previously obtained information. The two most popularly used techniques for the purpose of time series analysis are LSTM and ARIMA. The original datasets have been divided into 80% (training) dataset and 20% (testing) dataset. Also, the number of epochs has been set to 150.
2.1.1. Auto-Regressive Integrated Moving Average (ARIMA)

The ARIMA model has been presented by Box and Jenkins in 1970. It is also time to time referred to as the Box-Jenkins methodology consisting of a set of exercises for assessing, diagnosing, and identifying ARIMA models with the time series data (Ariyo et al., 2014). It is actually a combination of three different models, which are the autoregressive, integrated, and moving average models. ARIMA model can be implemented on a non-stationary type of data. Non-stationary type of data has means, variances, and co-variances that change over time (W. Wang et al., 2015).

ARIMA models can be represented with $p$, $d$, $q$

$p$: the count of total time lags in the Auto-Regressive model

$d$: the degree of the differencing model

| Table 3. Tuning parameters of Kalman filter |
|--------------------------------------------|
| Noise Covariance Matrix (Q) | Noise Covariance Matrix (R) | Initial State Error Covariance Matrix (P) |
|-----------------------------|-------------------------------|-------------------------------------------|
| 0.001                       | 1                             | 1000                                      |

| Table 4. Comparison of Mean and Variance |
|------------------------------------------|
| Air Pollutant     | Mean  | Variance |
|-------------------|-------|----------|
| CO_CPCB           | 0.463 | 0.033    |
| CO_Sensor         | 0.774 | 0.222    |
| CO_Sensor Fusion Data | 0.504 | 0.030    |
| NH3_CPCB          | 6.990 | 1.311    |
| NH3_Sensor        | 6.268 | 2.633    |
| NH3_Sensor Fusion Data | 6.750 | 0.941    |
| O3_CPCB           | 33.44 | 232.25   |
| O3_Sensor         | 43.42 | 184.49   |
| O3_Sensor Fusion Data | 39.00 | 136.18   |

| Table 5. Comparison of LSTM and ARIMA implementation on test dataset of pollutants |
|-----------------------------------------------------------------------------------|
| Air Pollutant     | ARIMA  | ARIMA | LSTM | LSTM  |
|                   | RMSE   | MAE   | RMSE | MAE   |
|-------------------|--------|-------|------|-------|
| CO_CPCB           | 0.13   | 0.12  | 0.13 | 0.09  |
| CO_Sensor         | 0.09   | 0.08  | 0.02 | 0.01  |
| CO_Sensor Fusion Data | 0.36  | 0.28  | 0.18 | 0.09  |
| NH3_CPCB          | 0.44   | 0.39  | 0.14 | 0.13  |
| NH3_Sensor        | 0.48   | 0.44  | 0.29 | 0.26  |
| NH3_Sensor Fusion Data | 0.16  | 0.12  | 0.13 | 0.11  |
| O3_CPCB           | 13.6   | 13.13 | 2.06 | 1.70  |
| O3_Sensor         | 3.66   | 3.01  | 2.82 | 2.55  |
| O3_Sensor Fusion Data | 3.23  | 2.91  | 1.90 | 1.69  |
q: the order of moving average model

ARIMA can be implemented on stationary datasets, so it is important to turn the non-stationary dataset to a stationary dataset. The ARIMA model is used a lot for forecasting. Also, it has been utilized for its known ability to produce short moment forecasts. This model works in such a way that the future estimation of a variable is a direct blend of past errors and past qualities (Ariyo et al., 2014).

It can be represented mathematically as shown in equation (1)
Figure 4. Carbon monoxide sensor module.

Figure 5. MQ-131 Ozone sensor Module.

Figure 6. MQ-135 Air quality sensor module.
\[ Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_1 - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q} \]  

\[ Y_t = \text{actual value} \]
\[ \varepsilon_1 = \text{random error at } t \]
\[ \phi_i = \text{coefficient} \]
\[ \theta_j = \text{coefficient} \]

If any trend is identified in the data, it is removed using the process of differencing and power transformation. The important steps in building ARIMA models are the model identification, parameter estimation, and the model utilization. ARIMA model development as a flowchart can be represented as shown in Figure 2.

The time series analysis of air pollutant level data is done by both the ARIMA and the LSTM models. The ARIMA model is basically made of three different individual models, which are autoregressive model \((p)\), differencing model \((d)\), and moving average model \((q)\). To calculate the coefficients of Auto-Regressive and Moving average models, Partial Auto-Correlation Function (PACF) and Auto-Correlation Function (ACF) are used. The number of times we differentiate the data determines the coefficient of differencing. The Dickey-Fuller test determines if the data are stationary or not; as a result, it decides the number of times we have to differentiate. The dickey fuller test also confirms that the data is non-stationary. So, the data are differentiated twice to make it stationary, and the coefficients are given in Table 1.

ARIMA model has been made by using the training data by using the coefficients from the table. The results are then validated against the test dataset, which has been kept completely separate.
Figure 8. Circuit board.

Figure 9. Sensor fusion of IoT and ground station data.
2.1.2. Deep Long short-term memory (LSTM)

In a time-series dataset, a certain dependency amongst the input variables is noticed. The recurrent neural network (RNN) has been known to be very reliable to handle the dependency in all the input variables. LSTM comes under a special category of RNN with some features to learn the sequence of data. Each of the LSTM comprises a batch of modules in which the information streams have been stored. These modules represent a transport line between each module. They relay information from the past and gather it for the present. The purpose of the gates in the cells is that the data can be filtered, disposed of, or added. The basis of these gates is sigmoidal neural networks. They allow the cells in some situations to let the data pass or dispose of (Siami-Namini et al., 2018). Gates are found to be of three types, which are being used in each LSTM to control the state of each cell. Forget Gate gives a number between 0 (i.e. ignore it) and 1 (i.e. keeping it).

The memory gate picks the new information that is kept in the cell. A sigmoid layer opts for the values that are to be modified followed by the ‘tanh’ layer for making a vector of the new values that could be appended to the state. The output gate controls the output of each and every cell. The output value depends on the cell state, and the filtered and newly added data. The parameters of the LSTM can be seen in Table 2.

An LSTM cell consists of the following components:

1. Forget Gate (f)
2. Candidate Gate (C)
3. Input Gate (I)
4. Output Gate (O)
5. Hidden State (H)
6. Memory State (C)

In an original LSTM model, let the input at time step t be \( x_t \) (Siami-Namini et al., 2018). To calculate the hidden state (St) from the last state (St-1) (Sagheer & Kotb, 2019), the first step would be to decide which data are going to be tossed from the cell state. This is decided by forget gate (ft):

\[
f_t = \sigma(x_t U_f + S_{t-1} W_f + b_f)
\]

Next, it has to be decided which of the information is kept in the cell state, and this step has two folds. First, the input gate (It) layer decides which of the values should be updated. Second, the tanh layer creates a vector of new candidate values Ct. It can be represented as

\[
i_t = \sigma(x_t U_i + S_{t-1} W_i + b_i)
\]

Then, the old state Ct-1 updates to a new cell state Ct. It could be represented as

\[
C_t = C_{t-1} \odot f_t \odot i_t \odot C_t
\]

The output depends on the state of the cell, and the output gate decides the part of the cell state that is produced as the output. Also, the cell states go past the ‘tanh’ layer and then multiply it by the output gate.
The LSTM has three groups of parameters:

1. Weights that are inputs: $U^r, U^i, U^o, U^c$
2. Recurrent weights: $W^r, W^i, W^o, W^c$
3. Bias: $b_r, b_i, b_o, b_c$

### 2.2. Performance indices

To compare the accomplishments of LSTM and ARIMA models, certain measures of performance indices are required. Root Mean Square Error (RMSE) comes in the form of statistical criteria and can be used to evaluate and compare the efficiency of both models.

#### 2.2.1. Root mean square error (RMSE)

The performance of a model is said to be high when the value of RMSE is low. The RMSE values tell us how close the actual values are to the predicted values. It can be mathematically expressed as shown in equation (7).
\[
\sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2}
\]  

(7)

2.2.2. Mean absolute error (MAE)

MAE can be explained as the mean of absolute value of errors. It can be derived using equation (8).

\[
\frac{\sum_{i=1}^{n} |y_i - x_i|}{n}
\]  

(8)

2.3. Hardware specifications

The three types of datasets are being used to compare the LSTM and ARIMA models. Among them, the IoT data are obtained from a hardware model consisting of an Arduino UNO, MQ131 Ozone sensor module, MQ-135 Air Quality sensor module, and MQ-7 Carbon monoxide sensor module. This section discusses the hardware specifications of each module and the sensitivity adjustment/calibration for each sensor. Figure 3 shows the IoT node architecture used for collecting the real-time sensor outputs.

2.3.1. MQ-7 Carbon monoxide sensor module

This sensor in Figure 4 is used for the detection of the concentration of carbon monoxide (CO). The standard measuring circuit of the sensor consists of a heating circuit, which has a control function and a signal output circuit that responds to changes in surface resistance of the sensor. The sensor also has a potentiometer to increase and decrease the sensitivity of the sensor. (https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf)
2.3.2. **MQ-131 Ozone sensor module**
MQ 131 in Figure 5 is used to detect ozone concentration. It is a semiconductor sensor. It has four pins for voltage, ground, and output pins. For proper working, it has an inbuilt heater and a potentiometer to adjust the sensitivity of the sensor.

2.3.3. **MQ-135 air quality sensor module**
This sensor in Figure 6 determines the levels of gas concentrations of many gasses at the same time. NOx, NH3, alcohol, benzene, CO2, and smoke can be detected. It has four pins for outputs, voltage, and ground. The given input voltage is 5 volts.

2.3.4. **Arduino uno**
Arduino as shown in Figure 7 works on an ATMEGA328P controller. The board contains Pulse Width Modulation (PWM), analog, and digital pins. Arduino has six analogs (can be utilized for PWM tasks) and 14 digital pins. The Arduino can be programmed with Arduino Integrated Development Environment (IDE). It is primarily powered by a USB cable.

The circuit board with the connections as shown in Figure 8 collects data from sensors.

2.4. **Sensor fusion**
The sensor fusion is used to get more accurate values by the fusion of ground station data and the IoT data from the sensors. For sensor fusion, the Kalman filter algorithm is used. Kalman filter is a linear weighted average of two sensor values. A basic block diagram of sensor fusion is shown in Figure 9. If we take a time series of the following information points $x_1, x_2, x_3$ $\ldots$ $x_n$, someone who can forecast would compute an estimate for the point $x_{n+2}$. A smoother
Figure 13. LSTM and ARIMA implementation on a training dataset of Fused O3.

Figure 14. LSTM and ARIMA implementation on a training dataset of Fused NH3.
would take a look back at the data and would calculate the best $x_i$ by considering the points before and after $x_i$. A filter can correct $x_{n+1}$ by considering $x_1, x_2, x_3 \ldots \ldots x_n$, and not an exact measure of $x_{n+1}$.

For example, we have a time series with $x_1, x_2, x_3 \ldots \ldots x_n$. The average run of the time series is given by equation (9)

$$\mu_n = \sum_{1}^{n} x_i \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots }
An estimate of the new standard deviation is given by

\[
\sigma_n^2 = \sigma_n^2 + K(x_{n-1} - \mu)^2 \tag{12}
\]

\[
\sigma_{n+1}^2 = (1 - k)\sigma_n^2 \tag{13}
\]

Taking a situation in which there are two instruments with a reading \(x\). The reading from an instrument can be \(x_1\) and that of another instrument can be \(x_2\). It is known that the first instrument has an error, which is modeled by a Gaussian with a standard deviation of one. The second instrument has an error that is normally distributed around zero with a standard deviation of two. The objective is to combine both the readings so that there is a single estimation. In the event, both the instruments are considered good on an equal level and the average of both the numbers is computed. If an instrument is considered as superior to the other, then the estimate of the superior is kept.

The goal is to get a weighted average of the readings to compute an approximate of \(x\), which is denoted by \(\hat{x}\)

\[
\hat{x} = \frac{x_1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} \tag{14}
\]

It can be simplified to get
\[
\dot{x} = \frac{x_1 \sigma_1^2 + x_2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{15}
\]

It can be expressed as

\[
\dot{x} = x_1 + K(x_2 - x_1), \tag{16}
\]

where the gain is

\[
k = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}. \tag{17}
\]

It is known that the error curve is Gaussian, so the probability of \(x\) being the right measure is computed and mathematically given by

for the first instrument

\[
P_1(x) = \frac{1}{\sqrt{2\pi\sigma_1}} e^{-\frac{(x-x_1)^2}{2\sigma_1^2}} \tag{18}
\]

and

\[
P_2(x) = \frac{1}{\sqrt{2\pi\sigma_2}} e^{-\frac{(x-x_2)^2}{2\sigma_2^2}} \tag{19}
\]
It is known that both the measurements do not depend on each other, so they can be merged by multiplying their probability distributions and normalizing.

\[ p(x) = p_1(x)p_2(x) = Ce^{\frac{-\frac{1}{2}(x-x_0)^2}{\sigma_1^2}} \]

where \( C \) is the obtained constant after multiplying the probability distributions.

\[ p(x) = Fe^{\frac{-\frac{1}{2}(x-x_0)^2}{\sigma_2^2}} \]

The center of distribution can be expressed as

\[ \hat{x} = \frac{x_1\sigma_2^2 + x_2\sigma_1^2}{\sigma_2^2 + \sigma_1^2} \]

(22)

The variance can be expressed as

\[ \sigma = \frac{\sigma_2^2\sigma_1^2}{\sigma_2^2 + \sigma_1^2} \]

(23)

If the gain factor is included, then the best estimate can be written as

\[ \hat{x} = x_1 + K(x_2 - x_1) \]

(24)

Hence, the general form of Kalman filter can be seen as
\[ \hat{\sigma} = (1 - K)\sigma^2 \]  \hspace{1cm} (25)

Figure 9 gives us a holistic view of the overall system.

The sensor fusion algorithm has been extensively explained and derived in this section. The tuning parameters of Kalman filter are given in Table 3.

2.5. Sensor calibration

Sensors have to be calibrated before they can be used to get data from the environment. Calibration is necessary as this improves the accuracy of the sensor by reducing errors in the obtained output from the sensor. Steps for calibration:

1. Takedown the load resistance value and check the sensitivity characteristic of the sensor from the sensor datasheet provided by the manufacturer. The details are given in Appendix – A
2. The graph plots the ratio of \( R_s/R_o \) and Parts Per Million (ppm or PPM). The plot shows the sensitivity of the sensor changes for all the gases it is made to detect.
3. Now, take two points from the sensor graph and with these two points form a line, which is approximately equal to the original curve.
4. The sensor resistance of clean air \((R_o)\) is found and the change in sensor resistance \(R_s\) value in fresh air is also calculated.
5. The idea behind this script for calibration and reading is to create a line and calculate the amount of gas. To make these two points are required for calculating the slope.
6. Once the \(R_o\) value is found, the value of \(R_s\) can also be calculated. The desired ratio \(R_s/R_o\) is then calculated with the help of Arduino code.
7. This is taken as a whole and plotted as linear regression to show the relation between the ratio \(R_s/R_o\) and the PPM.
8. The sensitivity characteristics for each sensor are plotted individually and shown in Figures 10, 11, and 12.

From the characteristic curves in Figures 10, 11, and 12, it can be seen that the sensor has been calibrated as the sensitivity characteristics match with the datasheet sensitivity characteristics given in Appendix – B. Hence, the sensor’s calibration is confirmed. The IoT node measurements are contrasted and the air pollutant information from the Central Pollution Control Board (CPCB) (https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf), Authority of Vijayawada is introduced in Appendix – C. The prototype cost analysis is also given in Appendix – C.

3. Experimental results and analysis

3.1. Data pre-processing

Each of the air pollutants has its sub-indices, which are set according to the CPCB, India. Hence, the data have to be processed before they are used to implement the machine learning techniques. Air pollutants in some situations have non-existing or zero values. This is solved by removing the value from the data set. The CPCB data has been collected from the ground station of Vijayawada. The test data have 15 samples, while the training data have 520 samples. The data have been taken 15 minutes apart. For forecasting, 15 steps have been taken into the future.
3.2. Sensor fusion data

A general algorithm of sensor fusion that can be used to obtain the values of IoT sensor reading and ground station data. In a general situation, a process that undergoes state transitions $x_{k-1}, x_k$, and $x_{k-2}$. The propagator matrix $A$ permits us to forecast the state at a time $K + 1$, given the best past estimate $\hat{x}_k$ of the state at time $k$:

$$\hat{x}_{k+1} = A\hat{x}_k + v$$  \hspace{1cm} (26)

The error in the forecast is given by $v$, a variable that has a Gaussian distribution which centers at 0. If the covariance matrix of the system state estimated at time $k$ is named $P$, the covariance matrix of $\hat{x}_{k+1}$ can be expressed as

$$P_{k+1} = A^T P_k A + Q,$$  \hspace{1cm} (27)

where $Q$ is the covariance matrix of the noise $v$.

The measurement $z_{k+1}$ is prone to errors. The state and the measurement are linearly related to

$$z_k = Hx_k + R,$$

where $H$ is a matrix and $R$ is the covariance matrix of the measurement error.

The Kalman gain:

$$K_{k+1} = P_{k+1} H^T (HP_{k+1} H^T + R)^{-1}$$  \hspace{1cm} (28)

State update:

$$x_{k+1} = \hat{x} - K_{k+1} [z_{k+1} - H\hat{x}_{k+1}]$$  \hspace{1cm} (29)

Covariance update:

$$P_{k+1} = (1 - K_{k+1} H) P_{k+1}$$  \hspace{1cm} (30)

The ground station data and the sensor data have been fused using the Kalman filter. The estimated value is the fused data. Mean and variance of fused data, ground station and IoT data have been computed and are tabulated in Table 4. It can be seen from Table 4 that the mean and variance of the sensor fused data are much lower than when compared to both the ground station and the IoT data. An inference can be made from the table that the sensor fusion data have less uncertainty associated with measurements and more accuracy compared with both ground station and IoT data. Because of less variance, the fused data can be used for accurate forecasting.

3.3. ARIMA and LSTM implementation

In general, ARIMA and LSTM have been used very popularly for forecasting and predictions; therefore, both ARIMA and LSTM models have been compared to know which model is better for this use case. Furthermore, the results of ARIMA and LSTM on fused data are compared with the results from the sensor and the ground station data.

Figure 13 shows the implementation of the LSTM and ARIMA model on the training dataset after the sensor fusion of O3 sensor data and ground station O3 data.
Figure 14 shows the implementation of the LSTM and ARIMA model on the training dataset after the fusion of NH3 sensor data and NH3 actual data from CPCB.

Figure 15 shows the implementation of the LSTM and ARIMA model on the training dataset after the sensor fusion of CO sensor data and CO ground station data.

Both the LSTM and ARIMA models have been validated against the unseen test dataset. Moreover, MAE and RMSE have been used to evaluate the model accuracy.

Figures 16, 17, and 18 show the forecasting results of ARIMA and LSTM on a fused test dataset of CO, O3, and NH3, respectively. The RMSE and MAE of each implementation can be seen in Table 5.

The least MAE and RMSE values have been observed in LSTM in all the cases and hence can be concluded that LSTM is a better fit for the dataset than the ARIMA model. Also, it can be observed that the implementation of the LSTM and ARIMA model has the least MAE and RMSE in the sensor fusion dataset of O3 and NH3 compared with the sensor data and ground station data. Therefore, the sensor fusion has resulted in less uncertainty and more accuracy when compared to the data from CPCB and the sensor data.

4. Conclusion
This paper has dealt with the objective of forecasting three different air pollution levels such as CO, NH3 and O3. This has been done by implementing the LSTM model and the ARIMA model and then comparing both of them to see which gives the best results. Both the ARIMA and the LSTM model are used popularly for time series analysis, but in this case, the LSTM model has proven to be a better model for forecasting air pollution levels. Also, to enhance the accuracy of our forecasting result, the concept of sensor fusion is used by implementing the Kalman filter. The fused output has shown much lower variance and means from that of the sensor data and the ground station data. Future works can be done by choosing better algorithms and then using the cloud-like Microsoft azure for deploying the model. Also, a dashboard could be included for the better visualization of results.

Acknowledgments
We thank our PG student V JOSHI KUMAR of Vellore Institute of Technology, Vellore, for his assistance in real-time data logging using IoT sensor Node in Vijayawada Region.

Funding
The authors received no direct funding for this research.

Author details
Geetha Mani1
E-mail: geetha.mani@vit.ac.in
Rohit Voley2
E-mail: rohit.volety2018@vitstudent.ac.in

1 School of Electrical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India.
2 School of Electronics Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India.

Disclosure of potential conflicts of interest
No potential conflict of interest was reported by the author(s).

Citation information
Cite this article as: A comparative analysis of LSTM and ARIMA for enhanced real-time air pollutant levels forecasting using sensor fusion with ground station data, Geetha Mani & Rohit Voley, Cogent Engineering (2021), 8: 1936886.

References
Ariyo, A. A., Adebunmi, A. O., & Ayo, C. K. (2014, March). Stock price prediction using the ARIMA model. In 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation (pp. 106–112). IEEE held in Cambridge, UK.

Ho, Q. P., Metia, S., & Phung, M. D. (2020). Sensing data fusion for enhanced indoor air quality monitoring. IEEE Sensors Journal, 20(8), 4430–4441. https://doi.org/10.1109/JSEN.2020.2964396

Kulkarni, G. E., Muley, A. A., Deshmukh, N. K., & Bhalchandra, P. U. (2018). Autoregressive integrated moving average time series model for forecasting air pollution in Nanded city, Maharashtra, India. Modeling Earth Systems and Environment, 4(4), 1435–1444. https://doi.org/10.1007/s40808-018-0493-2

Li, H., Wang, J., Li, R., & Lu, H. (2019). Novel analysis-forecast system based on multi-objective optimization for air quality index. Journal of Cleaner Production, 208, 1365–1383. https://doi.org/10.1016/j.jclepro.2018.10.129

Li, P., & Zhang, J. S. (2018). A new hybrid method for China’s energy supply security forecasting based on ARIMA and xgboost. Energies, 11(7), 1687. https://doi.org/10.3390/en11071687

Peng, H., Lim, A. R., Teokles, A., Jin, J., Cannor, A. J., & Hsieh, W. W. (2017). Evaluating hourly air quality forecasting in Canada with nonlinear updatable machine learning methods. Air Quality, Atmosphere & Health, 10 (2), 195–211. https://doi.org/10.1007/s11869-016-0414-3
Sagheer, A., & Kotb, M. (2019). Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing, 323, 203–213. https://doi.org/10.1016/j.neucom.2018.09.082
Sankar Ganesh, S., Arulmozhiwarman, P., & Tatavarti, R. (2017). Forecasting air quality index using an ensemble of artificial neural networks and regression models. Journal of Intelligent Systems, 28(5), 893–903. https://doi.org/10.1015/j/jisys-2017-0277
Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2018, December). A comparison of ARIMA and LSTM in forecasting time series. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1394–1401). IEEE.
Wang, D., Wei, S., Luo, H., Yue, C., & Gruender, O. (2017). A novel hybrid model for air quality index forecasting based on two-phase decomposition technique and modified extreme learning machine. Science of the Total Environment, 580, 719–733. https://doi.org/10.1016/j.scitotenv.2016.12.018
Wang, P., Zhang, H., Qin, Z., & Zhang, G. (2017). A novel hybrid-Garch model based on ARIMA and SVM for PM2.5 concentrations forecasting. Atmospheric Pollution Research, 8(5), 850–860. https://doi.org/10.1016/j.apr.2017.01.003
Wang, W., Chau, K., Xu, D., & Chen, X. (2015). Improving forecasting accuracy of annual runoff time series using ARIMA based on EEMD decomposition. Water Resources Management, 29(8), 2655–2675. https://doi.org/10.1007/s11269-015-0962-6
World Health Organization. 2016. Ambient air pollution: a global assessment of exposure and burden of disease. World Health Organization. https://apps.who.int/iris/handle/10665/250141.
Wu, Q., & Lin, H. (2019). Daily urban air quality index forecasting based on variational mode decomposition, sample entropy and LSTM neural network. Sustainable Cities and Society, 50, 101657. https://doi.org/10.1016/j.scs.2019.101657
Appendix – A

| Air Pollutant Sensor | Specifications |
|----------------------|----------------|
| MQ 135               | RL = 20kΩ      |
|                      | Rs = 30kΩ to 200kΩ |
| MQ 7                 | RL = 5kΩ       |
|                      | Rs = 2kΩ to 20kΩ |
| MQ 131               | RL = 50 kΩ     |
|                      | Rs = 2kΩ to 20kΩ |

RL = Load Resistor of Sensor

RS = Resistor of Sensor Due to Concentration of gas

Appendix B: Sensitivity Chart given in Manufacturer datasheet Subscription

(i) Sensitivity Chart for MQ – 7

![Sensitivity Chart for MQ – 7](image)

(ii) Sensitivity Chart for MQ – 131:

![Sensitivity Chart for MQ – 131](image)
(i) Sensitivity Chart for MQ - 135:

Appendix C
i) Sample Data comparison of CPCB and IoT sensor node of Carbon Monoxide (Co)

| Date and Time     | IoT Sensor Node Data | CPCB_Vijayawada |
|-------------------|----------------------|-----------------|
| 2020-04-08 21.00.00 | 0.39                 | 0.35            |
| 2020-04-08 21.15.00 | 0.38                 | 0.38            |
| 2020-04-08 21.30.00 | 0.53                 | 0.47            |
| 2020-04-08 21.45.00 | 0.56                 | 0.65            |
| 2020-04-08 22.00.00 | 0.4                  | 0.72            |
| 2020-04-08 22.15.00 | 0.35                 | 0.73            |
| 2020-04-08 22.30.00 | 0.39                 | 0.76            |
| 2020-04-08 22.45.00 | 0.42                 | 0.77            |
| 2020-04-08 23.00.00 | 0.39                 | 0.76            |
| 2020-04-08 23.15.00 | 0.37                 | 0.76            |
| 2020-04-08 23.30.00 | 0.38                 | 0.74            |
| 2020-04-08 23.45.00 | 0.4                  | 0.76            |
| 2020-04-09 00.00.00 | 0.22                 | 0.58            |
| 2020-04-09 00.15.00 | 0.21                 | 0.53            |
| 2020-04-09 00.30.00 | 0.18                 | 0.55            |
| 2020-04-09 00.45.00 | 0.24                 | 0.55            |
| 2020-04-09 01.00.00 | 0.22                 | 0.55            |
| 2020-04-09 01.15.00 | 0.09                 | 0.54            |
| 2020-04-09 01.30.00 | 0.77                 | 0.53            |
| 2020-04-09 01.45.00 | 0.06                 | 0.52            |
ii) Sample Data comparison of CPCB and IoT sensor node of Ammonia (NH₃)

| Date and Time     | IoT Node Sensor Data | CPCB_Vijayawada |
|-------------------|----------------------|-----------------|
| 2020-04-09 21.00.01 | 4.8                  | 6.8             |
| 2020-04-09 21.15.01 | 5.2                  | 6.8             |
| 2020-04-09 21.30.01 | 5.3                  | 7.1             |
| 2020-04-09 21.45.01 | 5.3                  | 7               |
| 2020-04-09 22.00.01 | 4.9                  | 6.9             |
| 2020-04-09 22.15.01 | 4.8                  | 6.8             |
| 2020-04-09 22.30.01 | 5.2                  | 6.6             |
| 2020-04-09 22.45.01 | 5.4                  | 6.5             |
| 2020-04-09 23.00.01 | 4.8                  | 6.2             |
| 2020-04-09 23.15.01 | 5.1                  | 6.1             |
| 2020-04-09 23.30.01 | 5.2                  | 6.3             |
| 2020-04-09 23.45.01 | 5.2                  | 6.5             |
| 2020-04-10 00.00.01 | 4.7                  | 6.5             |
| 2020-04-10 00.15.01 | 4.5                  | 6.5             |
| 2020-04-10 00.30.01 | 4.8                  | 7               |
| 2020-04-10 00.45.01 | 5.1                  | 7.2             |
| 2020-04-10 01.00.01 | 5.1                  | 7               |
| 2020-04-10 01.15.01 | 5.4                  | 6.7             |
| 2020-04-10 01.30.01 | 5.6                  | 6.7             |
| 2020-04-10 01.45.01 | 5.6                  | 6.9             |

(i) Sample Data comparison of CPCB and IoT sensor node of Ozone (O₃)
(i)  
Cost Analysis of Single IoT Node

| Date and time       | IoT Node Data | CPCB_Vijayawada |
|---------------------|---------------|-----------------|
| 2020-04-08 21.00.00| 49.4          | 32.3            |
| 2020-04-08 21.15.00| 47.1          | 30.4            |
| 2020-04-08 21.30.00| 38.4          | 30.2            |
| 2020-04-08 21.45.00| 30.4          | 29.8            |
| 2020-04-08 22.00.00| 26            | 29              |
| 2020-04-08 22.15.00| 25.9          | 28.8            |
| 2020-04-08 22.30.00| 29.9          | 28.4            |
| 2020-04-08 22.45.00| 30.4          | 28              |
| 2020-04-08 23.00.00| 28.4          | 28.2            |
| 2020-04-08 23.15.00| 29.9          | 28.2            |
| 2020-04-08 23.30.00| 32.5          | 27.6            |
| 2020-04-08 23.45.00| 30.9          | 27.6            |
| 2020-04-09 00.00.00| 33.2          | 26.7            |
| 2020-04-09 00.15.00| 34.3          | 26.7            |
| 2020-04-09 00.30.00| 33            | 27              |
| 2020-04-09 00.45.00| 31.9          | 27.2            |
| 2020-04-09 01.00.00| 31.1          | 24.3            |
| 2020-04-09 01.15.00| 32            | 24.5            |
| 2020-04-09 01.30.00| 33.9          | 22.5            |
| 2020-04-09 01.45.00| 35.6          | 22.1            |

| Hardware Components Utilized | Cost per unit (INR) |
|-----------------------------|---------------------|
| Arduino Uno                 | 300                 |
| MQ7 Sensor                  | 150                 |
| MQ - 131                    | 2500                |
| MQ - 135                    | 125                 |
| Wiring components           | 50                  |
| Total Cost required to develop the prototype | 3,125.00 |
