International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

Fusing Long Short-Term Memory and Autoencoder Models for Robust Anomaly Detection in Indoor Air Quality Time-Series Data

M.Veera Brahmam[1], S.Gopikrishnan[2]
[1]School of Computer Science and Engineering
VIT-AP University
Amaravati, AP, India
veerabrahmam.20phd7156@vitap.ac.in
[2]School of Computer Science and Engineering
VIT-AP University
Amaravati, AP, India
gopikrishnan.s@vitap.ac.in

Abstract—People spend most of their time indoors by choice or by need. Carbon dioxide (CO2) accumulation can cause various adverse health effects, including vertigo, headache, and fatigue. Therefore, monitoring indoor air quality(IAQ) is necessary for various health reasons. The market is flooded with air quality monitoring devices. However, the ordinary public does not make use of them because they are expensive and difficult to obtain. Several research studies have been carried out to monitor indoor air quality with the help of the Internet of Things(IoT), which has greatly simplified the method for monitoring IAQ. In this research, we offer an improved IoT based IAQ monitoring system with AI-powered recommendations. Our suggested system relies on the Message Queuing Telemetry Transport(MQTT) protocol for communication between IoT devices. In addition, the gathered CO2 occupancy data is used together with the deep learning approach of Long Short-Term Memory and Autoencoder (LSTM-AE) to detect anomalies or outliers in CO2 concentrations. Due to a close connection between air quality and human health and well-being, the detection of anomalies in the data of IAQ has emerged as an essential topic of study. Anomalies requiring the observation of correlations spanning numerous data points (i.e., often referred to as long-term dependencies) were not detectable by conventional statistical and basic machine learning (ML) related techniques in the sector of IAQ. Hence this research uses the LSTM-AE model to address this issue. In comparison to previous similar models, our experimental results on a generated CO2 occupancy time series reveal a robust and powerful accuracy of 99.49%.

Keywords- IoT; Deep Learning; LSTM; Autoencoder; Anomaly detection; IAQ.

I. INTRODUCTION

People spend most of their time inside, thus their health, comfort, and overall well-being are significantly impacted by the air quality within buildings and other structures. For example, allergies, asthma, and other respiratory problems can be brought on or worsened by poor indoor air quality (IAQ). Additionally, it may result in tiredness, headaches, vertigo, and nausea [1]. Gases, particulate matter, mould, bacteria, and other contaminants can impact IAQ. Both indoor and outdoor environments can suffer from poor indoor air quality (IAQ) due to various sources of pollution. Outdoor pollutants, including emissions from vehicles and industries, can enter buildings through windows and doors [2]. On the other hand, indoor pollution can arise from combustion appliances, building materials, cleaning products, and office equipment. One widely used indicator for assessing IAQ is CO_2, which is present in abundance [3, 4]. Notably, CO_2 is generated whenever an individual exhales.

Emerging research suggests a connection between CO_2 levels and susceptibility to COVID-19 infection [5]. As infected

Individuals release both pathogens and CO_2 through exhalation, monitoring indoor CO_2 concentrations becomes a dependable indicator of potential infection risk. Consequently, the measurement of CO_2 levels has been proposed as an indirect means to assess the likelihood of transmitting infectious respiratory diseases [6]. Investigating indoor carbon dioxide (CO_2) levels also impact occupancy tracking, which can impact buildings' energy usage [7]. The utilization of CO_2 sensors to determine occupancy has been explored in various studies and holds promise [8, 9, 10]. Precise monitoring of building occupancy plays a vital role in achieving energy efficiency, with potential energy savings ranging from 30% to 40% in certain instances [10]. Moreover, comprehending the indoor CO_2 levels is crucial for safeguarding the well-being and safety of occupants, as government regulations and industry guidelines set distinct permissible concentration thresholds for indoor spaces.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

As an example, CO_2 concentration is typically restricted to a maximum of 1000 ppm in many applications [11]. However, European standards set the permissible limit for CO_2 in indoor IAQ at 1500 ppm [12]. Considering that occupants exhale a significant amount of CO_2 , it is crucial to design an effective anomaly detection system to avoid biased decisions when analyzing the dataset. The rise of the Internet of Things has made affordable sensors and open-source IoT platforms widely accessible, enabling their combination with artificial intelligence technologies in IAQ systems. This research therefore provides a sophisticated method for detecting anomalies in CO_2 concentrations. There are two major contributions made by this paper:

- The authors of this study used Internet of Things (IoT) sensors to create a system for continuously monitoring indoor air quality.
- The authors used LSTM-AE model to predict or forecast future CO2 concentration levels based on historical or existing data. In order to understand patterns in sequences, the model employs a LSTM network made up of several LSTM units. To further enhance training effectiveness, AE reduces the dimensionality of data.
- The performance of our proposed model is compared with other similar approaches that utilize various aspects of LSTM and/or AE. We conducted experiments using a comprehensive set of evaluation criteria, and the results show that our model is capable of detecting anomalies effectively, achieving a detection accuracy of over 99%.

The remainder of the paper is divided into five sections: Section 2 presents the related works on data prediction and anomaly detection in sensor networks. Section 3 presents the preliminaries section. The problem formulation for outlier detection in IoT is described in Section 4. Section 5 outlines the recommended approach and technique. Section 6 presents the experimental results and evaluation. Section 7 comes to end with conclusion section.

II. RELATED WORKS

Kallio et al. [11] compared two different approaches for predicting CO_2 levels. They found that a decision tree outperformed an Artificial Neural Network (ANN) regarding energy consumption and computational efficiency. Additionally, they discovered that using a one-minute forward predicting time-window technique resulted in the accuracy higher than that of the accuracy achieved with ten- or fifteen-minute time windows. However, the inclusion of additional factors such as humidity and temperature did not affect the accuracy of the CO_2 forecast. Additionally, the researchers developed two neural network-based systems for predicting CO_2 concentration and one system for forecasting the comfort conditions during the daytime, considering temperature, humidity, and CO_2.

Mumtaz et al. [13] suggested an approach for IAQ assessment that provides users access to a web portal and a mobile application that visually represents the air quality. They determined the IAQ level based on five air quality parameters(CO_2, CO, NO_2, CH_4, and $PM_{2.5}$). The authors ranked IAQ conditions using neural networks and achieved a 99.1 percent accuracy rate. Future CO_2 concentrations were also predicted using Long-Short-Term Memory (LSTM). However, the achieved IAQ classification relied on the data collected at outdoors, which is affected by numerous other factors and may need to be revised for an indoor environment. In addition, sensor lifetime and calibration factors may be complex for their technique.

In contrast, Ahn et al. [14] developed an IAQ prediction system using LSTM and its variant Gated Recurrent Unit (GRU). They found that GRU outperformed LSTM with an accuracy rate of up to 84.69 percent. However, the optimization of the time step size for this model took approximately 38 hours. Furthermore, Tagliabue et al. [15] proposed a data collection architecture using the IoT network and introduced two prediction models. One model forecasted the comfortable conditions of temperature, humidity, and CO_2 for a day, while the other model predicted CO_2 concentration using neural networks. Their research revealed that the Mean Square Error (MSE) for the test period was approximately 75 ppm (10.6%) compared to the average CO_2 content.

Using a deep neural network, Wambura et al. [16] proposed the One Sketch Fits All Time (OFAT) technique for solving the problem of accurate long-range forecasts inside high-dimensional feature-evolving time series. The suggested approach addresses the difficulty caused by the non-stationary nature of feature-evolving time series, which causes the length of the input sequence (rows) to fluctuate as new data points are added, and their feature values (columns) evolve. Experiments conducted on real-world data sets and a stringent evaluation demonstrated that OFAT has a quick processing time, dependable performance, and precise recognition. However, one of the shortcomings of this method is that it needs to account for interactive real-time forecasting in data streams.

Sharma et.al [17] introduced a cost-effective framework, IndoAirSense, to predict and forecast indoor air quality (IAQ) in specific classrooms at a university. To estimate real-time indoor air quality (IAQ), the researchers employed a combination of multilayer perceptron (MLP) and extreme Gradient Boosting Regression (XGBR). Additionally, they utilized a modified version of the long short-term memory (LSTM) model, known as LSTM without a forget gate (LSTM-wF), to simplify the prediction of indoor air pollutants. However, due to the absence of the forget gate in this model, which is responsible for

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

maintaining long-term memory, it was unable to effectively identify anomalies in the time-series dataset.

Xu et al. [18] presented an LSTM model with an integrated error correction model (ECM) to improve the accuracy of indoor temperature predictions specifically in public buildings. When re-predicting the testing data set, they construct an ECM by co-integrating the data from predictions and actual measurements in the same sequence. Also, Jung et al. [19] applied LSTM for indoor environment prediction in facility management. They collected data on interior temperature, humidity, and lighting from three different Internet of Things (IoT) sensors and trained LSTM to identify instances in which those readings departed from a threshold.

Hossain et al. [20] introduced a dual prediction approach for forecasting the daily air quality index (AQI) in two major cities of Bangladesh, Dhaka, and Chattogram. The first and second hidden layers of the resulting prediction model were respectively a gated recurrent unit (GRU) and a long short-term memory (LSTM) variant of the recurrent neural network (RNN) model. The results showed that when compared to utilising either a GRU or LSTM model, their hybrid model performed better overall and more closely captured the AQI patterns in both locations.

Ottosen and Kumar [21] presented a method for detecting anomalies in a low-cost air quality dataset. They proposed two techniques: one using K-nearest neighbors (KNN) and the other utilizing autoregressive integrated moving average (ARIMA). The KNN approach focused on identifying point anomalies by computing the average Euclidean distance between each point and the points that remained, assigning a score to indicate the level of anomaly for each point. On the other hand, ARIMA was employed for identifying contextual anomalies by comparing the model and measurement data points and Anomaly scores were calculated based on the absolute value of the difference between the observed data and the model predictions. By applying K-means clustering, the dataset was divided into two clusters, distinguishing between normal points and contextual anomalies.

Li et al. [22] suggested a technique using fuzzy C-means clustering. Their methodology involved utilizing a reconstruction criterion to reconstruct the optimal cluster centers and partition matrix. This reconstruction was performed using multivariate subsequences of the data. To establish a threshold for identifying outliers in the multivariate data, they employed a fitness function based on the reconstruction error in conjunction with the particle swarm optimization (PSO) algorithm. However, it should be noted that their proposed algorithm faced challenges in uncovering the underlying structure of high-dimensional multivariate time series, as the PSO algorithm tended to become trapped in local optima.

It is our contention that the current studies in this area have a number of shortcomings. One of these is that traditional time-series techniques, like ARIMA, and machine learning methods that rely on regression (e.g., K-means or KNN) frequently depend on human experts to extract features from the data, which can be time-consuming and costly. Furthermore, these approaches are often affected by outliers and have difficulty dealing with large data sets and unknown probability distributions, leading to poor performance.

Leveraging LSTM in time-series prediction models allows for the anticipation of future values by leveraging patterns inherent in sequential data., leading to more accurate predictions and better decision-making capabilities such as anomaly detection. In contrast to current approaches, our proposed model incorporates AE and can deliver significantly improved results when handling complex auto-correlation sequences with large datasets, even when dealing with unpredictable data distributions.

III. PRELIMINARIES

A. LSTM

Building upon RNNs, LSTM architecture offers the capacity for "long-term memory," allowing the current neural node to access a comprehensive list of all past information, rather than just a single point in time.

- Figure 1 presents an illustration of an LSTM unit, which comprises four essential components: a cell, an input gate, an output gate, and a forget gate. These components work together to regulate the flow of information within the cell across different time intervals. The cell serves as a storage unit, preserving.
- Values over time, while the input gate, output gate, and forget gate control the movement of information into and out of the cell..
- The Cell State represents the network's current long-term memory, holding a collection of past information.
- The preceding Hidden State corresponds to the output generated by the LSTM unit in the previous time step and can be seen as short-term memory.
- Lastly, the input data represents the value received at the current time step.

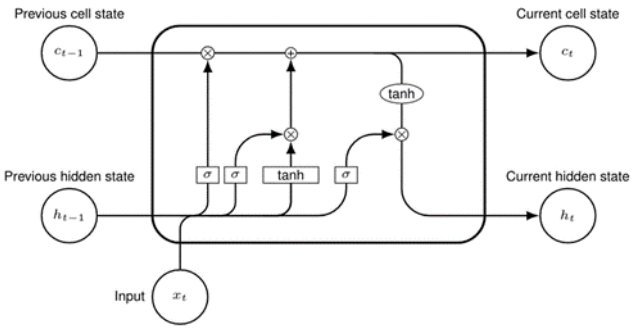

Figure 1. LSTM working mechanism

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

1) Forget gate:

In an LSTM neural network architecture, the forget gate is a key component that controls the flow of information through the memory cell, allowing the LSTM to selectively retain or discard information from previous time steps. The forget gate is responsible for determining how much of the previous memory cell state should be forgotten or erased, based on the input at the current time step and the internal state of the LSTM.

Mathematically, the forget gate in an LSTM is typically implemented as a sigmoid activation function applied element-wise to the weighted sum of the input at the current time step and the output from the previous time step. The sigmoid function maps the weighted sum to a value between 0 and 1, where 0 indicates complete forgetfulness and 1 indicates complete retention. This allows the LSTM to control the amount of information to retain or forget from the previous memory cell state.

The forget gate is defined by the following equations in an LSTM:

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

Where: f_t is the forget gate at time step t. σ is sigmoid activation function W_f is the weighted matrix for the forget gate. h_{t-1} is the output from the previous time step t. x_t is the input at current time step. b_f is the bias for bias term for the forget gate.

The output gate from the forget gate is then used to modulate the previous memory cell state C_{t-1} element-wise, effectively erasing the information that needs to be forgotten.

$$C_t = f_t \odot C_{t-1} \quad (2)$$

By using the forget gate, an LSTM can selectively retain or discard information from previous time steps, allowing it to capture long-term dependencies in sequential data and mitigate the vanishing or exploding gradient problem often encountered in recurrent neural networks.

2) Input gate:

The input gate is responsible for determining how much new input information should be added to the cell state. It takes the current input data and the previous hidden state as inputs and passes them through a sigmoid activation function to generate a gate output vector with values between 0 and 1. This gate output vector determines the amount of new input information that should be allowed to enter the cell state.

Mathematically, the computation of the input gate in an LSTM cell can be represented as follows:

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \quad (3)$$

Where,

i_t is the input gate output at time step t

x_t is the input data at time step t

h_{t-1} is the hidden state from previous time step (t-1)

W_i,U_i and b_i are the learnable weights and biases associated with the input gate

The equation below illustrates the process of determining the proportion of new information, denoted as "$\tilde{C}_t$", among the presented information.

$$\tilde{C}_t = tanh(W_c[h_{t-1}, x_t]) + b_c) \quad (4)$$

Where,

W_c is the weight matrix of the input gate

The bias of the input gate is denoted as b_c

tanh is the activation function utilized in the range of [-1,1] where the negative values are used to reduce the impact.

To control the amount of the additional information, these Two processes () are multiplied point wise.

3) Outpiut gate:

The output gate in LSTM (Long Short-Term Memory) is one of the key components that regulates the flow of information within the LSTM unit. Its primary function is to control the output produced by the LSTM cell at a particular time step.

The output gate takes into account the current input, the previous hidden state, and the current cell state. It computes a sigmoid activation function on a combination of these inputs. The sigmoid function outputs values between 0 and 1, representing the extent to which the information should be allowed to pass through the output gate.

Mathematically, the output gate can be represented as follows:

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

Where,

o_t=output gate activation at time step t.

σ=sigmoid activation function.

W_o= weight matrix associated with the output gate.

h_{t-1}=previous hidden state

x_t=current input

b_o=bias term associated with the output gate

The output gate determines which parts of the current cell state should be exposed as the output. It applies an element-wise multiplication (also known as a Hadamard product) between the output gate activation and the cell state. This selectively controls the information flow and filters out irrelevant or less important details, allowing the LSTM unit to focus on the most relevant information. Mathematically, the output of the LSTM cell is computed as follows:

$$h_t = o_t \cdot tanh(C_t) \quad (6)$$

Where,

h_t =output of the LSTM cell at time step t.

C_t=current cell state.

By adjusting the output gate activation, the LSTM can control the trade-off between preserving long-term memory and

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

producing the most relevant output for the current task. The output gate mechanism allows the LSTM to selectively expose important information to subsequent layers or use it for prediction purposes while suppressing irrelevant or noisy details.

B. Autoencoder

An AE, short for autoencoder, is an unsupervised neural network and is designed to discover useful representations of unlabeled data. Its purpose is to train the neural network to filter out irrelevant or insignificant data, often referred to as "noise," and generate an efficient encoding of the input data set. A typical AE consists of input and output layers, as well as multiple hidden layers. Encoding, latent space representation, decoding, and reconstruction loss are the four main functions of an AE, as shown in Figure 2.

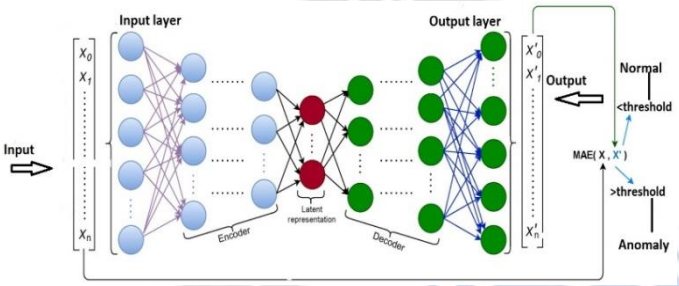

Figure 2. AE Working mechanism

1) Encoding:

The input data is fed into the encoder, typically a series of fully connected layers. Each layer applies linear transformations followed by non-linear activation functions to extract and capture important features of the input data. The encoder gradually reduces the dimensionality of the input and maps it to the lower-dimensional latent space.

2) Latent space representation:

The latent space representation is a compressed version of the input data, capturing the most relevant and essential features. It acts as a bottleneck layer, forcing the network to learn a more compact representation.

3) Decoding:

The decoder takes the latent space representation and aims to reconstruct the original input data. Like the encoder, it typically consists of fully connected layers with non-linear activations. The decoder's purpose is to learn to generate an output that closely resembles the input.

4) Reconstruction loss:

During training, the auto encoder compares the reconstructed output from the decoder with the original input and calculates a reconstruction loss, which quantifies the dissimilarity between them. Common loss functions used for reconstruction include mean squared error (MSE), mean absolute error(MAE), or binary cross-entropy, depending on the nature of the input data. To minimize the gap between the original and reconstructed inputs, a standard AE model, as shown in equation 7, will compute a reconstruction loss (L). This reconstruction loss is often used for the job of spotting outliers.

$$L(x - \hat{x}) = \frac{1}{n}\sum_{t=1}^{n} |x_t - \hat{x}_t| \quad (7)$$

Where,
x= input
$\hat{x}$=output
n=number of training data set samples.

However, in our model, the approach has been broadened to calculate a reconstruction loss as shown in equation 8.

$$x_i = \frac{1}{n}\sum_{n=1}^{n} |x_i - \hat{x}_i| \quad (8)$$

$$n = \begin{cases} i & \leq \frac{N+1}{2} \\ N - i + 1 & n > \frac{N+1}{2} \end{cases} \quad (9)$$

Where,
N= total number of samples
$n = n^{th}$ Sample
$X_i = x_1, \ldots\ldots, x_i$
The authors use the formula10 to get the reconstruction loss
For all-time series samples:

$$reconstructionloss = \frac{1}{N}\sum_{N=i}^{1} x_i \quad (10)$$

Where,
N= number of samples
x= reconstruction loss calculated for every sample

IV. PROBLEM FORMULATION

Using IoT sensors, this study aims to monitor IAQ and CO2 concentration. The authors also measured the change in Particulate Matter (PM) in the presence of a room occupant, but the PM level remained virtually unchanged. Using Pearson's r = 0.01, and a significance level of p = 0.91, the experiment's findings [23] showed a moderate correlation between indoor and outdoor PM levels. A person's PM levels can fluctuate due to routine indoor activities like desk work or leisure whereas CO2 is the most prevalent indoor pollutant associated with high population density. According to the data collected, the rate of change of CO2 is substantially higher than that of PM. Therefore, the authors utilize CO2 as an indicator of IAQ in this study. Our model's LSTM networks consist of numerous LSTM units that can preserve significant feature values in a sequence. The LSTM units work together to understand each sequence's

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

pattern. Additionally, AE decreases data dimensions to make training more efficient.

V. PROPOSED METHOD

This section presents our suggested approach that utilizes a novel approach for predicting future sensor data in IoT network with the LSTM-AE model. The authors begin by outlining their approach, which involves creating the input sequence, and LSTM encoder, LSTM decoder. Then they elaborate on the training and testing phases of their model's algorithm.

Figure 3 provides an overview of our proposed model, which combines an LSTM neural network and an AE to create an LSTM-AE hybrid model. In the first step, we transform the original data set into a high-dimensional input vector of fixed size.

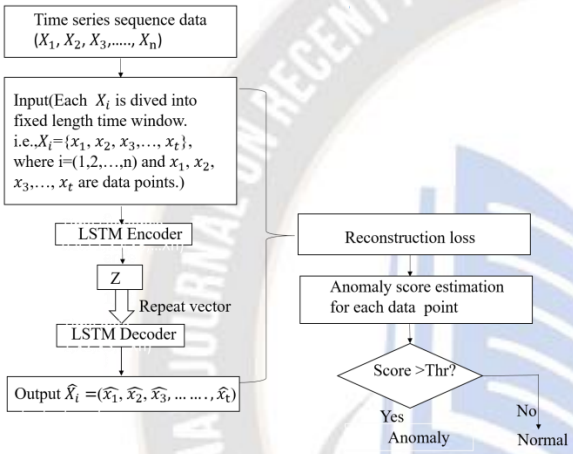

Figure 3. Proposed LSTM-AE architecture

In order to map the high-dimensional input vector representation to the low-dimensional latent space representation, the LSTM encoder uses several LSTM units to analyze the time-series sequence and discover patterns in the features. In order to recover the initial fixed-length input sequence, the LSTM decoder uses this compressed representation in latent space. Finally, the rate of reconstruction error is computed by comparing the output with the input.

The time series data is created as a sequence $[X_1, X_2, X_3, \ldots, X_n]$, where each X is a fixed T-length time window data. This means that the data is divided into multiple windows, and each window has a fixed length of T time units.

Within each time window X, the data is represented as a sequence$[x_1, x_2, x_3, \ldots, x_t]$, where x_t is an m-dimensional input at time instance t. This means that the data captures m features at each time point.

The data is then reshaped into a 2-D array, where the first dimension represents the number of samples, and the second dimension represents the number of time steps. This is done to convert the time series data into a format that can be used for training a machine learning model.

For example, if the sequence represents CO_2 data, the reshaped 2-D array will have the list of samples at ten time steps, with each time step representing the m-features captured at that point in time. This format allows the machine learning model to learn the patterns and relationships in the data across time and make predictions for future time points.

A. Encoder

The LSTM encoder functions as a layer that compresses features into batches of time-based sequences, similar to a folding process. This involves independent convolution operations on the different timesteps of the feature sequences. In order to identify the most important features in the input sequence, the AE encoder interacts with a series of LSTM unit cells, as depicted in figure4.

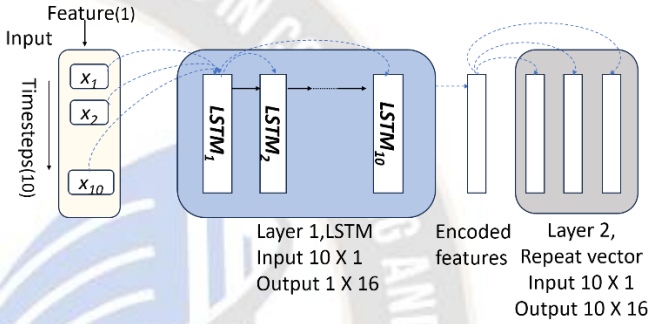

Figure 4. LSTM-Encoder

The X_i time series consists of ten samples collected over ten 1-minute intervals. To prepare the data for the encoder, the 1-dimensional data set is transformed into a 2-dimensional data set. This is accomplished by representing the input data set as a 2D vector, where one dimension represents the ten time steps and the other dimension represents the feature, i.e., the CO_2 readings. This results in a vector of 10x1, which is fed to the encoder. The encoder's first layer includes an LSTM network comprising ten LSTM cell units, with each unit processing one sample. The ten LSTM cells operate sequentially, with the first unit passing its result to the second. The second unit determines whether to retain or forget the previous result from the first LSTM. If it decides to keep it, the second unit stores it in the long-term memory and passes the information of the first sample along with the feature information from the current sample to the third LSTM unit. This process continues in a similar manner for the remaining LSTM units.

In our model, the final LSTM unit receives all the significant information from the nine preceding units. It processes all the valuable samples and produces an output that contains information about all relevant samples. This output is represented as a 1x16 vector and is considered as the encoded features. It should be noted that in our model, we have included a Repeat Vector as the second layer to generate multiple copies of the 1x16 encoded feature vectors, equal to the number of time

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

steps. In our case, the number of time steps is ten, so the Repeat Vector layer creates ten copies of the encoded features as a 2-dimensional vector, which is 10x16 in size.

B. Decoder

The LSTM decoder serves the primary function of behaving as a layer that unfolds a sequence and restores its structure after the sequence has been folded on time steps. Figure 5 depicts in detail how the decoder interacts with LSTM cells to replicate the outputs.

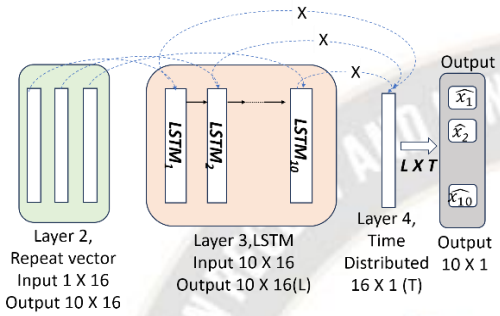

Figure 5. LSTM-Decoder

Each 1x16 set provided to the decoder represents a single characteristic from the original time series. This information is passed on to a layer 3 network made up of ten LSTM cell units. The number of features in the sequence (here, ten data points) is reflected in the number of LSTM cells. Each LSTM cell unit takes in an encoded feature in 1x16 form as input and outputs a learned representation of that feature in the same vector size (1x16). The time distributed layer generates a 1x16 output vector, which is used in a matrix multiplication with the 10x16 output vector. This algorithm yields a 10x1 vector representing the final findings.

C. Anomaly Detection

An anomaly or outlier is an observation that doesn't fit with the rest of the data. A certain threshold can be used as a criterion when a finding is out of the ordinary. Outliers in a data set are called anomalies since they don't fit the typical distribution.

Our model is trained on a data set with CO_2 values within the typical range by applying this threshold-based anomaly detection method. This is necessary for calculating the percentages of inaccuracy in reconstruction error for the normal CO_2 values. After complete training, the maximum reconstruction error rate is chosen, and the various reconstruction errors have been computed across all samples. Once a threshold is established, the entire range of CO_2 values is included in the testing data set. For every sample within the test set, the reconstruction error rate is computed for each CO_2 value. An anomaly sample is one in which the rate of reconstruction error exceeds a threshold.

The proposed model's method is described in detail in Algorithm 1. There are essentially two goals to train for. The primary goal is to lessen the amount of distortion introduced during reconstruction, making it so that the output created from the simplified input representation is a close approximation of the original.

Algorithm1

INPUT: Time series data from CO_2 occupancy data Set

OUTPUT: Reconstruction loss or errors loss

1: **Begin**

1:Training set $(X_i)=\{x_0, x_1, \ldots\ldots\ldots, x_n\}$

2:Test set(X_i') $=\{x_0', x_1', \ldots\ldots, x_m'\}$

3: Time steps=t

4: $X_i = \{x_i, \ldots\ldots\ldots, x_{i+t}\}$ based on 't' time steps(t=10) , where i=0 to (n-t), 'n' is the total training samples.

5: $X_i'=\{x_i', \ldots\ldots\ldots, x'_{i+t}\}$ based on 't' time steps(t=10) , where i=0 to (m-t), 'm' is total testing samples.

/* Training of LSTM-AE */

6: $\widehat{X_i}$ =LSTM-AE(X_i) // LSTM-AE model is applied on X_i , where i=0 to (n-t)

7: error or reconstruction loss=$\sum|\widehat{X_i} - X_i|$

/* Calculation of reconstruction error for X_i */

Assuming 'arr' is the 2D matrix

8: errorloss arr= errorloss arr[i, i:i+t]= $|\widehat{X_i} - X_i|$

/* Calculation of reconstruction error for all trained data */

9: errorloss[i]=$\frac{\sum(\text{errorloss arr}[:,i]}{\sum(\text{error loss arr}[:,i]!=0)}$,

Where,

i=0 to n,.

$\sum$(errorloss arr[:, i] = sum of all the data from column i

$\sum$(errorloss arr[:, i]!=0)= sum of non-zero elements of Column i.

10: **End**

Second, our model learns from training data that includes the normal reconstruction error rate of CO_2 points.

To begin the training phase, the original data set needs to be transformed into time-series sequences. Training data set is represented by data set Xi. In our model, each sequence consists of ten CO_2samples on ten-time steps. The model's training i.e., LSTM-AE training begins by inputting each sequence to the encoder. During this process, a single LSTM is trained sequentially on each sample in the sequence. when each sequence training is finished, the encoder's latent space rearranges the combined data points into an encoded feature representation in 1-D. The encoded feature is duplicated using the Repeat Vector layer to create multiple copies.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

The decoder uses LSTM cells, where the number of LSTM cells is proportional to the number of time steps and encoded features, to produce the output. Each LSTM cell processes a single encoded feature, resulting in a set of vectors. At the Time Distributed Dense Layer, the output is created as a single-dimensional vector based on what comes out of processing by all LSTM cells.

The steps from 7-9 in algorithm1 show the reconstruction error loss between output and input. The model's weights and parameters are modified using back propagation approach.

The reconstruction error loss employed in this research is the mean absolute error (MAE) method as shown in the equation11.

$$errorloss(MAE) = \frac{\sum_{i=1}^{n} ||x_i - \hat{x}_i||}{n} \quad (11)$$

Where,
n= total number of samples
x_i=original input
$\hat{x}_i$=output produced by the encoder

In order to minimize the reconstruction error loss, the model is trained on all sequence data. The activation function "tanh" was employed in the tenth long short-term memory (LSTM) cell, and its output was captured by a network of 16 neurons in the encoder's latent space. There are two dropout layers (rate = 0.2) in both the encoder and the decoder. A Repeat vector layer is utilized between the encoder and decoder, and a Time Distributed dense layer is used as the final intermediate step before the output layer.

The time series input to the LSTM encoder consists of ten data points, each representing ten time steps. This time, all ten data points span the whole range of CO_2values, which was not the case earlier. The LSTM decoder takes an input sample and outputs a time series consisting of 10 data points at ten-time steps after encoding and decreasing the feature representation of the sample.

For every data point, a reconstruction error rate is computed and subsequently compared to a threshold value (Thr). The calculation of the reconstruction loss follows the same methodology as described during the training process.. A data point is regarded as an anomaly if the value of the resulting reconstruction loss is greater than a predetermined threshold (Thr), and normal otherwise. Equation 12 illustrates this classification procedure.

$$X' = \begin{cases} X' is\ anomaly, & if test_{arr}[i] > Thr \\ X' is\ normal, & otherwise \end{cases} \quad (12)$$

Here Thr=$\mu + 2\sigma$ and μ is nothing but reconstruction loss or errorloss[i] in Algorithm1. If any test data test_{arr}[i]) >Thr, then it is called as anomaly.

D. Hardware and software components

1) Hardware components:

SCD30, CO_2 sensor, BME 680 IAQ sensor, ESP 8266 Wi-fi module and Raspberry Pi 3B single-board are the hardware components used in this study.

a) SDC30 CO_2 sensor:

The SCD30 is a very precise nondispersive infrared (NDIR) sensor for measuring CO_2 characteristics.

b) BME 680 IAQ sensor:

The BME680 sensor is a popular environmental sensor that is capable of measuring multiple parameters such as temperature, humidity, pressure, and volatile organic compounds (VOCs) in the air. It is manufactured by Bosch Sensortec and is often used in various applications including indoor air quality monitoring, weather stations, and Internet of Things (IoT) devices.

c) ESP 8266:

The ESP 8266 is a widely used and popular Wi-Fi module that is designed for Internet of Things (IoT) applications. It is manufactured by Espressif Systems and has gained popularity due to its low cost, ease of use, and extensive community support. The ESP8266 module is capable of providing wireless connectivity to micro controllers or other embedded systems, enabling them to connect to Wi-Fi networks and communicate with other devices or servers.

2) Software:

a) Message Queuing Telemetry Transport (MQTT):

It is a reliable machine-to-machine (M2M) protocol designed for Internet of Things applications. TT follows a client-server model, where MQTT clients connect to an MQTT broker to exchange messages. Clients can be sensors, devices, or applications, while the broker acts as a message broker or intermediary that routes messages between clients. MQTT uses a publish-subscribe model, where clients can publish messages to topics and subscribe to receive messages from topics of interest.

b) Node-Red:

It is an open-source utility for connecting hardware devices, APIs, and other IoT services using a flow-based architecture. The MQTT node of Node-Red can subscribe to MQTT and cloud data and store it in a database. Using the powers of visual programming, Node-Red can identify system faults such as sensor failure.

c) Influxdb:

InfluxDB, developed by the Influx Data organization, is a time series database chosen for this design due to its compatibility with the Node-RED tool and its capability to handle timestamped data collection.

d) Grafana:

It is a dashboard that helps you exhibit our data and a tool for viewing and analysing time series.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

All the simulations are carried out using matlab.it is possible to load data from an InfluxDB database into MATLAB and then display it on a Grafana dashboard. Here's a general outline of the process:

- Connect to the InfluxDB database in MATLAB: MATLAB provides the "Database Toolbox" that supports connecting to various databases, including InfluxDB. We can use the database function to establish a connection to our InfluxDB database.
- Query data from InfluxDB: Once connected, we can use SQL-like queries (InfluxQL) or the InfluxDB Query Language (Flux) to retrieve data from our database. MATLAB provides functions such as fetch or fetch2 to execute queries and retrieve the data as a MATLAB table or structure.
- Manipulate and analyze data in MATLAB: MATLAB offers a wide range of data analysis and manipulation capabilities. We can process the retrieved data, perform calculations, apply algorithms, and generate visualizations using MATLAB's built-in functions or toolboxes.
- Export data to Grafana: Once We have processed and prepared our data in MATLAB, We can export it to Grafana. Grafana supports various data sources, including InfluxDB. We will need to set up a data source in Grafana and configure it to connect to our InfluxDB database. Then, we can create a new dashboard in Grafana and use the data source to populate the visualizations with the data we exported from MATLAB.

Both sensors are linked to Wi-Fi modules, as shown in Figure 6 of the system architecture for this design. Using the Arduino programming language, the sensors and Wi-Fi modules are configured to communicate to the Wi-Fi gateway. Due to its flexible and potent processing performance, Raspberry Pi is used as a hub that communicates with IoT applications[24]. Node-Red, influxdb, and Grafana are installed on the Raspberry Pi to retrieve, display, and analyze data from the MQTT server.

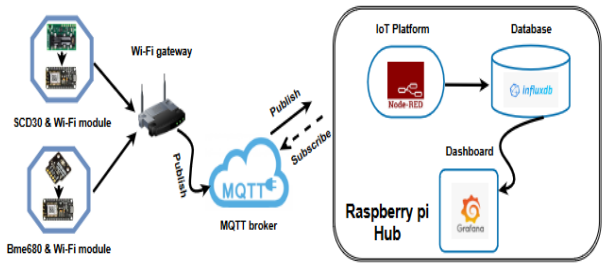

Figure 6. Architecture of deployed IAQ monitoring system

E. CO_2 Occupancy data set

To investigate the connection between CO_2 levels, weather conditions, and student performance, we deployed two sensors, the BME680 and SCD30, to measure the level of CO_2, temperature, humidity, and CO_2, in different classrooms of our VIT-AP University. The sensors were placed on a wall 1.5 meters above the floor. We collected CO_2 readings for four months between February 1st, 2022, and May 30th, 2022, with a 1-minute interval. The CO_2 readings are displayed in Fig7. As expected, we observed no changes in CO_2 levels during University breaks such as in May 2022. However, when students were present in the classrooms, we observed fluctuations in CO_2 levels, some of which were anomalous. The data set contained a total of 2,10,105 CO_2 readings.

1) Data preprocessing

To ensure data cleanliness, we performed several actions on the original records. Firstly, we removed any duplicate records which had identical timestamps and CO_2 readings. Additionally, we removed records containing both NaN(Not a number) values for the CO_2 reading and timestamp. Next, we kept data where no CO_2 value was recorded, but the time stamp was correct even though the value was NaN. For these records, we replaced the empty or NaN values with the numerical value of 0. The value 0 was substituted for any empty or NaNs in these records. Following this cleaning process, we had 1,60,008 records remaining

2) Training and test data set

a) Training data set

According to [25], the normal range for CO_2 levels typically falls between 0 and 968 parts per million (PPM). Using the 2-sigma rule of the normal distribution (i.e., around CO_2 readings 968 when the mean of the CO_2readings is about 488), we found that the vast majority of the readings fell within the permissible normal range (as shown in Fig.8). To ensure that our model was only trained on CO_2 data points within this normal range, we separated the first three months of the original data set (from 02/02/2022 to 31/05/2022). We then used the 2-sigma rule to identify any samples that did not fall within the normal range, and removed them. This process of creating the training data set is illustrated in figure. 7. By doing so, we aimed to train our model only on the CO_2 data points that were considered normal.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

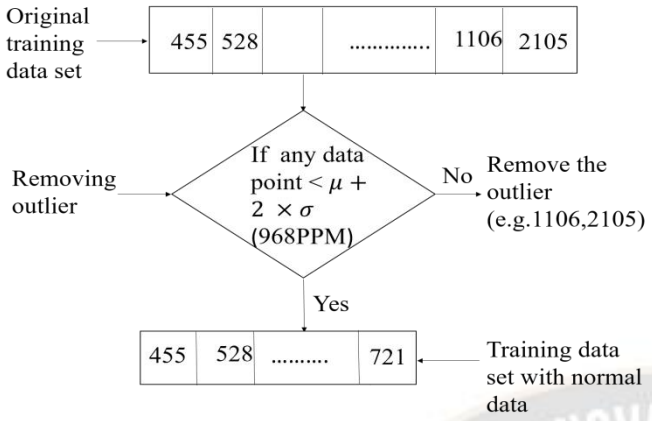

Figure 7. Training data set creation

b) Test data set

For our testing data set, we utilized one month of the original data, specifically from 01/05/2022 to 31/05/2022. This data set included a range of CO_2 readings, both normal and anomalous. To evaluate the performance of our model in detecting anomalous data points from normal ones, we added a label of 0 if the CO_2 reading fell within the normal range(968ppm). Conversely, we added a label of 1 if the CO_2 reading was outside of this range. These labels were only utilized for model evaluation purposes. Figure 8 illustrates how we added these labels in the testing data set.

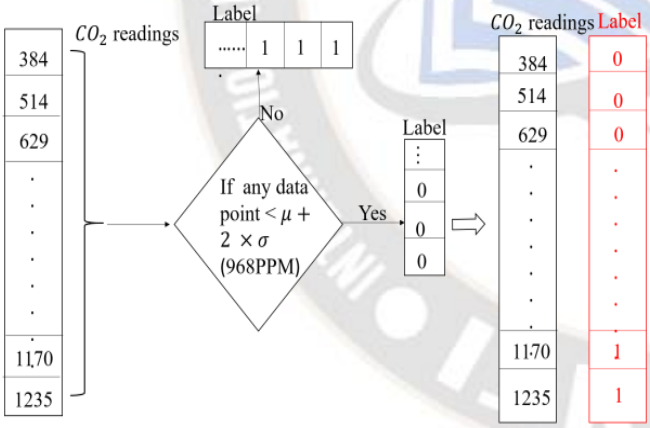

Figure 8. Test data set creation

c) Data normalization

We used a data normalization technique to cut down on model training's execution time and computational complexity to counteract the influence of varying scales across CO_2

readings. Specifically, we utilized a standard scalar normalization technique, which is represented by equation 13:

$$Z_i = \frac{x_i - \mu}{\sigma} \quad (13)$$

In the above equation, Z_i represents the normalized numerical values within the range of 0 to 1.

Where,

x_i =Specific data point

μ= mean

σ=standard deviation

TABLE I. LSTM-AE SIMULATION HYPER PARAMETERS

Hyper parameter	value
Dropout	0.2
Learning rate	0.001
Batch size	64
Epoch	30

VI. EXPERIMENTAL SETUP

The experiments were conducted utilizing the system configuration described in Table I. In order to assess how well our model performed, we employed various performance measures, including the F1- score, precision, recall, and classification accuracy. The corresponding confusion matrix can be found in Table II

TABLE II. CONFUSION MATRIX

Total Samples		Predicted Status	
		Normal	*Anomaly*
Actual status	Normal	TN	FP
	Anomaly Predicted Status	FN	TP

- When a data point is correctly detected as anomalous, it is referred to as a true positive (TP).
- If a normal data point is accurately detected as normal, it is considered a true negative (TN).
- A false positive (FP) occurs when a normal data point is incorrectly detected as anomalous.
- When an anomalous data point is incorrectly detected as normal, it is referred to as a false negative (FN).

Using the terms mentioned earlier, the performance metrics are computed in the following manner:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (14)$$

$$Precision = \frac{TP}{TP+FP} \quad (15)$$

$$Recall = \frac{TP}{TP+FN} \quad (16)$$

$$F1-score = 2 \times \left(\frac{precision \times recall}{precision+recall}\right) \quad (17)$$

Figure 9 displays how the loss varies across different epochs. The blue line represents the training loss, which gauges the model's error rate during training. It appears that the training loss

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

stabilizes rather rapidly, at around eight epochs. To evaluate our model's performance during training, we reserved 10% of the training data set as the validation set. As expected, the validation loss does not stabilize until after eight epochs. However, once it reaches this point, the validation loss exhibits a loss rate similar to that of the training loss, which is approximately 0.07%. This suggests that our model is well-suited and performs well, as it does not underfit or overfit

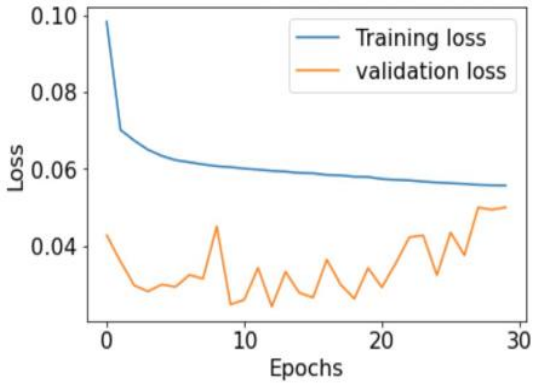

Figure 9. Loss variation across different epochs

We ran experiments to see how our model responded to changes in its architecture, specifically the number of hidden layers and LSTM cell units. Encoder and decoder models with one hidden layer, two hidden layers, and an undefined number of hidden layers (referred to as 'n') were evaluated. While all models kept the same total number of LSTM units, they varied in the number of hidden layers positioned between the encoder and decoder. There were some subtle changes in the model's performance as a function of the number of hidden layers. An F1-score exceeding 95.02% was achieved by utilizing a model architecture comprising only one hidden layer as an example.. Comparatively, the F1-scores for the two-layer and three-layer models were 94.58% and 94.41%. The performance of the model was highly sensitive to the output vector size by the various designs. As can be seen in Table III, the best F1-score (95.02%) was attained by the simplest one-layer model whose output vector size was also the shortest.

TABLE III. DIFFERENT ARCHITECTURES PERFORMANCES

Number of layers	Number of units	F1-score	recall	precision	Accuracy
1	128	93.31	86.01	100	99.30
1	**16**	**99.49**	**90.01**	**100**	**99.49**
2	64, 16	99.41	88.77	100	99.41
3	128,64,16	99.40	88.48	100	99.40

Table IV illustrates the results obtained from various model parameters, specifically the impact of different learning rates and batch sizes on performance. The experiment kept the dropout and epoch values constant throughout.

A batch size of 64 led to the most favorable outcomes, resulting in the highest F1-score (95.02%), accuracy (99.49%), and recall (90.01%). In contrast, the smallest batch size of 10 demonstrated the lowest F1-score (93.67%), accuracy (99.42%), and recall (88.08%). This performance discrepancy can be attributed to the larger volume of data processed when using the larger batch size. While the smaller batch size led to a decrease in overall detection accuracy, it offered the advantage of faster response times. For instance, when compared to a batch size of 64, running the proposed model with a batch size of 10 resulted in a response time three times faster.

Likewise, the highest learning rate of 0.001 demonstrated the most favorable performance, whereas decreasing the learning rate led to a degradation in performance, resulting in a decrease in accuracy from 99.49% with the learning rate of 0.001 to 99.39% with the learning rate of 0.00001. The impact on response time, however, was minimal and showed insignificant variations in relation to the learning rate.

The number of time steps in a sequence, which is determined by the size of the time-sliding window, can have an effect on our model's performance. This choice has an impact on how the reconstruction error rate is calculated. Therefore, we conducted tests to examine the sensitivity of our model to variations in the size of the time-sliding window and its subsequent effect on performance.

We tested our model with time sliding windows of various sizes, including 10, 15, 20, 25, 30, 35, and 40. The smallest window size of 10 demonstrated better performance, achieving the higher True Positive Rate (TPR) and F1-score at 95.02%. as shown in figure 10. On the other hand, the time sliding window size of 15 exhibited the lowest F1-score (90.12%) and accuracy (99.20%). The smaller window sizes, such as 15 and 20, resulted in increased misclassifications, with a TPR just above 80%. Conversely, larger window sizes above 20 demonstrated fewer misclassifications and improved TPR rates and F1-scores.

The total number of test samples was 43,787, consisting of 41,697 normal samples and 2,200 abnormal samples as per our labeling. Our model accurately identified 1,991 anomalies out of the 2,200 anomaly points, resulting in an accuracy rate of 90.01%. It correctly identified all 41,697 normal data points, achieving a 100% accuracy rate. Notably, our model did not produce any false positives, incorrectly classifying normal samples as abnormal, which means it correctly classified all normal samples as normal. but it generated 212 false negatives by misclassifying abnormal samples as normal. Overall, our model achieved an accuracy of 99.49%.

The precision of our model was 100%, the recall was 90.01%, and the F1-score was 95.02%.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

Table V presents a performance evaluation of our model, on the CO_2 occupancy data set, with other models utilizing different variations of LSTM-AE. Our approach demonstrates superior performance in terms of accuracy (99.49%) and precision (100%). Among the comparable models, the one proposed by Yin et al. [26] exhibits competitive performance, with comparable accuracy and F1-score. Additionally, Nguyen et al. [27] introduced a model with a higher F1-score (96.98%), albeit at a slightly lower accuracy when compared to our method. Further investigation reveals that they employed a one-class support vector machine (SVM) as an additional classifier to mitigate false positives.

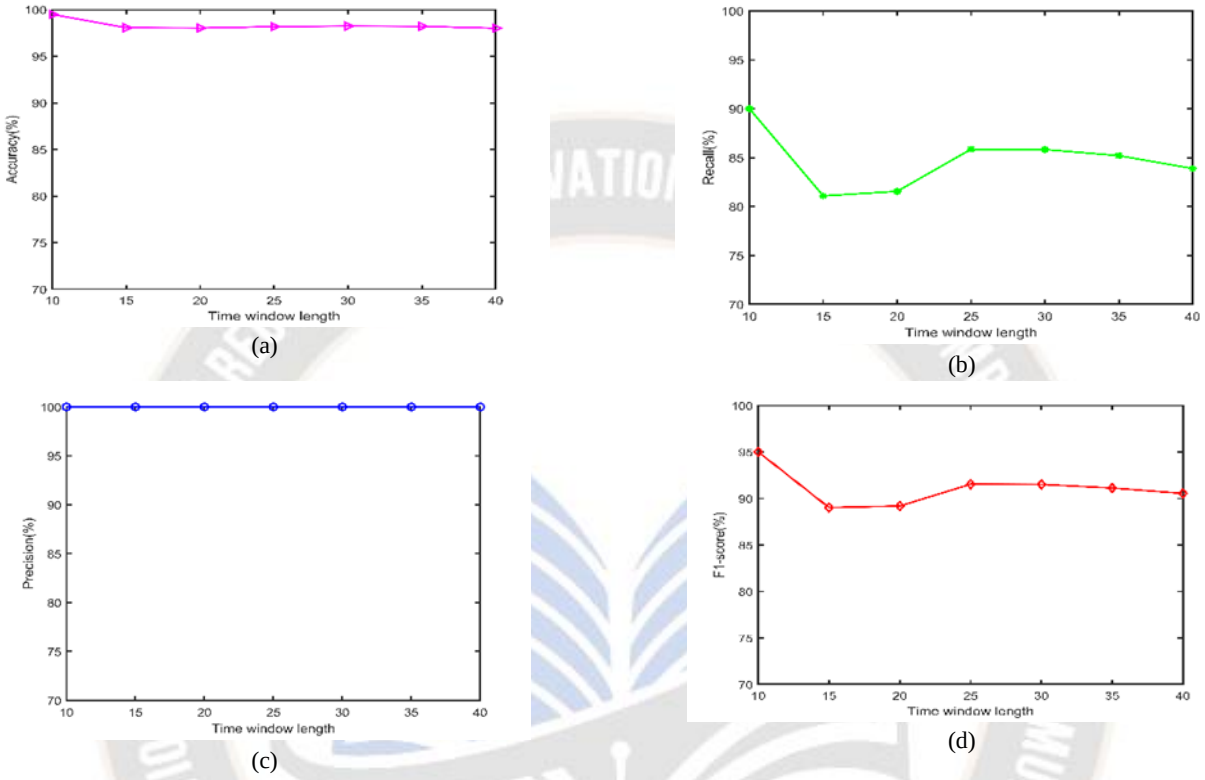

Figure 10. LSTM-AE performance under various time window length.(a)accuracy.(b)precision. (c)recall. (d)F1 –score

TABLE IV. PARAMETERS COMPARISON

	Parameters	Performances				
Learning rate:0.001	***Batch size***	***F1-Score***	***Recall***	***Precision***	***Accuracy***	***Time(s)/epoch***
	64	**95.02**	**90.01**	**100**	**99.49**	**35.69±1.29**
	32	94.2	89.03	100	99.44	45.18±3.41
	10	93.67	88.08	100	99.42	99.70±1.52
Batch size:64	***Learning rate***	***F1-Score***	***Recall***	***Precision***	***Accuracy***	***Time(s)/epoch***
	0.00001	93.56	86.98	100	99.38	35.84±1.14
	0.0001	93.91	88.53	100	99.42	35.79±1.29
	0.001	**95.02**	**90.01**	**100**	**99.49**	**35.69±1.29**

TABLE V. PERFORMANCE COMPARISON

Model	Method	F1-score	Recall	Precision	Accuracy
Kang et al.[28]	LSTM-AE	91.45	85.77	97.94	94.44
Nguyen et al.[27]	LSTM-AE-OCSVM	96.98	99.59	98.45	98.36
Liu et al.[29]	LSTM-AE	-	97.55	97.55	98.57
Yin et al.[26]	LSTM-AE	95.97	94.16	97.84	99.25
Our proposed method	**LSTM-AE**	**95.02**	**90.01**	**100**	**99.49**

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

VII. CONCLUSION

Our proposed approach utilizes a hybrid deep-learning technique, specifically combining LSTM-AE, to effectively identify contextual anomalies within IAQ data sets. Within our model, the LSTM component focuses on learning the common patterns found in CO_2 time sequence data. Conversely, the AE component reduces data dimensionality and calculates the optimal reconstruction error for each time sequence. By establishing a threshold based on the average reconstruction error observed in normal time-series data, we can detect contextual anomalies that deviate from the expected pattern. Therefore, our approach capitalizes on the complementary Strengths of LSTM and AE for anomaly detection in IAQ data sets. We also have plans to expand our research in various directions. Firstly, we aim to incorporate sensor data from additional sources, such as specific matter, temperature, and relative humidity, to broaden the scope of our present efforts. Additionally, we intend to apply our proposed approach to different application domains to evaluate its generalizability and practicality.

REFERENCES

[1] He, J., Xu, L., Wang, P., Wang, Q., 2017. A high precise e-nose for daily indoor air quality monitoring in living environment. Integration 58, 286–294.

[2] Motlagh, N.H., Zaidan, M.A., Lagerspetz, E., Varjonen, S., Toivonen, J., Mineraud, J., Rebeiro-Hargrave, A., Siekkinen, M., Hussein, T., Nurmi, P., et al., 2019. Indoor air quality monitoring using infrastructure-based motion detectors, in: 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), IEEE. pp. 902– 907.

[3] Wargocki, P., Porras-Salazar, J.A., Contreras-Espinoza, S., Bahnfleth, W., 2020. The relationships between classroom air quality and children's performance in school. Building and Environment 173, 106749.

[4] Maduranga, M., Kosgahakumbura, K., Karunarathna, G., 2020. Design of an iot based indoor air quality monitoring system

[5] Peng, Z., Jimenez, J.L., 2021. Exhaled co2 as a covid-19 infection risk proxy for different indoor environments and activities. Environmental Science & Technology Letters 8, 392–397.

[6] Zeng, Y., Chen, J., Jin, N., Jin, X., Du, Y., 2022. Air quality forecasting with hybrid lstm and extended stationary wavelet transform. Building and Environment 213, 108822.

[7] Franco, A., Leccese, F., 2020. Measurement of co2 concentration for occupancy estimation in educational buildings with energy efficiency purposes. Journal of Building Engineering 32, 101714.

[8] Zhu, Y., Al-Ahmed, S.A., Shakir, M.Z., Olszewska, J.I., 2022. Lstm based iot-enabled co2 steady-state forecasting for indoor air quality monitoring. Electronics 12, 107.

[9] Wei, Y., Jang-Jaccard, J., Sabrina, F., Alavizadeh, H., 2020. Largescale outlier detection for low-cost pm 1.0 sensors. IEEE Access 8, 229033–229042.

[10] Kumari, S., Singh, S.K., 2022. Machine learning-based time series models for effective co2 emission prediction in india. Environmental Science and Pollution Research , 1–16.

[11] Kallio, J., Tervonen, J., Räsänen, P., Mäkynen, R., Koivusaari, J., Peltola, J., 2021. Forecasting office indoor co2 concentration using machine learning with a one-year dataset. Building and Environment 187, 107409.

[12] Tekler, Z.D., Ono, E., Peng, Y., Zhan, S., Lasternas, B., Chong, A., 2022. Robod, room-level occupancy and building operation dataset, in: Building Simulation, Springer. pp. 2127–2137.

[13] Tekler, Z.D., Ono, E., Peng, Y., Zhan, S., Lasternas, B., Chong, A., 2022. Robod, room-level occupancy and building operation dataset, in: Building Simulation, Springer. pp. 2127–2137.

[14] Christopher Davies, Matthew Martinez, Catalina Fernández, Ana Flores, Anders Pedersen. Applying Recommender Systems in Educational Platforms. Kuwait Journal of Machine Learning, 2(1). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/171

[15] Ahn, J., Shin, D., Kim, K., Yang, J., 2017. Indoor air quality analysis using deep learning with sensor data. Sensors 17, 2476.

[16] Tagliabue, L.C., Cecconi, F.R., Rinaldi, S., Ciribini, A.L.C., 2021. Data driven indoor air quality prediction in educational facilities based on iot network. Energy and Buildings 236, 110782.

[17] Wambura, S., Huang, J., Li, H., 2020. Long-range forecasting in feature-evolving data streams. Knowledge-Based Systems 206, 106405.

[18] Sharma, P.K., Mondal, A., Jaiswal, S., Saha, M., Nandi, S., De, T., Saha, S., 2021. Indoairsense: A framework for indoor air quality estimation and forecasting. Atmospheric Pollution Research 12, 10– 22.

[19] Xu, C., Chen, H., Wang, J., Guo, Y., Yuan, Y., 2019. Improving prediction performance for indoor temperature in public buildings based on a novel deep learning method. Building and Environment 148, 128–135.

[20] Jung, Y., Kang, T., Chun, C., 2021. Anomaly analysis on indoor office spaces for facility management using deep learning methods. Journal of Building Engineering 43, 103139.

[21] Hossain, E., Shariff, M.A.U., Hossain, M.S., Andersson, K., 2020. A novel deep learning approach to predict air quality index, in: Proceedings of International Conference on Trends in Computational and Cognitive Engineering: Proceedings of TCCE 2020, Springer. pp. 367–381.

[22] Ottosen, T.B., Kumar, P., 2019. Outlier detection and gap filling methodologies for low-cost air quality measurements. Environmental Science: Processes & Impacts 21, 701–713.

[23] Li, J., Izakian, H., Pedrycz, W., Jamal, I., 2021. Clustering-based anomaly detection in multivariate time series data. Applied Soft Computing 100, 106919.

[24] Zusman, M., Gassett, A.J., Kirwa, K., Barr, R.G., Cooper, C.B., Han, M.K., Kanner, R.E., Koehler, K., Ortega, V.E., Paine 3rd, R., et al., 2021. Modeling residential indoor concentrations of pm2. 5, no2, nox, and secondhand smoke in the subpopulations

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7618
***Article Received**: 01 June 2023 **Revised**: 27 July 2023 **Accepted**: 11 August 2023*

and intermediate outcome measures in copd (spiromics) air study. Indoor air 31, 702– 716

[25] Jolles, J.W., 2021. Broad-scale applications of the raspberry pi: A review and guide for biologists. Methods in Ecology and Evolution 12, 1562–1579.

[26] Liu, Y., Pang, Z., Karlsson, M., Gong, S., 2020. Anomaly detection based on machine learning in iot-based vertical plant wall for indoor climate control. Building and Environment 183, 107212.

[27] Dehaq E. Mohsen, Ehsan M. Abbas, Maan M. Abdulwahid. (2023). Performance Analysis of OWC System based (S-2-S) Connection with Different Modulation Encoding. International Journal of Intelligent Systems and Applications in Engineering, 11(4s), 400–408. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2679

[28] Yin, C., Zhang, S., Wang, J., Xiong, N.N., 2020. Anomaly detection based on convolutional recurrent autoencoder for iot time series. IEEE Transactions on Systems, Man, and Cybernetics: Systems 52, 112– 122.

[29] Nguyen, H.D., Tran, K.P., Thomassey, S., Hamad, M., 2021. Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. International Journal of Information Management 57, 102282.

[30] Kang, J., Kim, C.S., Kang, J.W., Gwak, J., 2021. Anomaly detection of the brake operating unit on metro vehicles using a one-class lstm autoencoder. Applied Sciences 11, 9290.

[31] Liu, P., Sun, X., Han, Y., He, Z., Zhang, W., Wu, C., 2022. Arrhythmia classification of lstm autoencoder based on time series anomaly detection. Biomedical Signal Processing and Control 71, 103228.