International Journal on Advanced Science Engineering Information Technology

Vol.14 (2024) No. 6
ISSN: 2088-5334

Comprehensive Analysis and Improved Techniques for Anomaly Detection in Time Series Data with Autoencoder Models

Sarvarbek Erniyazov [a], Yong-Min Kim [b], Mohammed Abdul Jaleel [c], Chang Gyoon Lim [a,*]

[a] *Department of Computer Engineering, Chonnam National University, Yeosu, Jeonnam, Republic of Korea*
[b] *Department of Electronic Commerce, Chonnam National University, Yeosu, Jeonnam, Republic of Korea*
[c] *Department of Applied Informatics, Kimyo International University in Tashkent, Tashkent, Uzbekistan*
*Corresponding author: *cglim@jnu.ac.kr*

***Abstract*—Anomaly detection is critical in various sectors, offering significant advantages by precisely identifying and mitigating system failures and errors, thus preventing severe losses. This study provides a comprehensive comparative anomaly detection analysis through two sophisticated deep learning models: Autoencoder and Long Short-Term Memory (LSTM) Autoencoder, explicitly focusing on temperature and sound time series data. The paper starts with a detailed theoretical foundation, elaborating on both models' mechanics and mathematical formulations. We then advance to the empirical phase, where these models are rigorously trained and tested against a robust dataset. The effectiveness of each model is meticulously assessed through a suite of metrics that gauge their accuracy, sensitivity, and robustness in anomaly detection scenarios. Additionally, we explore the deployment of these models in a real-time environment, where they actively engage in anomaly detection on incoming data streams. The anomalies detected are dynamically displayed on a user-friendly graphical interface, making the results readily accessible and interpretable for users at all levels of technical expertise. Quantitative evaluations of the models are conducted using key performance metrics such as accuracy, precision, recall, and F1-score. Our analysis reveals that the LSTM Autoencoder model excels with an impressive accuracy rate of 99%, while other metrics also affirm its superior performance, marking it as exceptionally effective and reliable. This study highlights the LSTM Autoencoder's advanced anomaly detection capabilities and establishes its superiority over the traditional Autoencoder model in processing complex time series data. The insights gained here are crucial for industries focused on predictive maintenance and quality control, where early anomaly detection is key to maintaining operational efficiency and safety.**

***Keywords*— Anomaly detection; autoencoder; LSTM autoencoder; time-series data.**

Manuscript received 11 Apr. 2024; revised 9 Jul. 2024; accepted 12 Sep. 2024. Date of publication 31 Dec. 2024.

I. INTRODUCTION

The rapid growth of sensors and advancements in sensing technologies have been influencing many industries, including manufacturing [1], agriculture [2], transportation [3], robotics [4], and to name a few. These sensors continuously generate univariate or multivariate data, which can be helpful to monitor and predict equipment performance, identify potential failures, optimize maintenance schedules, and ensure safety in various industrial applications [5], [6], [7], [8]. Identifying anomalous subsequences in time-series sensor data is crucial. They can signal critical events, including production faults, delivery delays, system malfunctions, or irregular heartbeats, making their detection highly significant [9]. Mechanical devices like autonomous vehicles, engines, and aircraft parts commonly have multiple sensors to track their functioning and condition over time. Given that time series data are typically large and complex, conventional approaches, such as mathematical models assuming stability or prediction models based on error predictions, are often insufficient in dealing with the unpredictability of such data [10].

The main issues in anomaly detection for time-series sensor data include high dimensionality and complexity due to multiple variables and their interactions, which complicate accurate modeling [11]. Temporal dependencies pose a challenge as patterns can influence anomalies over time, requiring advanced models to handle sequential data [12]. Non-stationarity is another concern, as the statistical properties of time-series data can change, necessitating models that adapt to evolving patterns. Additionally, the imbalanced nature of datasets, where anomalies are much rarer than everyday observations, as presented in Fig. 1, can lead to biased models that overlook anomalies. Finally, the

scarcity of labeled data for training models makes it difficult to identify anomalies, as manual labeling is often impractical and resource-intensive.

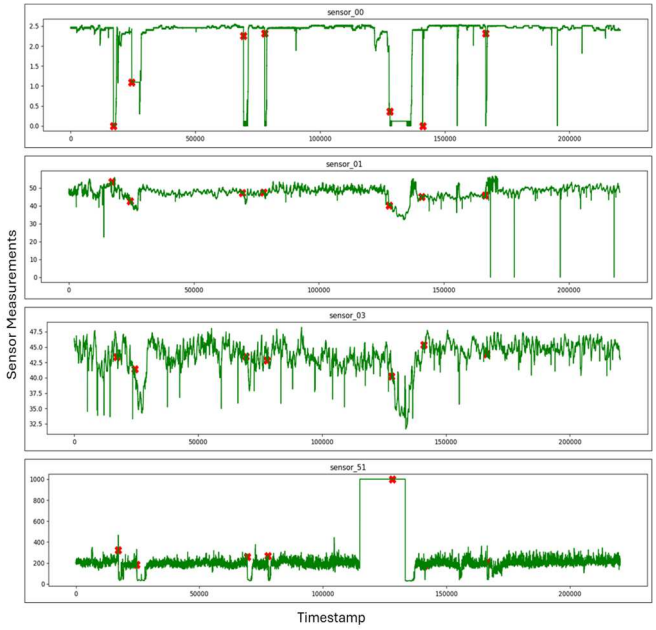

Fig. 1 Time-series data with rare anomaly observations compared to normal observations

Researchers have introduced numerous algorithms to automatically identify these unusual patterns by applying statistical models [13], linear models, proximity, machine learning, and deep learning algorithms [14]. Statistical solutions are most widely utilized for anomaly detection, and k-NN-based algorithms are the commonly employed and simplest unsupervised method to detect outliers. This distance-based algorithm determines the anomaly score by calculating the k-nearest-neighbors (k-NN) distance. Despite its effectiveness, this approach is computationally intensive, highly sensitive to the choice of k, and can struggle when normal data points lack sufficient neighboring points. Breunig et al. [15] proposed the widely adopted unsupervised method for local density-based anomaly detection, introduced as a Local Outlier Factor (LOF). LOF identifies the k-nearest-neighbors for each instance by measuring its distances to all other instances. This method assesses the local density deviation of a data point relative to its neighbors, making it particularly effective for detecting anomalies in complex datasets. The LOF algorithm enhancement is presented in the connectivity-based outlier factor (COF) approach to improve anomaly detection capabilities in linear systems [16]. Additionally, clustering-based algorithms have been introduced as unsupervised anomaly detection methods, which group similar data points into clusters based on characteristics or features. Any data points that do not fit well into any cluster or lie a significant distance from cluster centroids are considered an anomaly. However, the non-deterministic nature of clustering algorithms does not always allow for precisely determining the correct number of clusters, complicating the computation of anomaly scores.

Recent advancements in deep learning algorithms have significantly enhanced time-series analysis. Wei et al. [17] proposed an anomaly detection method using LSTM-autoencoder architecture, training their model on normal timestamps to identify later anomalies based on error thresholds for indoor quality data. Similarly, in [18], four different types of outliers have been measured in ECG data by applying RNN-LSTM approach. Combining the Wavelet and Hilbert transform with a deep learning algorithm is proposed in [19] to detect irregularities within temporal data patterns. This work introduces a novel approach to detecting new or unusual patterns in data using deep autoencoders. Their method involves compressing data through these autoencoders and setting an error threshold to identify novelty groups by combining density-based clustering [20]. Munir et al. also adopted deep autoencoders but focused on the fraud detection domain within large-scale accounting data [21]. By leveraging the autoencoders' ability to learn standard data patterns, their system effectively highlights anomalies that deviate from these patterns, aiding in the early detection and prevention of fraud.

In this work, we present anomaly detection in temperature and sound time series using Autoencoder and LSTM Autoencoder architecture and provide a comparative analysis of their performance using various metrics. Anomaly detection in temperature and sound time series data is crucial across multiple applications due to its significant impact on safety, efficiency, and cost savings. In temperature data, detecting anomalies can prevent equipment failures, ensure safety by identifying hazardous conditions, maintain product quality, and enhance energy efficiency. For sound data, anomaly detection is essential in machine condition monitoring for predictive maintenance, enhancing security through surveillance, and monitoring health by identifying irregular sounds in medical contexts. Additionally, it plays a key role in environmental noise monitoring and quality control in manufacturing processes.

II. Material and Method

A. Technical Background of Anomaly Detection

As we mentioned above, anomaly detection is a critical data analysis method utilized in various domains such as finance, network security, and health monitoring to pinpoint atypical data points that significantly deviate from the norm. Anomalies are classified into point, contextual, and collective anomalies, each pertinent to distinct scenarios like detecting fraud or analyzing network traffic. Conventional statistical techniques have progressed into more advanced machine learning methods, encompassing supervised, unsupervised, and semi-supervised learning [22], [23], [24]. Notably, AutoEncoders in semi-supervised contexts have garnered attention for their effectiveness in learning normal data distributions and detecting outliers based on reconstruction errors [25].

Deep learning approaches, including convolutional (CNN) and recurrent neural networks (RNN), further enhance anomaly detection capabilities, particularly in managing high-dimensional and dynamic data streams [26], [27], [28]. Successful anomaly detection necessitates sophisticated analytical methods to tackle challenges such as high dimensionality and imbalanced datasets, with assessment often relying on metrics like Precision, Recall, and the AUC-ROC curve rather than conventional accuracy measures. This

field rapidly evolves as new data intricacies surface, emphasizing the necessity for innovative anomaly detection algorithms.

B. Principles of Autoencoder and LSTM-Autoencoder

An autoencoder is a form of unsupervised learning model primarily designed to compress input data into a significant representation and then decode it to reconstruct the input as closely as possible to the original. This encoding-decoding architecture are widely applied in feature extraction, image denoising, image compression and search, missing value imputation, and anomaly detection tasks. In terms of anomaly detection, obtaining normal and abnormal datasets is challenging due to the lack of explicit instructions or conditions for collecting genuine abnormal data from the target environment [29].

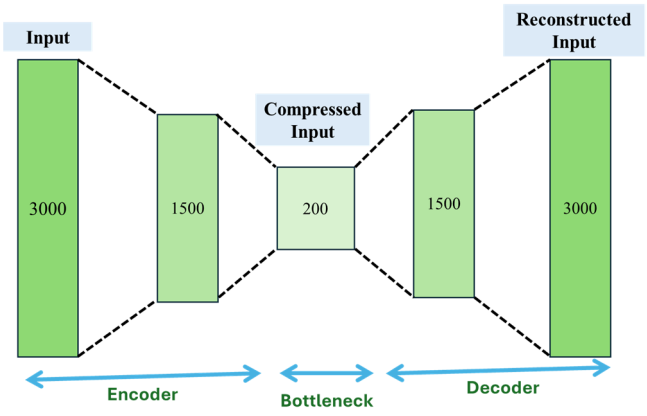

Fig. 1 Autoencoder architecture

Autoencoders and LSTM-Autoencoders are pivotal in anomaly detection for time series data, as they excel at learning data representations and reconstructing input patterns. Traditional autoencoders, comprising an encoder and decoder, reduce input dimensions to a compressed representation and reconstruct the original input by minimizing reconstruction error. Fig. 2 presents the conventional Autoencoder architecture, and the input dimensions are reduced through the encoder from 3000 to a compressed representation of 200 dimensions. The decoder then reconstructs the original input from this compressed representation. Traditional autoencoder architecture can be efficient when data does not have sequential relationships, as in the case of static or non-temporal data. This architecture works well for static data but fails to capture temporal dependencies crucial for time series data.

When there is sequential relationship between data points, such as in time-series data, it is crucial to preserve the temporal information. Fig. 3 illustrates the LSTM-Autoencoder architecture designed to address this need. The LSTM-Autoencoder consists of an LSTM-based encoder and decoder. The encoder processes sequential input data to capture temporal dependencies and compresses it into an encoded representation. The decoder then reconstructs the input from this encoded representation, ensuring that sequential relationships within the data are maintained. This makes the LSTM-Autoencoder architecture particularly suitable for time-series data, where the order and dependencies between data points are essential for accurate anomaly detection and prediction.

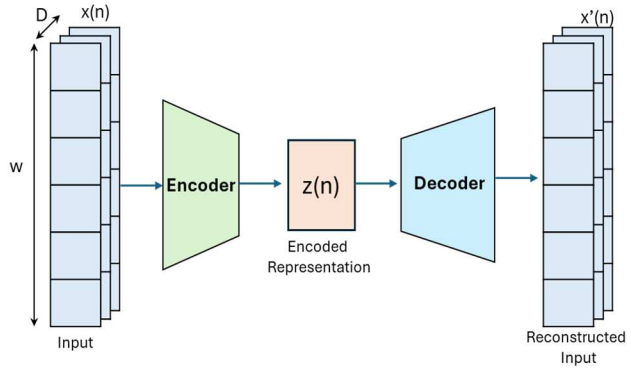

Fig. 3 LSTM-Autoencoder architecture

C. Data Collection and Preprocessing

In this work, we utilized the HAI (HIL-based Augmented ICS) Security Dataset, which was gathered from an industrial control system (ICS) test environment enhanced with a Hardware-In-the-Loop (HIL) simulator [30]. The HIL simulator replicates pumped-storage hydropower and steam-turbine power generation processes. The testbed comprises four distinct processes: the boiler process, the turbine process, the water treatment process, and the HIL simulation. The boiler process (P1) entails low-pressure and moderate-temperature water-to-water heat transfer, regulated using the Emerson Ovation Distributed Control System (DCS). The turbine process (P2) features a rotor kit that accurately replicates the dynamics of a real rotating machine, managed by GE's Mark VIe DCS. The data captures multiple measurements at one-minute intervals from August 4 to 7, 2022, and includes 280k data rows with five columns, including flow temperature (P1_FT01), flow temperature zero (P1_FT01Z), pressure indicator transmitter (P1_PIT01), turbine inlet temperature (P2_TIT01), and vibration transmitter readings (P1_VIBTR01).

During the pre-processing phase, we conducted data cleaning procedures by removing missing values, converting timestamp values to a DateTime format, and coercing any non-numeric values in the sensor readings to numeric types. Data points were sampled at one-minute intervals to maintain temporal structure for time-series analysis. We normalized and processed the dataset using PyTorch's "Dataset" and "DataLoader" classes.

D. Data Collection and Preprocessing AutoEncoder Model Architecture

As we mentioned above, an autoencoder learns to minimize the reconstruction error for normal data during training by creating nearly identical reconstructed input, as presented in Fig. 4Error! Reference source not found.. The detailed formulation of the autoencoder model for the time series data can be formulated as follows:

1) Input data: The input time-series data is represented as $\boldsymbol{x^{(i)} \epsilon R^m}$, where m is the number of features.

2) Encoder function: The encoder $\boldsymbol{g_\phi}$ transforms the input data x into a lower-dimensional representation $\boldsymbol{z \epsilon R^n}$ (with $n < m$) using Eq. (1).

$$z = g_\phi(x) = ReLU(W_2 \times ReLU(W_1 \times x + b_1) + b_2) \quad (1)$$

where $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$ represent weight matrices and $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$ are bias vectors. The activation function utilized in the neural network layers of the autoencoder is the ReLU (Rectified Linear Unit) function.

3) Latent space representation: The latent space, $\mathbf{z}$, encapsulates the key characteristics of the input data in a condensed format.

4) Decoder function: The decoder $\boldsymbol{f_\theta}$ reconstructs the input data from the latent representation with the following Eq. (2).

$$x' = f_\theta(z) = ReLU(W_4 \times ReLU(W_3 \times z + b_3) + b_4) \quad (2)$$

where $\boldsymbol{W_3}$ and $\boldsymbol{W_4}$ are weight matrices and $\boldsymbol{b_3}$ and $\boldsymbol{b_4}$ are bias vectors.

5) Reconstruction error: The reconstruction error is calculated as the mean squared error (MSE) between the original input $\boldsymbol{x}$ and the reconstructed input $\boldsymbol{x'}$ as presented in Eq. (3)

$$\mathbf{MSE} = ||\boldsymbol{x} - \boldsymbol{x'}||^2 \quad (3)$$

The training process aims to reduce the reconstruction error, which is defined as the difference between the original input and its reconstruction. During the anomaly detection stage, the trained autoencoder evaluates new data points. The reconstruction error is computed for each data point, and points with errors exceeding a predefined threshold are classified as anomalies (Eq. (4)):

$$\text{Anomaly} = \begin{cases} 1, & \text{if Error} > \text{Threshold} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

More precisely, abnormal inputs typically result in higher reconstruction errors during inference than regular inputs, thus facilitating their detection based on the differences in reconstructed values. However, the convolutional Autoencoder architecture does not consider the temporal relationships between data points. Therefore, the model analyzes each data point independently.

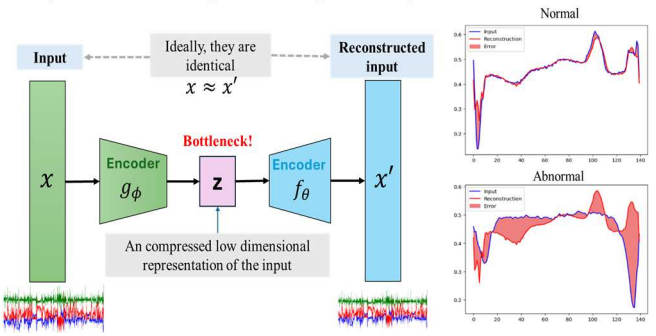

Fig. 4 Autoencoder architecture with normal and abnormal data detection results

In a more concise form, the loss in the Autoencoder architecture is calculated using the following Eq. (5):

$$L(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}(x^i - f_\theta(g_\phi(x^i)))^2 \quad (5)$$

where f_θ and g_ϕ show the encoder and decoder parameters, respectively. The overall equation sums up differences between the original input x and the reconstructed $f_\theta(g_\phi(x^i))$. During training, the Autoencoder aims to minimize this loss, meaning it tries to make the reconstructed outputs as close as possible to the original inputs. This minimization process tunes the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ of the encoder and decoder, respectively. In anomaly detection, Autoencoders are trained on normal data. When new data is inputted, if the reconstruction error (difference between the input and its reconstruction) is significantly high, the data is considered anomalous. Due to training mostly on normal data, the model fails to reconstruct outliers or anomalies effectively.

E. LSTM Autoencoder Model Architecture

To learn the temporal relationship between data points, the Autoencoder model uses the capabilities of the LSTM model as follows. The time series dataset is represented with a length of *L*, where each data point $\boldsymbol{x^{(i)} \epsilon R^m}$ consists of *m*-dimensional vector readings for *m* variables at time-instance $\boldsymbol{t_i}$.

$$\boldsymbol{X} = \{\boldsymbol{x^{(1)}}, \boldsymbol{x^{(2)}}, \boldsymbol{x^{(3)}}, \ldots, \boldsymbol{x^{(L)}}\} \quad (6)$$

First, the model is trained with normal time-series data as presented in Fig. 5. The value $\boldsymbol{x^{(i)}}$ at time $\boldsymbol{t_i}$ and the hidden state $\boldsymbol{h_E^{(i-1)}}$ of the encoder at time $\boldsymbol{t_i - 1}$ are used to compute the hidden state $\boldsymbol{h_E^{(i)}}$ of the encoder at time $\boldsymbol{t_i}$. The hidden state $\boldsymbol{h_E^{(3)}}$, generated by the encoder is referred to as a feature vector and serves as the initial hidden vector of the decoder. The decoder then takes the features generated by encoders as input and reconstructs the original data in reverse order. The autoencoder learns to minimize Mean Squared Error (MSE) between the input sequence $\mathbf{x^{(1)}}, \mathbf{x^{(2)}}, \mathbf{x^{(3)}}, \ldots, \mathbf{x^{(n)}}$ and reconstructed sequence $\mathbf{x^{(\wedge 1)}}, \mathbf{x^{(\wedge 2)}}, \mathbf{x^{(\wedge 3)}}, \ldots, \mathbf{x^{(\wedge n)}}$ the input. The learning objective of the model is as Eq. (7):

$$\text{Minimize} \sum_{X \in sN} \sum_{i=1}^{N} \left\| \mathrm{x}^{(i)} - \mathrm{x}^{(\wedge i)} \right\|^2 \quad (7)$$

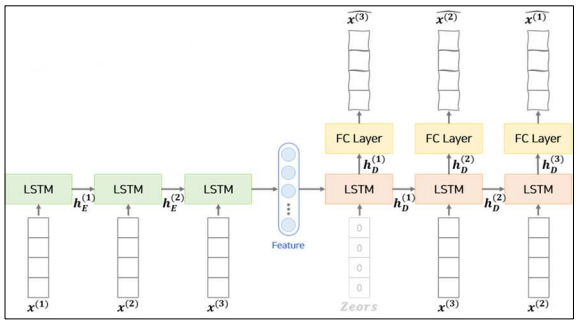

Fig. 5 LSTM Autoencoder training phase

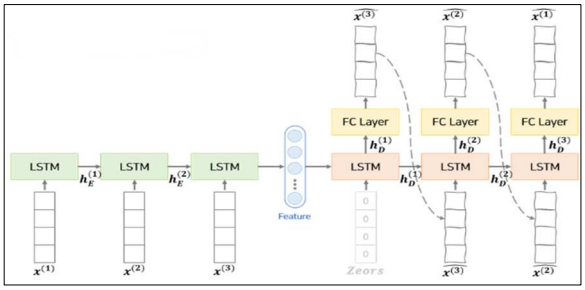

Fig. 6 LSTM-Autoencoder inference phase representation

In contrast to the learning phase, in the inference process (Fig. 6), the encoder generates a feature vector, which is then used as the initial hidden vector for the decoder. The main distinction is that the reconstructed output of the decoder $\boldsymbol{x}(\hat{}\mathbf{3})$ is inputted to the next decoder layer instead of the original data.

F. Performance Evaluation Metrics

The above algorithms are evaluated using *accuracy (A)*, *precision (P)*, *F-score*, and *recall(R)*, and metrics. *Precision(P)* indicates the ratio of accurately identified anomalies to the total number of points labeled as anomalies.

$$A = \frac{T_P+T_N}{T_P+T_N+F_P+F_P};\ P = \frac{T_P}{T_P+F_P};\ R = \frac{T_P}{T_P+F_N} \qquad (8)$$

Recall assesses the ratio of accurately detected anomalies to the total number of actual anomalies. Where True Positives (T_P) refers to the number of actual anomalies that are correctly identified as anomalies. True Negatives (T_N) denote the number of normal points accurately identified as normal. False Positives (F_P) are the normal points incorrectly identified as anomalies, while False Negatives (FN) represent the actual anomalies mistakenly identified as normal. The *F1-Score* is calculated as the harmonic mean of precision and recall, using the formula (Precision×Recall) / (Precision + Recall). On the other hand, *Accuracy* assesses the ratio of correctly identified instances (including both anomalies and normal instances) to the total number of cases.

III. Result and Discussion

A. Experimental Environment and Training Details

We utilized PyTorch, an open-source machine learning library, to implement both Autoencoder and LSTM Autoencoder models. Before feeding the data into the network, we applied Min-Max Scaling to rescale each feature within the 0 to 1 range. This scaling process involves subtracting the minimum value and dividing it by the range of each feature, ensuring that all features contribute equally to the model and preventing scale imbalances.

We employed the Adam optimizer with Mean Squared Error (MSE) loss during training. Thanks to the lightweight nature of our models, training on an NVIDIA GeForce RTX 3050 Laptop GPU was completed in under a minute. The dataset is split into training, validation, and testing sets by dividing 70%, 15%, and 15% segments, respectively.

We used a training set to train Autoencoder and LSTM Autoencoder-based anomaly detectors and validated the results using a validation set. Next, we used unseen data to test the performance of anomaly detectors. We created the GUI dashboard using the Python Plotpy library.

B. Comparative Analysis

Fig. 7 and Fig. 8 represent the training and validation loss results over the epochs for Autoencoder and LSTM-autoencoder-based anomaly detectors, respectively. Note that during the training we applied early stop condition to prevent overfitting and hyperparameter tuning the model. As can be seen from the graphs both models perform well with less training and validation losses. The comparison demonstrates the model's ability to effectively learn and reconstruct sequential patterns in the time-series data, leveraging the LSTM architecture's capability to capture temporal dependencies.

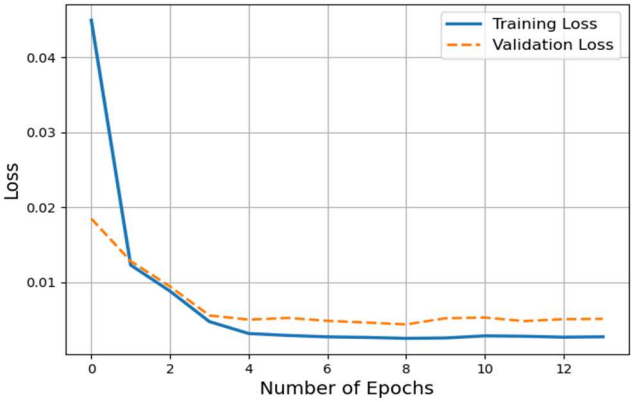

Fig. 7 Autoencoder training and validation loss results

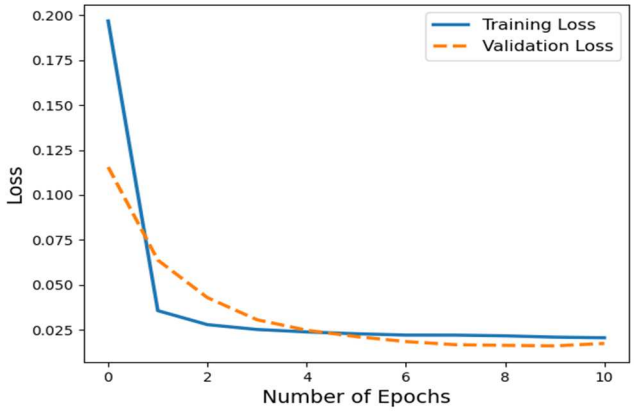

Fig. 8 LSTM-Autoencoder training and validation loss results

In this next step, we evaluate the input and reconstructed input comparison results over the test set for both Autoencoder and LSTM-Encoder approaches. Fig. 9 and Fig. 10 show the performance comparison of both models in terms of input and reconstruction of input values. As can be seen, the reconstruction capability of the Autoencoder based anomaly detection is slightly weak compared to the LSTM-Autoencoder based approach. Therefore, the input and reconstructed input is not fully overlapping in only autoencoder solution. Whereas the reconstruction error is not distinguishable in the LSTM-Autoencoder based approach as presented in Fig. 10.

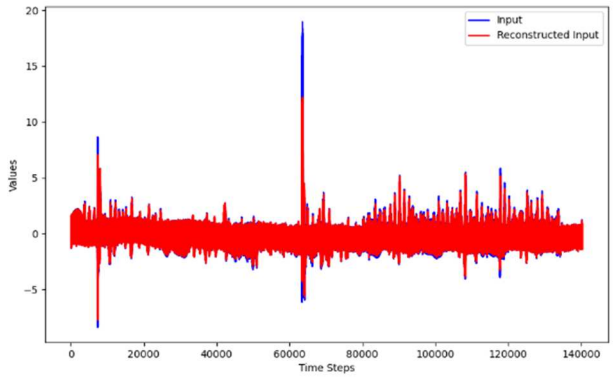

Fig. 9 Input and reconstructed input results using Autoencoder

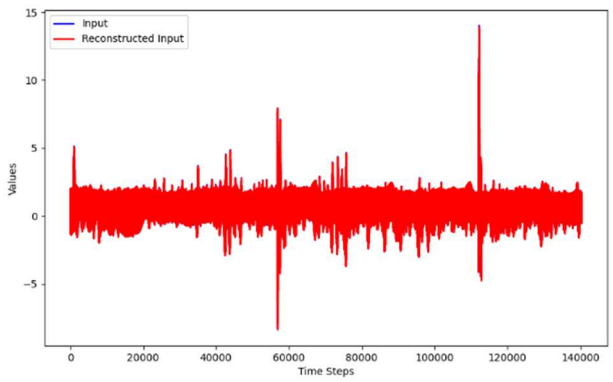

Fig. 10 Input and reconstructed input results using LSTM-Autoencoder

Fig. 11 highlights the comparative performance of the standard Autoencoder and the LSTM Autoencoder models across three error metrics. The LSTM Autoencoder consistently outperforms the standard Autoencoder, with lower MSE (0.01 vs. 0.03), MAE (0.05 vs. 0.09), and RMSE (0.13 vs. 0.21). This demonstrates the enhanced capability of LSTM-based architectures in handling sequential data for time-series anomaly detection tasks, providing more accurate and reliable reconstructions compared to the standard Autoencoder.

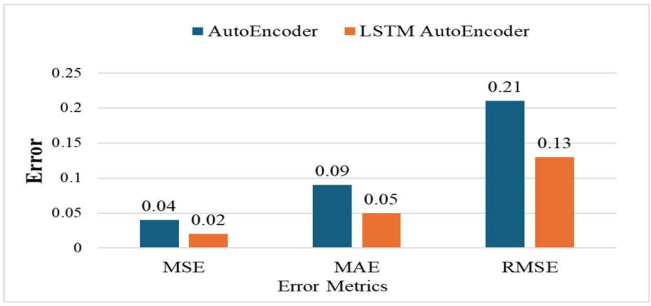

Fig. 11 MSE, MAE, RMSE error metrics

Fig. 12 presents a comparative analysis of performance metrics for the standard Autoencoder and the LSTM Autoencoder models, illustrating four key metrics: Accuracy, Precision, Recall, and F1 score. The LSTM Autoencoder achieves a notably higher accuracy of 0.991 compared to its 0.971, indicating its superior reliability in correctly identifying normal and anomalous data points. In terms of precision, the LSTM Autoencoder shows a value of 0.985, surpassing the Autoencoder's precision of 0.961, which suggests a higher proportion of true positive anomalies among the detected anomalies and fewer false positives.

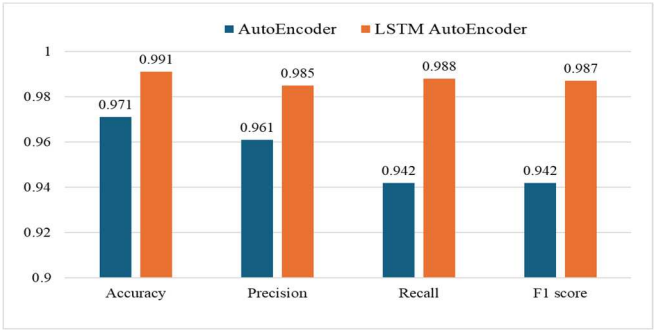

Fig. 12 Accuracy, Precision, Recall, and F1 score metrics

Furthermore, the LSTM Autoencoder demonstrates an impressive recall of 0.988, whereas the Autoencoder records a recall of 0.942, highlighting the LSTM Autoencoder's effectiveness at identifying actual anomalies with fewer false negatives. Lastly, the LSTM Autoencoder's F1 score is 0.987, outperforming the Autoencoder's F1 score of 0.942. This comprehensive performance evaluation underscores the overall superior capability of the LSTM Autoencoder in balancing precision and recall for anomaly detection tasks, making it a more robust model for time-series anomaly detection.

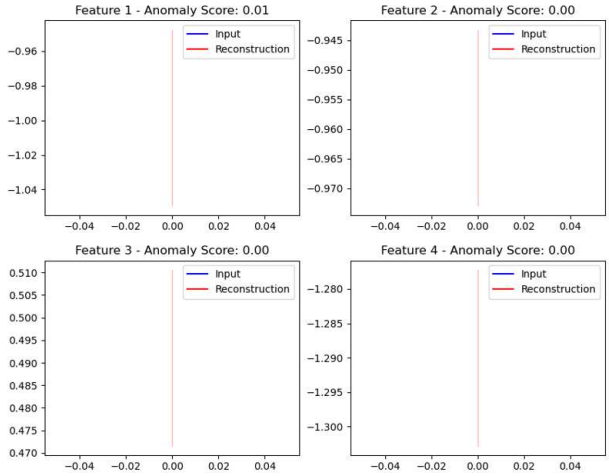

Fig. 13 GUI output with normal data

Fig. 13 and Fig. 14 represent the GUI to visualize the standard data and abnormal data detection from the deployed models, respectively. As can be seen from the figures, the model can efficiently detect the abnormal conditions for each feature and represent the input and its reconstructed output with an anomaly score. When there is no anomaly condition, the model shows an empty window with an anomaly score for each input value.

IV. Conclusion

This work represents a comparative analysis of deep learning models, specifically the Autoencoder and LSTM Autoencoder, for anomaly detection in time series data. We evaluated these models on temperature and sound data, demonstrating their capability to identify anomalies. The LSTM Autoencoder significantly outperformed the standard Autoencoder, achieving 99% accuracy and excelling in precision, recall, and F1-score. This model's robustness and reliability make it suitable for real-time anomaly detection. The results underscore the practical applicability of LSTM Autoencoder in preventing system failures and critical losses across various domains.

Acknowledgment

This research was supported by the Regional Innovation Strategy (RIS) through the National Research Foundation of Korea (NRF), which is funded by the Ministry of Education (MOE) (2021RIS-002), and the Technology Development Program (RS-2023-00266141), which is supported by the Ministry of SMEs and Startups (MSS, Korea).

REFERENCES

[1] H. Yang, S. Kumara, S. T. S. Bukkapatnam, and F. Tsung, "The internet of things for smart manufacturing: A review," *IISE Transactions*, vol. 51, no. 11, pp. 1190–1216, May 2019, doi:10.1080/24725854.2018.1555383.

[2] A. Khudoyberdiev, I. Ullah, and D. Kim, "Optimization-assisted water supplement mechanism with energy efficiency in IoT based greenhouse," *Journal of Intelligent & Fuzzy Systems*, vol. 40, no. 5, pp. 10163–10182, Apr. 2021, doi: 10.3233/jifs-200618.

[3] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris, "A Review of Machine Learning and IoT in Smart Transportation," *Future Internet*, vol. 11, no. 4, p. 94, Apr. 2019, doi: 10.3390/fi11040094.

[4] A. Khudoyberdiev, S. S. R. Garnaik, H. Y. Kim, and J. Ryoo, "A Study of the Man and Unmanned Teaming System for Improving Efficiency in a Fulfillment Center," *IEEE Access*, vol. 11, pp. 139587–139600, 2023, doi: 10.1109/access.2023.3341396.

[5] O. A. Qowiy et al., "Online Real-Time Monitoring System of A Structural Steel Railway Bridge Using Wireless Smart Sensors," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 14, no. 2, pp. 381–392, Apr. 2024, doi:10.18517/ijaseit.14.2.19291.

[6] S. Ahmad, F. Jamil, A. Khudoyberdiev, and D. Kim, "Accident risk prediction and avoidance in intelligent semi-autonomous vehicles based on road safety data and driver biological behaviours1," *Journal of Intelligent & Fuzzy Systems*, vol. 38, no. 4, pp. 4591–4601, Apr. 2020, doi: 10.3233/jifs-191375.

[7] L. Yuan, "Scientific conception of coal mine dust control and occupational safety", *Journal of China Coal Society*, vol. 45, pp. 1-7, January 2020.

[8] S. Ahmad, A. Khudoyberdiev, and D. Kim, "Towards the Task-Level Optimal Orchestration Mechanism in Multi-Device Multi-Task Architecture for Mission-Critical IoT Applications," *IEEE Access*, vol. 7, pp. 140922–140935, 2019, doi: 10.1109/access.2019.2942611.

[9] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series," *Proceedings of the VLDB Endowment*, vol. 15, no. 9, pp. 1779–1797, May 2022, doi: 10.14778/3538598.3538602.

[10] W.-H. Kim, G.-W. Kim, J. Ahn, and K. Chung, "Time-Series Data Augmentation for Improving Multi-Class Classification Performance," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 14, no. 3, pp. 887–893, Jun. 2024, doi:10.18517/ijaseit.14.3.18550.

[11] G. Li and J. J. Jung, "Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges," *Information Fusion*, vol. 91, pp. 93–102, Mar. 2023, doi:10.1016/j.inffus.2022.10.008.

[12] A. Khudoyberdiev, M. Abdul Jaleel, I. Ullah, and D. Kim, "Enhanced Water Quality Control Based on Predictive Optimization for Smart Fish Farming," *Computers, Materials & Continua*, vol. 75, no. 3, pp. 5471–5499, 2023, doi: 10.32604/cmc.2023.036898.

[13] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: A survey on the state-of-the-art" in *arXiv*:2004.00433, 2020, [online] Available: http://arxiv.org/abs/2004.00433.

[14] Y. Zhang, Y. Chen, J. Wang, and Z. Pan, "Unsupervised Deep Anomaly Detection for Multi-Sensor Time-Series Signals," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021, doi:10.1109/tkde.2021.3102110.

[15] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF," *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, May 2000, doi:10.1145/342009.335388.

[16] J. Tang, Z. Chen, A. W. Fu, and D. W. Cheung, "Enhancing Effectiveness of Outlier Detections for Low Density Patterns," *Advances in Knowledge Discovery and Data Mining*, pp. 535–548, 2002, doi: 10.1007/3-540-47887-6_53.

[17] 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, Feb. 2023, doi: 10.1109/jsen.2022.3230361.

[18] S. Chauhan and L. Vig, "Anomaly detection in ECG time signals via deep long short-term memory networks," *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–7, Oct. 2015, doi: 10.1109/dsaa.2015.7344872.

[19] S. Kanarachos, S.-R. G. Christopoulos, A. Chroneos, and M. E. Fitzpatrick, "Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and Hilbert transform," *Expert Systems with Applications*, vol. 85, pp. 292–304, Nov. 2017, doi: 10.1016/j.eswa.2017.04.028.

[20] T. Amarbayasgalan, B. Jargalsaikhan, and K. H. Ryu, "Unsupervised Novelty Detection Using Deep Autoencoders with Density Based Clustering," *Applied Sciences*, vol. 8, no. 9, p. 1468, Aug. 2018, doi:10.3390/app8091468.

[21] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019, doi:10.1109/access.2018.2886457.

[22] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," *2015 World Congress on Industrial Control Systems Security (WCICSS)*, pp. 45–49, Dec. 2015, doi: 10.1109/wcicss.2015.7420322.

[23] N. Sehatbakhsh, M. Alam, A. Nazari, A. Zajic, and M. Prvulovic, "Syndrome: Spectral analysis for anomaly detection on medical IoT and embedded devices," *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 1–8, Apr. 2018, doi: 10.1109/hst.2018.8383884.

[24] M. Xie, J. Hu, X. Yu, and E. Chang, "Evaluating Host-Based Anomaly Detection Systems: Application of the Frequency-Based Algorithms to ADFA-LD," *Network and System Security*, pp. 542–549, 2014, doi:10.1007/978-3-319-11698-3_44.

[25] L. Gjorgiev and S. Gievska, "Time Series Anomaly Detection with Variational Autoencoder Using Mahalanobis Distance," *ICT Innovations 2020. Machine Learning and Applications*, pp. 42–55, 2020, doi: 10.1007/978-3-030-62098-1_4.

[26] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines," *IEEE Access*, vol. 9, pp. 120043–120065, 2021, doi:10.1109/access.2021.3107975.

[27] F. A. Mazarbhuiya, M. Y. AlZahrani, and L. Georgieva, "Anomaly Detection Using Agglomerative Hierarchical Clustering Algorithm," *Information Science and Applications 2018*, pp. 475–484, Jul. 2018, doi: 10.1007/978-981-13-1056-0_48.

[28] Y. Lian, Y. Geng, and T. Tian, "Anomaly Detection Method for Multivariate Time Series Data of Oil and Gas Stations Based on Digital Twin and MTAD-GAN," *Applied Sciences*, vol. 13, no. 3, p. 1891, Feb. 2023, doi: 10.3390/app13031891.

[29] H. Torabi, S. L. Mirtaheri, and S. Greco, "Practical autoencoder based anomaly detection by using vector reconstruction error," *Cybersecurity*, vol. 6, no. 1, Jan. 2023, doi: 10.1186/s42400-022-00134-9.

[30] H.-K. Shin, W. Lee, J.-H. Yun and H. Kim, "*HAI 1.0: Hil-based augmented ICS security dataset*", *Proc. 13th USENIX Workshop Cyber Security Experimentation Test*, pp. 1-5, Aug. 2020, [online] Available: https://www.usenix.org/conference/cset20/presentation/shin.