Scientific Journal of Informatics
Vol. 13, No. 2, May 2026
p-ISSN 2407-7658 https://journal.unnes.ac.id/journals/sji/index e-ISSN 2460-0040

Performance Evaluation of Distributed Lag, Autoencoder, and LSTM Autoencoder Methods in Detecting Anomalies in Simulated Data Based on Air Quality Index in Jakarta

Yenni Angraini[1*], Adelia Putri Pangestika[2], I Made Sumertajaya[3]
[1, 2, 3]Statistics and Data Science Study Program, School of Data Science, Mathematics, and Informatics, IPB University, Indonesia

Abstract.
Purpose: This study evaluates the performance of distributed lag, autoencoder, and LSTM autoencoder methods in detecting point anomalies in simulated data generated from Jakarta's Air Quality Index (AQI). The evaluation is conducted across several simulation scenarios that represent factors that may influence anomaly detection performance.
Methods: Simulation scenarios were constructed by varying two anomaly characteristics: anomaly percentage (0.3%, 0.5%, and 1.0%) and anomaly depth (4.4σ, 4.7σ, and 5.0σ), yielding 90 datasets generated via repeated experiments. Anomaly detection was performed using a forecasting-based approach with a 4σ threshold on prediction errors. Model performance was evaluated using mean absolute percentage error (MAPE) for forecasting accuracy and balanced accuracy for anomaly detection.
Result: Increasing anomaly percentage significantly degrades both forecasting and anomaly detection performance across all methods. In contrast, anomaly depth has no significant effect on forecasting accuracy but strongly influences detection performance. Among the evaluated methods, the distributed lag model consistently shows the most robust anomaly detection performance across scenarios, outperforming the autoencoder and LSTM autoencoder, particularly at higher anomaly percentages and depths.
Novelty: This study introduces a more rigorous and structured evaluation framework for anomaly detection by integrating three key contributions. First, it provides a unified comparison of distributed lag, autoencoder, and LSTM autoencoder methods within a single experimental setting, a limitation in prior studies. Second, it employs a controlled simulation design that systematically varies the anomaly percentage and depth while preserving key characteristics of empirical AQI data, enabling a more objective assessment of model performance across diverse anomaly conditions. Third, it uses ANOVA and interaction analysis to formally examine the effects of anomaly characteristics on both forecasting and detection performance, moving beyond purely descriptive comparisons commonly used in previous research.

Keywords: Air Quality Index, Anomaly Detection, Autoencoder, Distributed Lag, LSTM Autoencoder
Received March 2026 / **Revised** April 2026 / **Accepted** April 2026

INTRODUCTION

An anomaly is an observation that deviates considerably from the dominant pattern of the data, suggesting the presence of a different mechanism underlying its generation. In many studies, anomalies are often treated as interference to be eliminated, but in specific contexts, they can be important information that requires further detection and analysis. Anomaly detection can be performed using various approaches, one of which is a statistical method-based approach to time series data. This approach has several advantages, including a clear model foundation [1] and relatively low computational complexity [2]. However, statistical methods generally have limitations in capturing nonlinear patterns [3], are less effective in modelling data with high complexity [4], and tend to be limited to low-dimensional data [5]. One statistical approach with potential applicability in forecasting-based anomaly detection is the distributed lag method.

Apart from statistical methods, deep learning-based methods have increasingly been utilized for anomaly detection. These methods are known to possess strong capabilities for modeling nonlinear relationships in high-dimensional data [6], do not require explicit parametric model specification [7], and are more

*Corresponding author.
Email addresses y_angraini@apps.ipb.ac.id (Anggraini)*, adelia_pangestika@apps.ipb.ac.id (Pangestika), imsjaya@apps.ipb.ac.id (Sumertajaya)
DOI: 10.15294/sji.v13i2.45581

adaptable to complex data dynamics [8]. However, deep learning methods generally require relatively longer computation times [9]. Despite this limitation, several approaches, such as autoencoders and long short-term memory (LSTM) autoencoders, are widely used for anomaly detection.

Anomaly detection is a crucial problem across many fields, including air quality monitoring. Air pollution is a significant global problem, contributing to approximately 8.8 million deaths worldwide [10] and triggering various respiratory diseases, such as acute respiratory infections (ARI). In Jakarta, the number of ARI cases increased from 147,439 in 2022 to 638,291 in 2023 [11]. This surge in cases indicates an unusual condition in air quality, making anomaly detection in the air quality index crucial as a basis for policy formulation and an early warning system for potential future air pollution. Since air quality index data are structured as a time series, they exhibit temporal dependence, with past values influencing current observations. As a result, anomaly detection should be conducted using time-series-based approaches that can capture this temporal relationship, such as statistical methods, distributed lag models, or deep learning techniques, including autoencoders and LSTM autoencoders.

The distributed lag method has not been widely applied directly for identifying anomalies in temporal data. However, this method exhibits quite good forecasting performance, so residual-based anomaly detection remains relevant for the application. [12] demonstrated that the distributed lag method yielded good forecasting accuracy for COVID-19 mortality data in 39 European countries, with most median absolute percentage errors remaining below 20%. On the other hand, the autoencoder method has been successfully employed by [13] to detect anomalies in laundry equipment power consumption data, demonstrating better performance than isolation forests. Furthermore, LSTM autoencoders have proven effective for detecting anomalies in software network data, achieving superior performance compared to one-class support vector machines [14].

Although autoencoders and LSTM autoencoders have been widely reported to be effective for anomaly detection, studies systematically comparing their performance with statistical approaches, such as the distributed lag model, remain limited, particularly in examining the influence of anomaly characteristics. Such a comparison is theoretically important because these methods are based on fundamentally different modelling paradigms: distributed lag models explicitly capture linear temporal dependencies through lag structures, whereas autoencoder-based methods learn nonlinear representations of data, and LSTM autoencoders are designed to model sequential and long-term temporal dependencies. Therefore, evaluating these approaches within a unified framework allows for a deeper understanding of how different assumptions about temporal structure affect anomaly detection performance.

From a practical perspective, this comparison is highly relevant for air-quality time series, which typically exhibit temporal dependence, seasonal patterns, and potential nonlinear behaviour. Therefore, the use of simulated data is crucial for evaluating the performance of the three methods in a controlled manner, particularly in examining the impact of variations in the percentage and depth of anomalies on detection performance. By systematically varying these anomaly characteristics, this study investigates the sensitivity and robustness of each method under different anomaly conditions, thereby providing more reliable guidance for selecting appropriate models in real-world air quality monitoring applications.

METHODS

Data

The simulation data used in this study were constructed based on empirical air quality index data from Jakarta, serving as the response variable, and calendar variation data as explanatory variables. The data are recorded hourly and span the period from January 1, 2019, to February 29, 2024. The air quality index data were obtained from AirNow (www.airnow.gov), while the calendar variation data consist of dummy variables taking a value of 1 for specific events and 0 otherwise. The specific events represented by these calendar variation variables are described in detail in Table 1.

Table 1. Calendar variation data

Variable	Events
X_1	Weekdays
X_2	National holidays
X_3	Collective leave
X_4	Seven days prior to and seven days following Eid Al-Fitr
X_5	Five days prior to and five days following Eid Al-Fitr
X_6	Three days prior to and three days following Eid Al-Fitr
X_7	Eid Al-Fitr
X_8	Christmas day
X_9	New year day
X_{10}	School holidays
X_{11}	Large scale social restriction
X_{12}	Transitional large scale social restriction
X_{13}	Community activities restrictions enforcement

Given that the empirical air quality index data initially contained missing values, a data imputation step was required to obtain a complete, analyzable time series. The imputed data were subsequently subjected to a series of treatments to construct a simulation dataset with controlled anomaly scenarios. These stages include initial anomaly labelling, anomaly cleaning, and synthetic anomaly injection. The initial anomaly labelling aims to identify indications of abnormal observations in the empirical data, given that explicit labels distinguishing anomalous and non-anomalous observations are not available. This information is then used in the anomaly-cleaning stage to remove observations labelled as anomalies, yielding a baseline dataset that represents normal conditions. These two steps ensure that the simulation data are controlled from the outset, such that all anomalies present in the final dataset originate solely from predefined scenarios. Subsequently, synthetic point anomalies are injected, which also serve as ground truth for evaluating anomaly detection methods. The anomaly injection process considers various combinations of anomaly percentage and anomaly depth. Overall, the sequence of steps, including imputation, initial labelling, cleaning, and anomaly injection, is described in detail as follows.

1. The data imputation process is carried out by considering the characteristics and lengths of missing data segments in the time series. This consideration is important because different missing-data patterns require distinct imputation approaches to preserve the data's temporal structure. The imputation methods used for each type of missing data are as follows:
    a. For missing data with a length of one consecutive period, imputation is performed using linear interpolation. This method estimates the missing value by connecting the nearest observed points before and after the missing period with a straight line, ensuring that the imputed value reflects temporal proximity. This approach is suitable for very short gaps, as it preserves local continuity without introducing unnecessary patterns.
    b. For missing data with a length of 2–4 consecutive periods, imputation is performed using seasonal decomposition approaches, namely seadec and seasplit. The Seadec method decomposes the time series into trend, seasonal, and residual components, imputes each component separately, and then recombines them. Meanwhile, seasplit separates the data by seasonal patterns, allowing imputation within subseries with similar seasonal characteristics. These approaches are more suitable for medium-length missing segments, as they better preserve seasonal patterns that simple interpolation cannot adequately capture.
    c. For missing data with a length of five or more consecutive periods, imputation is conducted using daily air quality index data obtained from AQICN. Since the daily data also contain missing values, they are first imputed using a Long Short-Term Memory (LSTM) method. The process begins by building a model using historical data before the missing period. The trained model is then used to predict the first missing value, and this predicted value is subsequently treated as actual input to iteratively predict the following missing values until all gaps are filled. In its implementation, the model architecture consists of two LSTM layers and one dense layer, each with 50 neurons. The training parameters follow previous studies: 50 epochs, a learning rate of 0.01, a batch size of 72, no dropout, and the Adam optimiser. After imputing all missing values in the daily data, the data are disaggregated back into hourly resolution. This disaggregation process considers the characteristics of the day and the average hourly air quality patterns to ensure that the reconstructed hourly distribution remains temporally realistic.

It is important to note that the imputation procedure may affect both forecasting accuracy and anomaly detection performance, as imputed values directly influence the underlying temporal structure of the data. However, in the empirical dataset used in this study, the true values of the missing observations are inherently unobservable, making it impossible to assess the accuracy of the imputation results directly. Consequently, no definitive ground truth is available for evaluating whether the imputed values precisely reflect the actual conditions. Nevertheless, the imputation procedure is applied consistently across all datasets and simulation scenarios. This uniform application ensures that any potential bias or uncertainty introduced by the imputation process is systematically shared across all experimental settings. Therefore, the impact of imputation is expected to be comparable and controlled, and thus does not compromise the validity of the comparative analysis conducted in this study.

2. The anomaly labelling process defines anomalies in the air quality index as observations exhibiting sudden increases or decreases. The identification of anomalous observations is based on Equation 1.

$$d = y_t - y_{t-1} \tag{1}$$

Let d denote the hourly change in the air quality index, calculated as the difference between the values at time t (y_t) and the preceding hour $t-1$ (y_{t-1}). The determination of anomaly threshold in this study follows the widely used sigma rule [15],[16],[17]. In this study, an observation is identified as an anomaly when the value of the difference falls outside the interval $\bar{d} \pm 4\sigma_d$, where $\bar{d}$ represents the mean of the differences and σ_d denotes the standard deviation. Although the 3σ rule is commonly applied, the 4σ rule is adopted to ensure that only truly extreme anomalies—those markedly different from typical observations—are identified. This choice is considered appropriate for air quality index data, which tend to exhibit high variability and short-term fluctuations due to environmental and anthropogenic factors. Under such conditions, less stringent thresholds, such as the 3σ rule or IQR, may not adequately distinguish extreme deviations from normal variations. Therefore, the 4σ rule provides a more robust and defensible criterion for the initial labelling of anomalies, ensuring that the constructed actual labels better represent genuinely extreme observations.

3. The anomaly cleaning process is performed using a winsorization approach, as follows.
   a. Substitute the difference value (d) based on Equation 2.

$$d' = \begin{cases} \bar{d} + 4\sigma_d, jika\ d > 0 \\ \bar{d} - 4\sigma_d, jika\ d < 0 \end{cases} \tag{2}$$

   Here, d denotes the difference in the air quality index before replacement, while d' represents the difference after the replacement process.
   b. Substitute the air quality index value based on Equation 3.

$$y_t' = d' + y_{t-1} \tag{3}$$

   Where y_t' is the air quality index value after replacement and y_{t-1} is the air quality index value at hour $(t-1)$.
   c. Repeat steps 3a and 3b until it is confirmed that there are no difference values that exceed the specified limits.

Table 2. Combination of simulation data scenarios

Number of scenarios	Anomaly percentage	Anomaly depth
1	0.3%	4.4σ
2	0.3%	4.7σ
3	0.3%	5.0σ
4	0.5%	4.4σ
5	0.5%	4.7σ
6	0.5%	5.0σ
7	1.0%	4.4σ
8	1.0%	4.7σ
9	1.0%	5.0σ

4. The anomaly injection process is performed randomly on the test dataset, as training methods such as distributed lag, autoencoder, and LSTM autoencoder require anomaly-free training data [18]. The percentage and depth of the anomalies influence the addition of point anomalies. The anomaly

percentages are 0.3%, 0.5%, and 1%. Meanwhile, the depth of the anomalies consists of 4.4σ, 4.7σ, and 5σ. The combination of the percentage and depth of the anomalies will produce nine scenarios as listed in Table 2.

5. Repeat steps 1 to 4 10 times to produce 90 simulation datasets.

Data Analysis Procedure

Data analysis was performed using R version 4.3.1 and Python version 3.11.11, following the steps outlined below. In R, the analysis utilised the dLagM package for distributed lag modelling and ggplot2 for data visualisation. In Python, the autoencoder and LSTM autoencoder models were implemented using several libraries, including NumPy and Pandas for data manipulation, TensorFlow and Keras for deep learning model development, and scikit-learn for preprocessing tasks such as data normalisation and evaluation.

Table 3. Training and testing split size for each fold in each scenario

Window shift scenario	Fold	Training data size	Testing data size	Unused data size
Semester	1	1 year 2 months	6 months	3 years 6 months
	2	1 year 8 months	6 months	3 years
	3	2 years 2 months	6 months	2 years 6 months
	4	2 years 8 months	6 months	2 years
	5	3 years 2 months	6 months	1 year 6 months
	6	3 years 8 months	6 months	1 year
	7	4 years 2 months	6 months	6 month
	8	4 years 8 months	6 months	-
Quarter	1	1 year 2 months	4 months	3 years 8 months
	2	1 year 6 months	4 months	3 years 4 months
	3	1 year 10 months	4 months	3 years
	4	2 years 2 months	4 months	2 years 8 months
	5	2 years 6 months	4 months	2 years 4 months
	6	2 years 10 months	4 months	2 years
	7	3 years 2 months	4 months	1 year 8 months
	8	3 years 6 months	4 months	1 year 4 months
	9	3 years 10 months	4 months	1 year
	10	4 years 2 months	4 months	8 months
	11	4 years 6 months	4 months	4 months
	12	4 years 10 months	4 months	-
Trimester	1	1 year 2 months	3 months	3 years 9 months
	2	1 year 5 months	3 months	3 years 6 months
	3	1 year 8 months	3 months	3 years 3 months
	4	1 year 11 months	3 months	3 years
	5	2 years 2 months	3 months	2 years 9 months
	6	2 years 5 months	3 months	2 years 6 months
	7	2 years 8 months	3 months	2 years 3 months
	8	2 years 11 months	3 months	2 years
	9	3 years 2 months	3 months	1 year 9 months
	10	3 years 5 months	3 months	1 year 6 months
	11	3 years 8 months	3 months	1 year 3 months
	12	3 years 11 months	3 months	1 year
	13	4 years 2 months	3 months	9 months
	14	4 years 5 months	3 months	6 months
	15	4 years 8 months	3 months	3 months
	16	4 years 11 months	3 months	-

1. The dataset is split into several folds for cross-validation using the expanding-window method. Expanding-window cross-validation is a technique for time series data in which the training set is progressively enlarged, while the test set consists of subsequent, unseen observations [19]. This approach preserves the temporal order of the data and prevents leakage from future observations. In addition, it allows the model to utilise all available historical information, thereby improving model stability [20] and enabling the evaluation of long-term temporal patterns [21]. In this study, the initial training set in the first fold spans 1 year and 2 months. This length is selected to ensure that seasonal patterns are adequately captured and that sufficient observations are available for reliable model training and evaluation. The test set size and window shift interval are kept consistent across folds. Furthermore, several scenarios are considered, namely semester, quarter, and trimester, to evaluate model performance under different forecasting horizons and data accumulation schemes. This design enables a more comprehensive assessment of model robustness and generalizability across varying

temporal settings. An illustration of the expanding-window cross-validation under a semester-based scenario, including the sizes and percentages of the training and test data, is shown in Figure 1. The exact train–test split sizes for each fold, as well as the number of folds in each window-shift scenario, are presented in detail in Table 3.

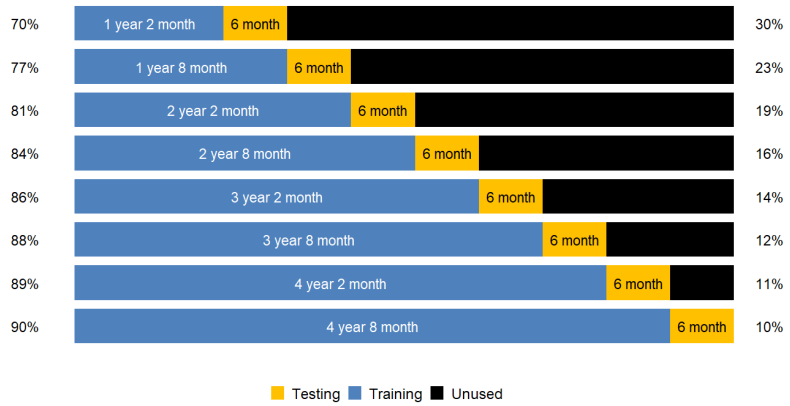

Figure 1. Illustration of semester's window shift scenario

2. Anomaly identification is carried out using the distributed lag method. The distributed lag method models the response variable not only on the current period's independent variables but also on those from the previous period [22]. For each fold and each window-shift scenario, the distributed lag method is applied to detect anomalies, as outlined below.
   a. Perform modelling on training data with a general model that follows Equation 4.

$$\hat{y}_t = \alpha_1 y_{t-1} + \cdots + \alpha_n y_{t-n} + \beta_1 X_{1t} + \cdots + \beta_{13} X_{13t} \quad (4)$$

   In this equation, $\hat{y}_t$ represent the estimated air quality index at hour t, α_n denotes the coefficient associated with the variable y_{t-n} which represents the air quality index observed at hour $(t-n)$. In addition, β_1 refers to the coefficient of the variable X_{1t}, where X_{1t} is the dummy variable X_1 at hour t. The determination of the value of n in this study was carried out by considering the partial autocorrelation function (PACF) plot.
   b. Perform forecasting on one period of test data.
   c. Combine the forecasting results from stage 2b with the training data from stage 2a to create a new, comprehensive training dataset.
   d. Repeat steps 2a to 2c iteratively until all observations in the test set within a window obtain predicted values based on the estimated model parameters.
   e. The error value is calculated using Equation 5.

$$e_i = y_i - \hat{y}_i \quad (5)$$

   Here e denotes the error value, while y_i and $\hat{y}_i$ represent the observed value and the predicted value for the i-th period, respectively.

   f. The forecasting result are evaluated using the mean absolute percentage error (MAPE), as presented in Equation 6.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \quad (6)$$

   Here n represents the total number of data periods.
   g. The determination of anomaly threshold in this study follows the 4σ rule, when an observation is classified as an anomaly if the difference value lies outside the interval $\bar{d} \pm 4\sigma_{pooled}$, with the upper and lower threshold computed using Equation 7 and Equation 8. The 4σ threshold is consistently applied at both the labelling and detection stages to ensure alignment between the anomaly definition, data generation, and the detection process. This approach also ensures a reasonable balance between sensitivity to true anomalies and avoidance of false positives, given the specific characteristics of the dataset.

$$t_u = \bar{e} + 4\sigma_{pooled} \quad (7)$$

$$t_l = \bar{e} - 4\sigma_{pooled} \quad (8)$$

Here t_u and t_l denote the upper and lower bound, respectively. The term $\bar{e}$ represents the mean error, while σ_{pooled} refers to the pooled standard deviation obtained from the sum of the sum squared errors (SSE) divided by total degrees of freedom of error across each fold.

h. Detecting anomalies based on the 4σ rule. An observation is classified as an anomaly when the error value exceeds the upper limit defined in Equation 7 or falls below the lower limit specified in Equation 8.

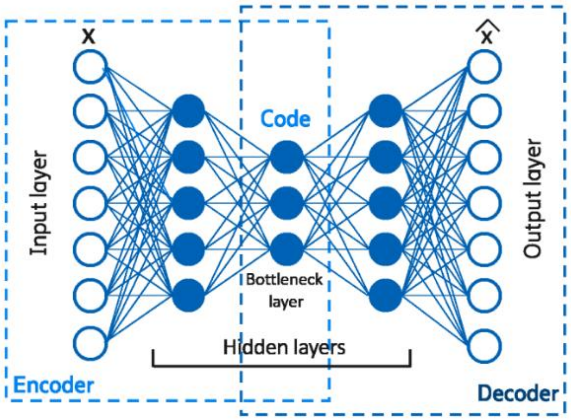

Figure 2. Architecture of autoencoder

3. Detect anomalies using autoencoder and LSTM autoencoder approaches. The autoencoder method is a technique that learns important information from data, allowing it to be used for analysing similar data. This method reduces the input to a simpler dimension (latent space) and reconstructs it back into an output with the original dimension (Figure 2). According to [23], this method is divided into two parts: the encoder part, which is responsible for reducing the input to a latent space based on Equation 9, and the decoder part, which is responsible for reconstructing the latent space back into an output with the original dimension based on Equation 10.

$$\boldsymbol{h} = f_e(\mathbf{W}_e\boldsymbol{x} + \boldsymbol{b}_e) \quad (9)$$

$$\hat{\boldsymbol{x}} = f_d(\mathbf{W}_d\boldsymbol{h} + \boldsymbol{b}_d) \quad (10)$$

Where:

$\boldsymbol{h}$	:	Latent space
f_e	:	Encoder section activation function
$\mathbf{W}_e$	:	Encoder section weighting matrix
$\boldsymbol{x}$	:	Input
$\boldsymbol{b}_e$	:	Encoder section bias vector
$\hat{\boldsymbol{x}}$	:	Output
f_d	:	Decoder section activation function
$\mathbf{W}_d$	:	Decoder section weighting matrix
$\boldsymbol{b}_d$	:	Decoder section bias vector

Meanwhile, the LSTM autoencoder method compresses the input data into a latent representation using an LSTM encoder. Then it reconstructs it to its original dimensionality using an LSTM decoder, as shown in Figure 1. The LSTM units used in both components adopt the conventional LSTM structure, consisting of a forget gate, an input gate, and an output gate, as illustrated in Figure 2. The forget gate determines the extent to which historical information from the previous cell state is retained or discarded based on its relevance [24]. In contrast, the input gate regulates the entry of new information by selecting memory update candidates, ensuring that only important information is stored [25]. Meanwhile, the output gate functions to control the part of the cell state used to form the hidden state

and produce output at time t [26]. Through this mechanism, cell state and hidden state updates are carried out sequentially based on Equation 11-16.

$$\boldsymbol{f}_t = f\left(\mathbf{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f\right) \tag{11}$$

$$\boldsymbol{i}_t = f(\mathbf{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \tag{12}$$

$$\tilde{\boldsymbol{c}}_t = f(\mathbf{W}_c \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_c) \tag{13}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \times \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \times \tilde{\boldsymbol{c}}_t \tag{14}$$

$$\boldsymbol{o}_t = f(\mathbf{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \tag{15}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \times f(\boldsymbol{c}_t) \tag{16}$$

Where:

$\boldsymbol{f}_t$	:	Forget gate using a sigmoid activation function
$\boldsymbol{i}_t$	:	Input gate employing a sigmoid activation function
$\tilde{\boldsymbol{c}}_t$	:	Previous cell state activated by tanh activation function
$\boldsymbol{c}_t$	:	Updateded cell state value
$\boldsymbol{o}_t$	:	Output gate utilizing a tanh activation function
$\boldsymbol{h}_t$	:	New hidden state generated using the tanh activation function
$\boldsymbol{x}_t$	:	Input feature vector
$\boldsymbol{b}$	:	Bias term vector
$\mathbf{W}$	:	Matrix of weights
$\boldsymbol{h}_{t-1}$	:	Hidden state from the previous time step

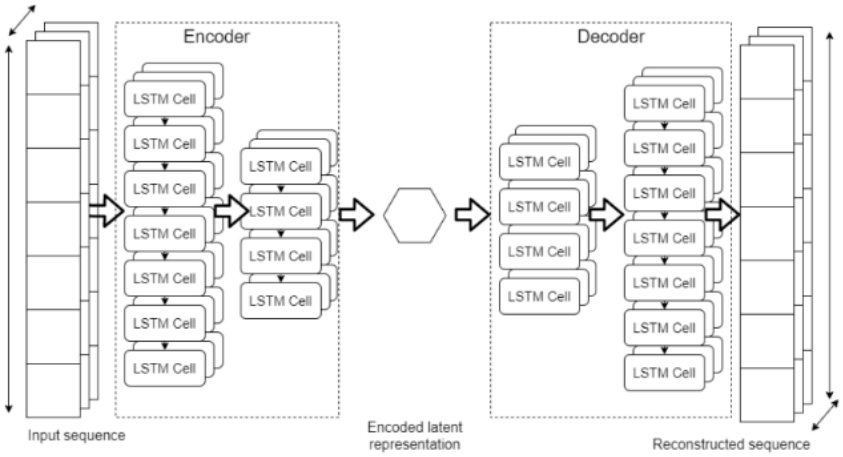

Figure 1. LSTM Autoencoder Architecture

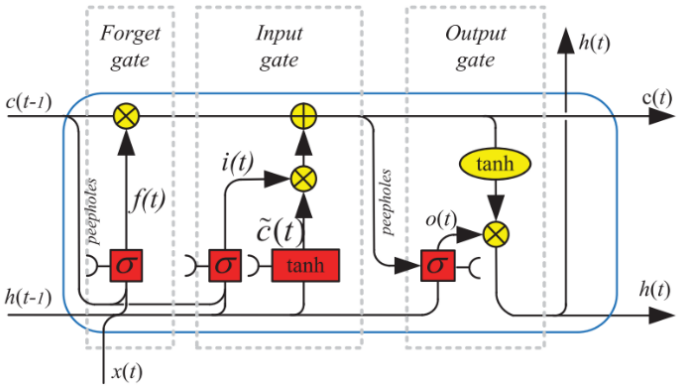

Figure 2. LSTM Architecture

The following step describes the procedure for identifying anomalies using the autoencoder and LSTM autoencoder.

a. Perform data normalisation using the min-max scaler method..
b. Establishing the architecture and determining the hyperparameters. The hyperparameters in this study were determined based on cross-validation results using an expanding-window, shifting-window approach across semesters, quarters, and trimesters. The hyperparameters used in this study included batch size, learning rate, and optimizer settings, as listed in Table 4. These parameters are selected for tuning because they directly influence training dynamics, convergence behaviour, and overall model performance. In contrast, architectural parameters such as latent dimension, number of layers, sequence length, and dropout are fixed to maintain consistency and comparability across all experimental scenarios. The number of training epochs in this study is set to 50 based on preliminary experiments evaluating the models' convergence behaviour. The training and validation losses stabilized before 50 epochs, indicating that additional training would yield only marginal performance improvements. This fixed-epoch setting is also applied consistently across all experimental scenarios to ensure a fair comparison between methods. In addition, the selection considers computational efficiency, particularly given the multiple scenarios evaluated, so that the overall analysis remains tractable without compromising model performance.

Table 4. Tried hyperparameter combinations

Number of hyperparameter combinations	Batch size	Learning rate	Optimizer
1	32	0.001	Adam
2	32	0.001	Rmsprop
3	32	0.01	Adam
4	32	0.01	RMSProp
5	64	0.001	Adam
6	64	0.001	RMSProp
7	64	0.01	Adam
8	64	0.01	RMSProp

c. Defining the input–output structure, reconstruction/prediction mechanism, and forecasting strategy for each model. For the LSTM autoencoder, the input is constructed using a sliding window of 10 consecutive observations, while the prediction target is the subsequent observation (one-step-ahead). The encoder captures temporal dependencies within the sequence. It maps them into a latent representation, which is then processed by the decoder and a final dense layer to produce a single predicted value for the next time step. In contrast, the autoencoder uses a single observation as both input and target, where the encoder–decoder structure learns to reconstruct the input by minimising reconstruction error. Consequently, the autoencoder operates as a pointwise reconstruction model rather than a sequence-based predictor. One-step-ahead forecasts are generated differently across models. In the LSTM autoencoder, predictions are obtained using a rolling-window approach, where the input window is iteratively updated to incorporate the most recent observations. For the autoencoder, predictions are generated independently for each time step based on the reconstructed output.
d. Perform model training on training data.
e. Perform one-step-ahead forecasting on the test data based on the trained models.
f. Calculate the error value based on Equation 5.
g. Evaluate forecasting results based on MAPE values.
h. Calculate the anomaly threshold based on the 4σ rule as stated in Equations 7 and 8.
i. Anomalies are identified using the predetermined thresholds, where an observation is classified as anomalous if its error value exceeds the upper bound defined in Equation 7 or falls below the lower bound defined in Equation 8.

4. Anomaly detection performance is evaluated using the balanced accuracy metric, which reflects the average classification accuracy across both the majority and minority classes. Balanced accuracy is calculated using the values obtained from the confusion matrix, including true positives (anomalies correctly identified as anomalies), true negatives (non-anomalous observations correctly identified as non-anomalous), false positives (non-anomalous observations incorrectly classified as anomalies), and false negatives (anomalies incorrectly classified as non-anomalous). The calculation of the balance accuracy value is stated in Equation 17.

$$Balance\ accuracy = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \quad (17)$$

Here, TP refers to true positives, TN indicates true negatives, FP represents false positives, and FN denotes false negatives.

5. Repeat steps 1 to 4 for each simulation dataset.
6. Conduct an align rank transform analysis of variance (ART ANOVA) to evaluate whether the given scenario affects forecasting performance (as measured by MAPE) and anomaly detection performance (as measured by balanced accuracy). ART ANOVA is employed because it does not require strict assumptions such as normality and homoscedasticity, making it more flexible and suitable for the data characteristics in this study, while still allowing for the assessment of interaction effects. In addition, interaction plots complement the statistical analysis by providing a clear visual representation of how factors jointly influence the response variables. Compared to post-hoc tests, interaction plots offer a more intuitive understanding of interaction patterns across factor levels, particularly in complex experimental designs. Post-hoc comparisons, multiple-comparison adjustments, and effect size measures were not conducted, as the primary objective of the analysis is to identify and interpret main and interaction effects rather than to perform detailed pairwise comparisons. In this context, the interaction plots are sufficient to support the interpretation by clearly illustrating the direction and magnitude of the effects across factor levels.

RESULTS AND DISCUSSIONS

Forecasting Performance of Distributed Lag, Autoencoder, and LSTM Autoencoder Methods

The modelling process plays a crucial role in determining the forecasting performance of the distributed lag, autoencoder, and LSTM autoencoder methods. The modelling process in the distributed lag method is carried out on the training data, beginning with the determination of the number of response-variable lags based on the PACF plot shown in Figure 5. The PACF is used because it captures the direct relationship between the response variable and its lagged values after accounting for intermediate lags, making it appropriate for identifying the relevant lag structure. The significance bounds in Figure 5 are barely visible because they are very close to zero. These bounds are computed as $\pm 1.96/\sqrt{45264}$, which equals approximately ±0.0094 given the large sample size. The lag order is then determined based on the cut-off point in the PACF, namely the last lag that remains statistically significant before the partial autocorrelation diminishes toward zero. Based on this criterion, a lag of three is selected for the model.

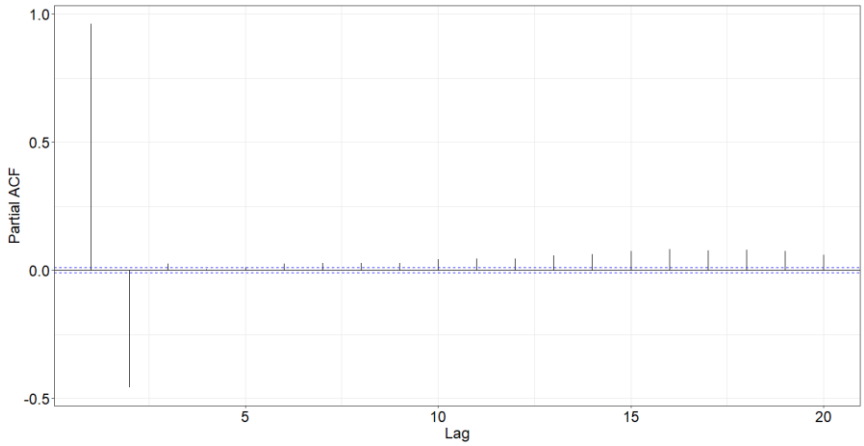

Figure 5. PACF plot of response variable

Subsequently, the model fitted on the training dataset is used to generate forecasts for the test dataset for one period ahead. The predicted values obtained for this period are then incorporated into the training dataset and utilised again in the modelling stage to forecast the test data for the subsequent period. This iterative procedure continues until forecasts are produced for all test periods within the window based on the predetermined model. During the modelling stage, potential multicollinearity among calendar dummy variables is not explicitly restricted. This is because the primary impact of multicollinearity lies in statistical inference, particularly in inflating standard errors and complicating parameter interpretation. In contrast, this study focuses on forecasting and anomaly detection, prioritising predictive accuracy. In this context,

including multiple variables that capture similar calendar effects can be beneficial in representing underlying patterns in the data. Thus, multicollinearity is not treated as a critical issue.

Model diagnostics are still performed; however, they are evaluated selectively with emphasis on aspects that influence forecasting reliability and anomaly detection. In particular, attention is given to residual patterns that may indicate systematic errors or remaining autocorrelation. Assumptions relevant primarily to inference, such as strict normality and homoscedasticity, are not explicitly enforced. Although violations of these assumptions may affect residual-based thresholding and, consequently, anomaly detection performance, their impact is mitigated by using multiple anomaly scenarios with varying percentages and depths, thereby allowing the evaluation to reflect model performance under diverse, potentially non-ideal conditions.

As with the distributed lag approach, the autoencoder-based forecasting procedure starts by fitting the model to the training data. The model architecture consists of two dense layers in the encoder and decoder, with 128 and 64 neurons, respectively, each employing the tanh activation function. The hyperparameters used during training are selected based on the optimal results obtained from cross-validation, as summarised in Table 5. These hyperparameters align with the research by [27], who utilised the Adam optimiser with a batch size of 32 and a learning rate of 0.001.

In the LSTM autoencoder approach, forecasting is carried out using a model architecture composed of two LSTM layers in the encoder and two LSTM layers in the decoder, with each layer containing 100 neurons. The hyperparameter settings adopted in this method are the same as those used for the autoencoder model. A comparison of forecasting accuracy for the distributed lag, autoencoder, and LSTM autoencoder methods is illustrated in Figure 6.

Table 5. Model validation results for each hyperparameter combination

Window shift	Batch size	Learning rate	Optimizer	Average MAPE
Semester	**32**	**0.001**	**Adam**	**11.073**
	32	0.001	RMSProp	12.459
	32	0.01	Adam	11.395
	32	0.01	RMSProp	13.298
	64	0.001	Adam	12.117
	64	0.001	RMSProp	13.100
	64	0.01	Adam	13.000
	64	0.01	RMSProp	13.008
Quarter	**32**	**0.001**	**Adam**	**11.308**
	32	0.001	RMSProp	12.296
	32	0.01	Adam	11.322
	32	0.01	RMSProp	12.123
	64	0.001	Adam	12.117
	64	0.001	RMSProp	12.383
	64	0.01	Adam	11.919
	64	0.01	RMSProp	11.911
Trimester	**32**	**0.001**	**Adam**	**11.304**
	32	0.001	RMSProp	12.047
	32	0.01	Adam	12.408
	32	0.01	RMSProp	12.110
	64	0.001	Adam	12.357
	64	0.001	RMSProp	12.993
	64	0.01	Adam	11.656
	64	0.01	RMSProp	12.443

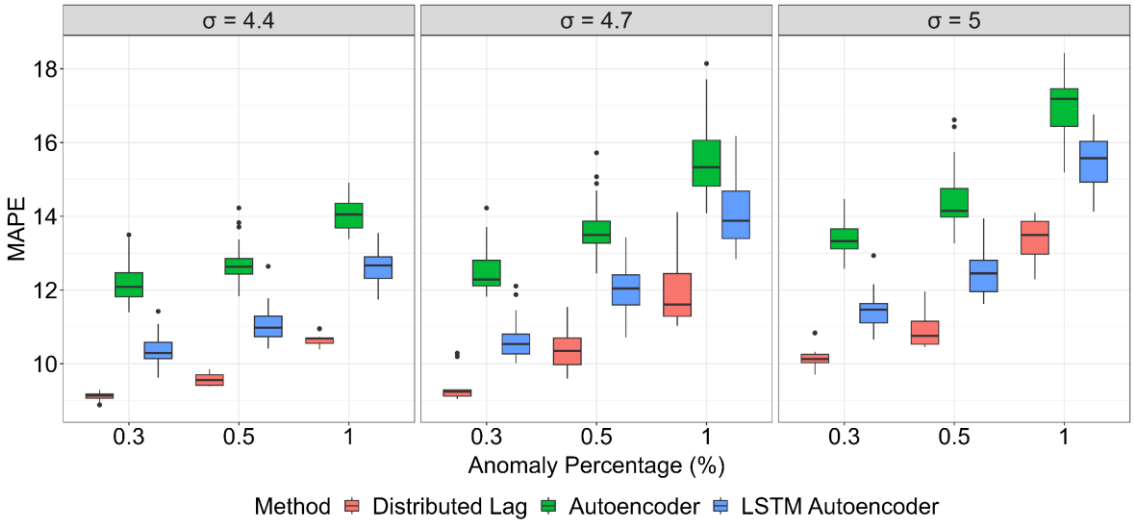

Figure 6. Forecasting performance in each anomaly percentage and depth

The forecasting results in Figure 6 show that, in general, the distributed lag, autoencoder, and LSTM autoencoder methods exhibit good forecasting performance across all anomaly percentages and depths, as indicated by MAPE values below 20%. This interpretation follows commonly adopted guidelines in the forecasting literature, where MAPE values below 20% are generally considered to indicate good or acceptable forecast accuracy (e.g., [28]; [29]; [30]). It is also supported by the characteristics of the air quality index data used in this study. The observed values are relatively moderate, so smaller actual values can lead to larger percentage errors and inflate MAPE. In addition, the hourly data resolution introduces high variability. The study period (2019–2024) captures both seasonal patterns and atypical conditions, such as the COVID-19 period, which may influence air quality dynamics. Furthermore, the presence of extreme observations may also contribute to higher MAPE values. However, the forecasting performance of the distributed lag, autoencoder, and LSTM autoencoder methods is significantly influenced by the percentage of anomalies in the dataset. This is evidenced by the ART ANOVA results, which show that the interaction between anomaly percentage and the anomaly detection method is statistically significant at the 5% level (Table 6).

Table 6. ART ANOVA test of the effect of percentage, depth, and anomaly detection method on forecasting performance

Source of variation	Degrees of Freedom	F Value	P-value	Conclusion
Percentage	2	1499.89	0.000	Significant
Depth	2	755.55	0.000	Significant
Method	2	2140.28	0.000	Significant
Percentage*Depth	4	73.42	0.000	Significant
Percentage*Method	4	16.18	0.000	Significant
Depth*Method	4	2.36	0.052	Not significant
Percentage*Depth*Method	8	0.23	0.984	Not significant

The interaction plot between the percentage and the anomaly detection method in Figure 7 further shows that the MAPE increases as the anomaly percentage increases. In contrast to the anomaly percentage, the anomaly depth does not significantly affect the forecasting performance of the distributed lag, autoencoder, and LSTM autoencoder methods. This is evident from the p-value of the interaction between the anomaly depth factor and the anomaly detection method in the ART ANOVA test, which exceeds the 5% significance level.

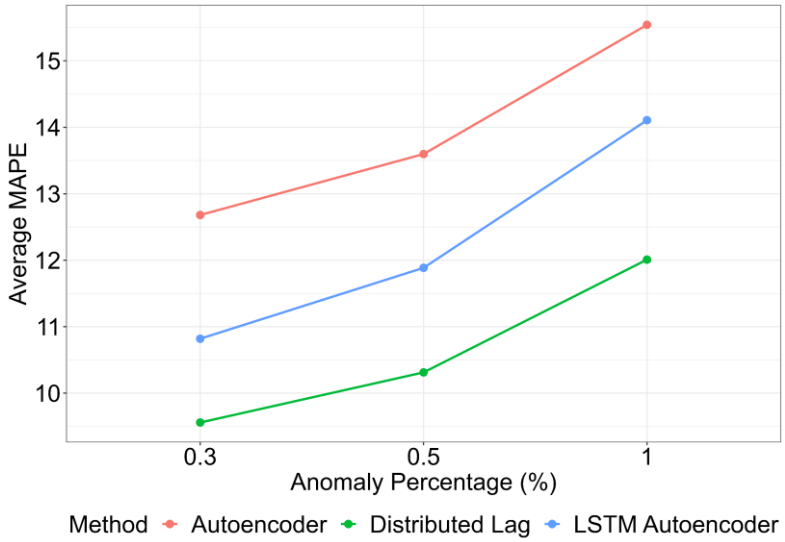

Figure 7. Interaction plot between percentage and anomaly detection method

Anomaly Detection Performance of Distributed Lag, Autoencoder, and LSTM Autoencoder Methods
The distributed lag, autoencoder, and LSTM autoencoder methods are employed to detect anomalies based on the 4σ rule. Under this criterion, an observation is identified as an anomaly when the corresponding forecast error lies outside the predetermined limits. Therefore, the forecasting performance of these models plays an important role in determining their anomaly detection capability. The detection performance of each method at different anomaly percentages is illustrated in Figure 8.

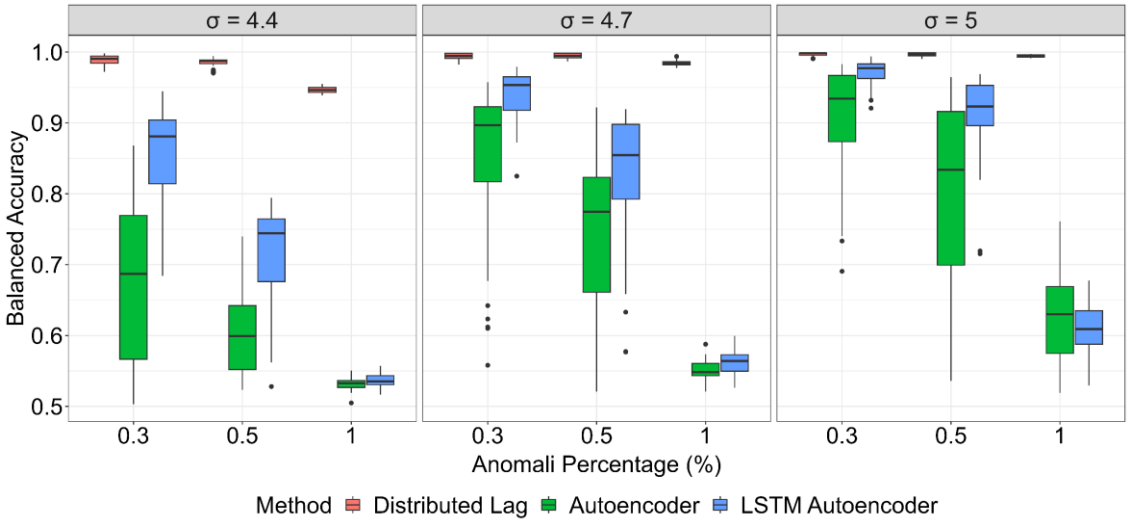

Figure 8. Anomaly detection performance at each anomaly percentage and depth.

Although the forecasting performance of the distributed lag, autoencoder, and LSTM autoencoder methods generally yields good results at various anomaly percentages and depths, the anomaly detection results in Figure 8 demonstrate optimal performance only under specific conditions. This result is consistent with the p-value of the ART ANOVA test for the interaction between percentage, depth, and anomaly detection methods, which is less than the 5% significance level (Table 7). The results suggest that variations in anomaly percentage and anomaly depth play an important role in determining the anomaly detection performance of the distributed lag, autoencoder, and LSTM autoencoder approaches. This observation is reinforced by the interaction plot in Figure 9, which shows the relationships among anomaly percentage, anomaly depth, and the detection methods.

Table 7. ART ANOVA test of the effect of percentage, depth, and anomaly detection method on anomaly detection performance

Source of variation	Degrees of Freedom	F Value	P-value	Conclusion
Percentage	2	682.70	0.000	Significant
Depth	2	358.39	0.000	Significant
Method	2	1101.05	0.000	Significant
Percentage*Depth	4	45.81	0.000	Significant
Percentage*Method	4	234.24	0.000	Significant
Depth*Method	4	91.90	0.000	Significant
Percentage*Depth*Method	8	29.19	0.000	Significant

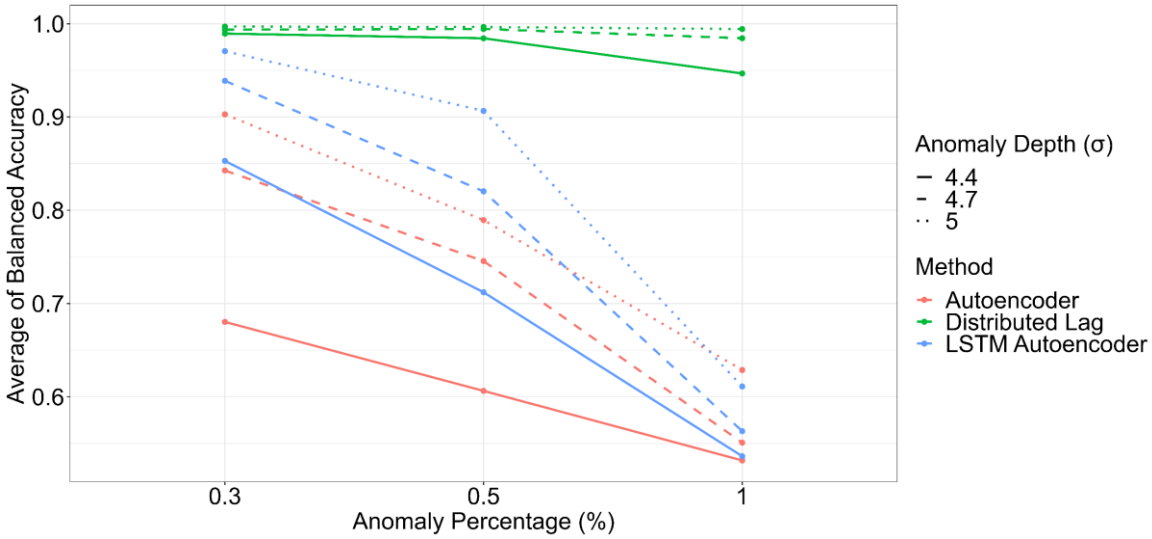

Figure 9. Interaction plot between anomaly percentage, anomaly depth, and anomaly detection method

The Effect of Anomaly Percentage and Depth on the Forecasting Performance of Distributed Lag, Autoencoder, and LSTM Autoencoder Methods

The forecasting performance of the distributed lag, autoencoder, and LSTM autoencoder methods is significantly influenced by the percentage of anomalies, with a higher MAPE value. These results align with research by [31]. This is possible because the more anomalies in the data, the more deviant the forecast results become, leading to greater forecasting errors.

Comparing the forecasting performance across each anomaly percentage for each anomaly detection method, the distributed lag method performed best. This method exhibited the most gradual increase in MAPE value with each increase in the percentage of anomalies, indicating its robustness. Although the distributed lag method is a conventional statistical method, its superiority stems from its use of a one-step-ahead forecasting procedure. These results align with research by [32], which demonstrated that one-step-ahead forecasting procedures in statistical methods, such as SARIMA, performed better than multistep forecasting procedures in deep learning methods, including recurrent neural networks (RNNs) and LSTMs.

Conversely, the autoencoder method performed the worst at each anomaly percentage. This is due to the fundamental characteristic of the autoencoder method, which focuses solely on reconstructing inputs into similar outputs without considering long-term dependencies in the data [33]. However, this method showed a gentler increase in MAPE than the LSTM autoencoder method.

Among the methods evaluated, the LSTM autoencoder exhibited the largest increase in MAPE as the anomaly percentage increased, indicating that it is the least robust with respect to variations in anomaly percentage. Nevertheless, its forecasting performance remained superior to that of the standard autoencoder. This advantage stems from the LSTM autoencoder's ability to model long-term dependencies in the data [34]. However, this capability did not enable it to consistently outperform the distributed lag method, as the latter achieved better forecasting performance across the three anomaly percentage scenarios.

In contrast to the anomaly percentage, anomaly depth does not significantly affect the forecasting performance of the distributed lag, autoencoder, and LSTM autoencoder methods. This is because, in the distributed lag method, extreme values enter linearly through the lag structure, and their influence is dampened by coefficients estimated based on the majority of regular observations. As a result, increasing the anomaly depth does not substantially change the temporal relationship. Meanwhile, in the autoencoder and LSTM autoencoder methods, the temporal representation learning process focuses on identifying dominant data patterns, making the model more responsive to anomaly occurrence than to their severity. Consequently, anomaly depth has only a local impact and is insufficient to significantly influence the underlying temporal structure of the forecasting process.

Effect of Anomaly Percentage and Depth on the Anomaly Detection Performance of Distributed Lag, Autoencoder, and LSTM Autoencoder Methods

The anomaly detection performance of the distributed lag, autoencoder, and LSTM autoencoder methods is significantly influenced by the percentage of anomalies present in the dataset. The higher the anomaly percentage, the lower the balance accuracy. This is because the higher the anomaly percentage, the more difficult it is to classify anomalous and non-anomalous observations, resulting in more misclassifications. Conversely, the lower the anomaly depth, the lower the balance accuracy. This is because the lower the anomaly depth, the smaller the difference between anomalous and non-anomalous observations. Consequently, classifying anomalous and non-anomalous observations becomes more challenging, leading to a higher number of misclassifications.

A comparison of anomaly detection performance across the three methods for different anomaly percentages and depths shows that the distributed lag approach achieves the best overall performance. This method demonstrates strong robustness across all anomaly percentages when the anomaly depths are 4.7σ and 5σ. However, when the anomaly depth is reduced to 4.4σ, its robustness is only maintained at an anomaly percentage of 0.5%. When the anomaly percentage increases to 1.0%, the performance of the distributed lag method declines considerably, as reflected by a notable reduction in balanced accuracy.

In contrast to the distributed lag method, which demonstrates relatively stable performance, the anomaly detection capability of the autoencoder and LSTM autoencoder methods is more sensitive to variations in anomaly percentage and anomaly depth. Overall, the autoencoder approach shows the weakest anomaly detection performance among the evaluated methods. Nevertheless, compared with the LSTM autoencoder, the autoencoder model exhibits greater robustness when the anomaly percentage increases. At the highest anomaly percentage (1%), the detection performance of both methods becomes relatively similar. Even when the anomaly depth reaches 5σ, the autoencoder still performs better than the LSTM autoencoder.
Despite this advantage, the performance of the autoencoder method is strongly influenced by anomaly depth. A decrease in anomaly depth leads to a more pronounced reduction in balanced accuracy. However, this substantial decline is mainly observed at an anomaly percentage of 0.5%. When the anomaly percentage increases to 1%, the reduction in balanced accuracy is less extreme compared to that observed at lower anomaly depths..

CONCLUSION

The anomaly detection performance of the distributed lag, autoencoder, and LSTM autoencoder methods is inseparable from the influence of anomaly characteristics in the data, such as the percentage and depth of anomalies. An increase in the percentage of anomalies will decrease the method's performance, while an increase in the depth of anomalies will improve the method's performance. Based on the evaluation results of the three methods on the combination of percentage and depth of anomalies, the distributed lag method shows the highest level of robustness compared to the other two methods. This method maintains anomaly detection performance at every percentage of anomalies for anomaly depths of 4.7σ and 5σ, and remains robust up to an anomaly percentage of 0.5% at a depth of 4.4σ. In contrast, the performance of the autoencoder and LSTM autoencoder methods tends to decrease as the percentage of anomalies increases and the depth of anomalies decreases.

Despite these findings, this study has several limitations, particularly the use of simulated anomalies that may not fully reflect real-world anomaly patterns. The analysis also relies on a selected thresholding rule, which may not generalise well or remain optimal when applied to datasets beyond the air quality index used in this study. In addition, the hyperparameter search for deep learning models remains limited, potentially limiting the performance of autoencoder-based approaches. Accordingly, future research should explore

more adaptive, data-driven thresholding strategies and conduct a more comprehensive investigation of hyperparameter configurations to enhance model performance. Moreover, this study does not fully examine other factors that may influence anomaly detection performance. Future work may consider additional characteristics of time series data, such as seasonal patterns, which can affect model sensitivity and robustness [35]. Furthermore, hybrid approaches that combine statistical and deep learning methods, such as the LSTM–ARIMA model [36], present a promising avenue for improving both the accuracy and robustness of anomaly detection in time series data.

REFERENCES

[1] V. Kozitsin, I. Katser, and D. Lakontsev, "Online forecasting and anomaly detection based on the ARIMA model," *Applied Sciences*, vol. 11, no. 3194, pp. 1–13, Apr. 2021, doi: 10.3390/app11073194.

[2] A. Choudhary, S. Kumar, and M. Sharma, "A Framework for Data Prediction and Forecasting in WSN with Auto ARIMA," *Wireless Personal Communications*, vol. 123, pp. 2245–2259, Oct. 2022, doi: 10.1007/s11277-021-09237-x.

[3] L. Rubio, A. P. Pinedo, A. M. Castano, and F. Ramos, "Forecasting volatility by using wavelet transform , ARIMA and GARCH models," *Eurasian Economic Review*, vol. 13, pp. 803–830, Dec. 2023, doi: 10.1007/s40822-023-00243-x.

[4] W. A. Nugroho, F. D. Rachman, B. K. Siahu, I. A. Iswanto, and S. Joddy, "Hybrid ensemble model approach for stock price forecasting using LSTM, random forest, ARIMA, and linear regression as meta-learner," in *Procedia Computer Science*, Elsevier B.V., Nov. 2025, pp. 901–910. doi: 10.1016/j.procs.2025.09.033.

[5] L. Luo, "Statistical inference method under high-dimensional data and its theoretical discussion in accurate analysis," in *International Conference on Big Data, Economic Development and Management*, Qingdao, Mar. 2025, pp. 402–408. doi: https://doi.org/10.56028/aemr.13.1.402.2025.

[6] S. Lin and Y. Feng, "Research on stock price prediction based on orthogonal Gaussian basis function expansion and Pearson correlation coefficient weighted lstm neural network," *Advances in Computer, Signals and Systems*, vol. 6, no. 5, pp. 23–30, Nov. 2022, doi: 10.23977/ACSS.2022.060504.

[7] F. M. M. R. Ferreira and R. J. F. Rossetti, "Underspecification and uncertainty in deep learning models : Is there a connection ?," *Neural Computing and Applications*, vol. 37, pp. 19579–19595, Jul. 2025, doi: 10.1007/s00521-025-11415-y.

[8] X. Meng, H. Chang, and X. Wang, "Methane concentration prediction method based on deep learning and classical time series analysis," *Energies*, vol. 15, no. 2262, pp. 1–15, Mar. 2022, doi: https:// doi.org/10.3390/en15062262.

[9] C. Guo, B. Zhao, and Y. Bai, "DeepCore : A Comprehensive library for coreset," in *Database and Expert Systems Applications: 33rd International Conference*, Jun. 2022, pp. 124–138. doi: https://doi.org/10.1007/978-3-031-12423-5_14.

[10] R. Janarthanan, P. Partheeban, K. Somasundaram, and P. N. Elamparithi, "A deep learning approach for prediction of air quality index in a metropolitan city," *Sustainable Cities and Society*, vol. 67, pp. 1–11, Jan. 2021, doi: 10.1016/j.scs.2021.102720.

[11] B. S. Situmeang *et al.*, "Pengaruh tingkat polusi udara terhadap tingkat pengidap penyakit ISPA di lingkup masyarakat kramat jati," *Jurnal of Comprehensive Science*, vol. 2, no. 12, pp. 1520–1539, Dec. 2023.

[12] A. Hadianfar, S. Rastaghi, H. Tabesh, and A. Saki, "Application of distributed lag models and spatial analysis for comparing the performance of the COVID-19 control decisions in European countries," *Nature*, vol. 13, no. 17466, pp. 1–11, Oct. 2023, doi: 10.1038/s41598-023-44830-z.

[13] A. L. Alfeo, M. G. C. A. Cimino, G. Manco, E. Ritacco, and G. Vaglini, "Using an autoencoder in the design of an anomaly detector for smart manufacturing," *Pattern Recognition Letters*, vol. 136, pp. 1–8, Aug. 2020, doi: 10.1016/j.patrec.2020.06.008.

[14] M. S. Elsayed, N. A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using LSTM based autoencoder," in *Q2SWinet 2020 - Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, A. Montefaoui and P. Sun, Eds., Alicante: Association for Computing Machinery, Nov. 2020, pp. 37–45. doi: 10.1145/3416013.3426457.

[15] U. Cetin and M. Tasgin, "Anomaly detection with multivariate k-sigma score using monte carlo," in *5th International Conference on Computer Science and Engineering, UBMK 2020*, 2020, pp. 94–98. doi: 10.1109/UBMK50275.2020.9219482.
[16] M. Wolny, "Anomaly detection in univariate time series using a multi-criteria approach," *Scientific Paper of Silesian University of Technology Organization and Management Series*, vol. 2024, no. 213, pp. 665–680, 2024, doi: 10.29119/1641-3466.2024.213.46.
[17] J. Shi, G. He, and X. Liu, "Anomaly Detection for Key Performance Indicators Through Machine Learning," in *Proceedings of 2018 6th IEEE International Conference on Network Infrastructure and Digital Content, IC-NIDC 2018*, IEEE, 2018, pp. 1–5. doi: 10.1109/ICNIDC.2018.8525714.
[18] E. Farooq, M. Milano, and A. Borghesi, "Federated LSTM autoencoders for time series anomaly detection in production-scale HPC systems," *Knowledge-Based Systems*, vol. 334, pp. 1–19, Dec. 2026, doi: 10.1016/j.knosys.2025.115043.
[19] A. Ostermann, A. Bajrami, and A. Bogensperger, "Short-term forecasting of German generation-based CO2 emission factors using parametric and non-parametric time series models," *Energy Informatics*, vol. 7, no. 1, pp. 1–28, Jan. 2024, doi: 10.1186/s42162-024-00303-9.
[20] G. Huicai, "Forecasting tourism demand using big data," The Hong Kong Polytechnic University, 2024. doi: 10.1007/978-981-97-0558-0_3.
[21] E. Sharma, L. Davis, J. Ivy, and M. Chi, "Data to donations: towards Iin-kind food donation prediction across two coasts," in *Global Humanitarian Technology Conference, (GHTC)*, E. Perkins, Ed., Seattle: IEEE, Nov. 2021, pp. 281–288. doi: 10.1109/GHTC53159.2021.9612484.
[22] L. Chaka, M. A. M. A. Elbasit, and S. Jombo, "Predicting precipitation using dynamic distributed lag models in arid and sub-humid regions of South Africa," *Scientific African*, vol. 29, pp. 1–15, Aug. 2025, doi: 10.1016/j.sciaf.2025.e02924.
[23] I. D. Mienye and T. G. Swart, "Deep autoencoder neural networks : a comprehensive review and new perspectives," *Archives of Computational Methods in Engineering*, vol. 32, pp. 3981–4000, Mar. 2025, doi: 10.1007/s11831-025-10260-5.
[24] A. A. Alguhi and A. M. Al-shaalan, "LSTM-Based prediction of solar irradiance and wind speed for renewable energy systems," *Energies*, vol. 18, no. 4594, pp. 1–18, Aug. 2025, doi: https://doi.org/10.3390/en18174594.
[25] Q. Sun, Z. Tang, J. Gao, and G. Zhang, "Short-term ship motion attitude prediction based on LSTM and GPR," *Applied Ocean Research*, vol. 118, no. 102927, pp. 1–12, Nov. 2022, doi: 10.1016/j.apor.2021.102927.
[26] U. B. Mahadevaswamy and P. Swathi, "Sentiment analysis using bidirectional LSTM network," in *International Conference on Machine Learning and Data Engineering*, Elsevier B.V., Jan. 2023, pp. 45–56. doi: 10.1016/j.procs.2022.12.400.
[27] C. M. Rosca, M. Carbureanu, and A. Stancu, "Data-driven approaches for predicting and forecasting air quality in urban areas," *Applied Sciences*, vol. 15, no. 8, pp. 1–36, Apr. 2025, doi: 10.3390/app15084390.
[28] S. A. Rokonuzzaman and K. S. R. W. S. Tan, "A comparative analysis of forecasting algorithms for predicting municipal solid waste generation in Chittagong City," *International Journal of Environmental Science and Technology*, vol. 22, pp. 14213–14224, 2025, doi: 10.1007/s13762-025-06488-0.
[29] M. R. Nurhambali, Y. Angraini, and A. Fitrianto, "Implementation of Long Short-Term Memory for Gold Prices Forecasting," *Malaysian Journal of Mathematical Sciences*, vol. 18, no. 2, pp. 399–422, 2024, doi: 10.47836/mjms.18.2.11.
[30] E. Vivas, H. Allende-cid, and R. Salas, "A Systematic Review of Statistical and Machine Learning Methods for Electrical Power Forecasting with Reported MAPE Score," *Entropy*, vol. 22, no. 1412, pp. 1–24, 2020, doi: 10.3390/e22121412.
[31] J. Luo, T. Hong, and M. Yue, "Real-time anomaly detection for very short-term load forecasting," *Journal of Modern Power Systems and Clean Energy*, vol. 6, no. 2, pp. 235–243, Jan. 2018, doi: 10.1007/s40565-017-0351-7.
[32] S. Suradhaniwar, S. Kar, S. S. Durbha, and A. Jagarlapudi, "Time series forecasting of univariate agrometeorological data: A comparative performance evaluation via one-step and multi-step ahead forecasting strategies," *Sensors*, vol. 21, no. 7, pp. 1–34, Apr. 2021, doi: 10.3390/s21072430.
[33] K. Berahmand, F. Daneshfar, E. S. Salehi, Y. Li, and Y. Xu, "Autoencoders and their applications in machine learning: a survey," *Artificial Intelligence Review*, vol. 57, no. 28, pp. 1–52, Feb. 2024, doi: 10.1007/s10462-023-10662-6.

[34] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, "LSTM-autoencoder-based anomaly detection for indoor air quality time-series data," *IEEE Sensors Journal*, vol. 23, no. 4, pp. 3787–3800, 2023, doi: 10.1109/JSEN.2022.3230361.
[35] W. Wu *et al.*, "Developing an unsupervised real-time anomaly detection scheme for time series with multi-seasonality," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 9, pp. 4147–4160, Sep. 2022, doi: 10.1109/TKDE.2020.3035685.
[36] S. Xue, H. Chen, and X. Zheng, "Detection and quantification of anomalies in communication networks based on LSTM-ARIMA combined model," *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 10, pp. 3159–3172, 2022, doi: 10.1007/s13042-022-01586-8.