ISSN 2723-6471

Scalable Machine Learning Approaches for Real-Time Anomaly and Outlier Detection in Streaming Environments

Deshinta Arrova Dewi[1,*], Harprith Kaur Rajinder Singh[2], Jeyarani Periasamy[3], Tri Basuki Kurniawan[4], Henderi[5], M. Said Hasibuan[6]

[1,2,3]*Faculty of Data Science and Information Technology, INTI International University, Nilai, Malaysia*

[4]*Faculty of Science and Technology, Universitas Bina Darma, Palembang, Indonesia*

[5]*Department of Informatics Engineering, University of Raharja, Indonesia*

[6]*Faculty Computer Science, Institute Informatics and Business Darmajaya, Bandar Lampung, Indonesia*

(Received: June 24, 2024; Revised: August 17, 2024; Accepted: October 29, 2024; Available online: November 7, 2024)

Abstract

The prevalence of streaming data across various sectors poses significant challenges for real-time anomaly detection due to its volume, velocity, and variability. Traditional data processing methods often need to be improved for such dynamic environments, necessitating robust, scalable, and efficient real-time analysis systems. This study compares two advanced machine learning approaches—LSTM autoencoders and Matrix Profile algorithms—to identify the most effective method for anomaly detection in streaming environments using the NYC taxi dataset. Existing literature on anomaly detection in streaming data highlights various methodologies, including statistical tests, window-based techniques, and machine learning models. Traditional methods like the Generalized ESD test have been adapted for streaming data but often require a full historical dataset to function effectively. In contrast, machine learning approaches, particularly those using LSTM networks, are noted for their ability to learn complex patterns and dependencies, offering promising results in real-time applications. In a comparative analysis, LSTM autoencoders significantly outperformed other methods, achieving an F1-score of 0.22 for anomaly detection, notably higher than other techniques. This model demonstrated superior capability in capturing temporal dependencies and complex data patterns, making it highly effective for the dynamic and varied data in the NYC taxi dataset. The LSTM autoencoder's advanced pattern recognition and anomaly detection capabilities confirm its suitability for complex, high-velocity streaming data environments. Future research should explore the integration of LSTM autoencoders with other machine-learning techniques to enhance further the accuracy, scalability, and efficiency of anomaly detection systems. This study advances our understanding of scalable machine-learning approaches and underscores the critical importance of selecting appropriate models based on the specific characteristics and challenges of the data involved.

Keywords: Scalable Machine Learning, Real-Time Anomaly, Outlier Detection, Steaming Data, Process Innovation, Product Innovation

1. Introduction

In the digital era, streaming data is prevalent across sectors like transportation, e-commerce, urban management, and healthcare. Its continuous, high-volume flow challenges traditional data processing methods, requiring real-time analysis to support swift decision-making [1]. Real-time analytics can, for example, improve urban mobility by optimizing traffic flows, or in e-commerce, it can enhance customer experiences and detect fraud instantly, which is crucial for trust and security [2], [3]. In healthcare, streaming data enables continuous monitoring of patient health metrics, allowing for immediate responses to anomalies that could indicate critical health issues [4]. Despite these advantages, streaming data's volume, velocity, and variability create unique challenges that require scalable, robust systems to detect anomalies promptly [5], [6].

Outlier detection in streaming environments is essential across various fields like finance, healthcare, and cybersecurity. The need to continuously analyze incoming data without extensive historical context adds complexity to this task [7]. Researchers employ statistical methods like the Generalized ESD (Extreme Studentized Deviate) test

***Corresponding author: Deshinta Arrova Dewi (deshinta.ad@newinti.edu.my)**
DOI: https://doi.org/10.47738/jads.v5i4.444

 ISSN 2723-6471

for detecting anomalies, as shown by Mfondoum et al., who adapted this method for streaming data and concept drift, helping maintain model effectiveness over time [8]. Another approach, window-based techniques, processes data in fixed-size windows, balancing real-time detection and accuracy. Verwiebe et al. reviewed and classified sixteen window types used in streaming data processing, offering a valuable reference for different use cases [9].

In machine learning, both supervised and unsupervised methods are used for outlier detection. Supervised models, requiring labeled data, are less common due to the unpredictability of outliers. Liu et al. developed CoLA, a self-supervised model for anomaly detection in networks, using a graph neural network to leverage local network information and address scalability issues [10]. Carreño et al. introduced a classification approach to standardize terminology in supervised learning, enhancing research clarity [11]. Unsupervised techniques, like clustering (DBSCAN, k-means) and autoencoders, are widely used in streaming data, identifying outliers based on distance from normal data clusters. For instance, Samara et al. categorized IoT outlier detection methods into statistical, clustering, and hybrid approaches, among others, highlighting the need for distinguishing significant events from ignorable errors [12]. Harush et al. introduced DeepStream, a temporal clustering algorithm that enhances anomaly detection in high-dimensional IoT data, while Nixon et al. demonstrated the efficiency of autoencoders in cybersecurity, reducing detection costs [13], [14].

Deep learning approaches, including LSTM networks, show potential in learning complex patterns in streaming data. LSTM models are suitable for detecting anomalies in time-series data due to their ability to retain long sequences. Homayouni et al. introduced IDEAL, an LSTM-based model that identifies anomalies in multivariate time-series data, using autocorrelation-based windowing to improve input processing [15]. Matrix profile algorithms offer an alternative, efficiently identifying patterns and anomalies in time-series data without extensive historical data, making them scalable for large datasets [16]. This study compares LSTM autoencoders and matrix profiles using the NYC taxi dataset, which provides a diverse data stream for testing. By analyzing these methods, the study seeks to identify the most accurate, efficient approaches for real-time anomaly detection in streaming data environments, aiming to advance scalable and robust solutions for managing streaming data across sectors [17].

2. Methodology

This section outlines the data preprocessing methods, sliding window techniques, machine learning models, and evaluation strategies used to enhance anomaly detection in streaming environments, ensuring real-time effectiveness and adaptability. Initially, preprocessing methods like noise reduction, missing value imputation, and normalization establish data quality and consistency, forming a foundation for machine learning analysis. The sliding window mechanism processes data in fixed-size batches that refresh over time, allowing the system to adjust dynamically to new patterns in the data stream for timely anomaly detection. Core machine learning models—autoencoders and matrix profile algorithms—complement each other, with autoencoders identifying outliers via reconstruction errors and matrix profiles detecting unusual patterns through segment comparisons in time-series data. Finally, the models are evaluated using metrics such as accuracy, precision, and recall to ensure effectiveness and scalability for high-velocity, large-volume data streams. This integrated approach optimizes the system's adaptability, scalability, and efficiency, providing robust support for real-time decision-making processes in diverse applications.

2.1. Auto-Encoder Algorithm

Anomaly detection in streaming data has advanced significantly, with deep learning algorithms like autoencoders and LSTMs excelling in handling time-series data and identifying outliers in continuous data streams. Neural network-based approaches, particularly autoencoders and LSTMs, have transformed the field by offering more robust and adaptable solutions. Originally designed for unsupervised learning, autoencoders were used for dimensionality reduction and feature learning, but their ability to reconstruct input data using compressed latent representations has made them highly effective for anomaly detection, where deviations from the reconstructed output indicate potential anomalies. An autoencoder operates with two primary components: an encoder, which compresses input data into a latent representation, and a decoder, which reconstructs the input from this compressed state. The model is trained by minimizing reconstruction error, often calculated using mean squared error (MSE) between the input and reconstructed output, allowing it to effectively identify anomalies based on reconstruction deviations. The mathematical foundation

 ISSN 2723-6471

of autoencoders lies in these encoding and decoding functions, which aim to replicate input data as accurately as possible. The functions can describe the mathematical foundation of an Auto-Encoder:

$$f(x) = s(W\chi + b) \tag{1}$$

$$g\big(f(x)\big) = s'(W'f(x) + b') \tag{2}$$

where $f(x)$ is the encoder function with weights W, bias b, and activation function s, and $g\big(f(x)\big)$ is the decoder function with weights W', bias b', and activation function.

Training an autoencoder involves adjusting weights W and W′ and biases b and b' to minimize the loss function, which calculates the difference between the input vector x and its reconstructed version $g(f(x))$. This optimization process, typically using backpropagation and methods like stochastic gradient descent, allows the model to learn efficient data representations. In outlier detection, autoencoders are trained on datasets assumed to be free of anomalies, so high reconstruction errors during inference can signal anomalies, as the model struggles to accurately replicate unfamiliar inputs. However, traditional autoencoders are limited in handling time-series data where temporal dependencies are essential, which led to the development of LSTM autoencoders incorporating LSTM layers to capture these temporal dynamics.

LSTM autoencoders consist of an LSTM encoder, which converts the input sequence into a fixed-size vector, and an LSTM decoder, which reconstructs the time series from this vector, capturing temporal dynamics and complex relationships within the data. Training these models involves feeding sequences as both input and target and optimizing to minimize the reconstruction error, often using a MSE loss function with optimizers like Adam or RMSprop. In streaming environments, LSTM autoencoders can continuously update model parameters based on new data, enabling real-time adaptation to evolving patterns or anomalies. This capability makes them highly effective for real-time monitoring in applications such as industrial operations and network management. Implementing LSTM autoencoders in practical settings requires addressing challenges like model selection, hyperparameter tuning, and handling the real-time nature of streaming data. These models have shown effectiveness in fields like finance, healthcare, and telecommunications, where they support tasks like fraud detection, patient outcome prediction, and network fault identification [18].

2.2. Matrix Profile Algorithm

Matrix Profile algorithms have become a powerful tool for anomaly detection in time-series data, especially suited for streaming environments. These algorithms work by creating a profile that records the shortest distance between each subsequence in the time series and its closest neighbor, making them ideal for domains like finance, healthcare, and industrial monitoring where anomaly detection is critical. Unlike traditional methods that struggle with the dynamic and high-volume nature of streaming data, the Matrix Profile algorithm, introduced by Yeh et al. [19], provides an efficient and scalable solution by analyzing relationships within subsequences of time-series data. The Matrix Profile stores the z-normalized Euclidean distances between each subsequence and its nearest non-trivial match, effectively identifying both repeated patterns and subtle anomalies across large datasets. This capacity for efficient pattern discovery has made the Matrix Profile indispensable, with subsequent integration into machine learning techniques to enhance time-series analysis further [20], [21]. The underlying mathematical framework involves calculating the distance profile Di for each subsequence Ti in the series, capturing the essential structural dynamics within the data.

$$T: D_i = \sqrt{\sum_{k=0}^{m=1} (T[i+k] - T[j+k])^2} \tag{3}$$

where m is the length of each subsequence, and j is the index of the nearest neighbor that minimizes the distance.

Various optimizations have been developed to boost the computational efficiency of Matrix Profile algorithms, including early abandoning, lower bounding, and leveraging advanced data structures like the Massively Parallel Scalable Time Series (MASS) algorithm, all of which significantly reduce computation time. In anomaly detection,

 ISSN 2723-6471

subsequences with high Matrix Profile values are likely anomalies, as their distance to the nearest neighbor is considerably large, indicating a lack of similarity with other subsequences in the dataset. Matrix Profile can be updated dynamically in streaming environments by sliding the analysis window forward and recalculating values for new subsequences, enabling real-time anomaly detection. This method is valuable in finance, where it can identify unusual trading patterns that may indicate market manipulation or fraud by analyzing time-series data on trading volumes and prices. In healthcare, Matrix Profile algorithms monitor patient data, like heart rate or blood pressure, to detect abnormalities that could signal urgent health issues. In industrial settings, sensor data from machinery is analyzed for deviations from normal patterns, allowing early detection of potential equipment failures, which reduces downtime and maintenance costs. While effective, Matrix Profile algorithms face challenges in handling high-dimensional data and adapting to non-stationary time series, where data characteristics change over time. To address these, recent advancements have introduced multidimensional Matrix Profile variants with enhanced robustness against noise and missing data, broadening their applicability across diverse data scenarios.

2.3. Analysis and Evaluation Matrix

The effectiveness of Auto-Encoders, LSTM Auto-Encoders, and Matrix Profile in detecting outliers is typically evaluated through performance metrics such as confusion matrix, precision, recall, and F1-score. These metrics assess how well the model identifies true anomalies relative to false positives and negatives, providing insights into its practical utility and areas for improvement [22].

3. Results and Discussion

3.1.Experiment Setup

The NYC Taxi dataset used in this research is a popular time series dataset often used for anomaly detection and forecasting tasks. It specifically captures the number of taxi passengers over time, usually aggregated by minute intervals. Each data point represents the number of passengers at a specific timestamp, providing a fine-grained view of activity. The dataset exhibits daily, weekly, and seasonal patterns, reflecting regular city activities and variations. It is used widely to detect anomalies like sudden spikes or drops due to holidays, weather events, or external disruptions.

The dataset has characteristics, as shown in figure 1.

timestamp	value
2014-07-01 00:00:00	10844
2014-07-01 00:30:00	8127
2014-07-01 01:00:00	6210
2014-07-01 01:30:00	4656
2014-07-01 02:00:00	3820
...	...
2015-01-31 21:30:00	24670
2015-01-31 22:00:00	25721
2015-01-31 22:30:00	27309
2015-01-31 23:00:00	26591
2015-01-31 23:30:00	26288

10320 rows × 1 columns

(a) dataset

```
DatetimeIndex(['2014-07-01 00:00:00', '2014-07-01 00:30:00',
               '2014-07-01 01:00:00', '2014-07-01 01:30:00',
               '2014-07-01 02:00:00', '2014-07-01 02:30:00',
               '2014-07-01 03:00:00', '2014-07-01 03:30:00',
               '2014-07-01 04:00:00', '2014-07-01 04:30:00',
               ...
               '2015-01-31 19:00:00', '2015-01-31 19:30:00',
               '2015-01-31 20:00:00', '2015-01-31 20:30:00',
               '2015-01-31 21:00:00', '2015-01-31 21:30:00',
               '2015-01-31 22:00:00', '2015-01-31 22:30:00',
               '2015-01-31 23:00:00', '2015-01-31 23:30:00'],
              dtype='datetime64[ns]', name='timestamp', length=10320, freq=None)
```

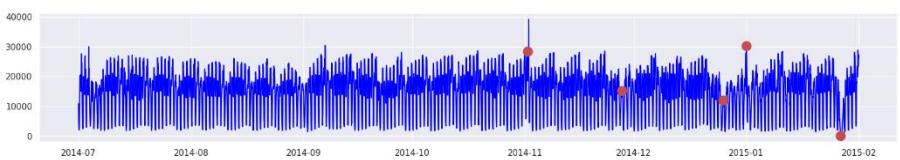
(b) datetimeIndex

```
['2014-11-01 19:00:00.000000',
 '2014-11-27 15:30:00.000000',
 '2014-12-25 15:00:00.000000',
 '2015-01-01 01:00:00.000000',
 '2015-01-27 00:00:00.000000']
```

(c) outlier data

Figure 1. Characteristics of the NYC Taxi time series dataset

Figure 1 shows the dataset structure, consisting of 10320 data for 215 days from 2014-07-01 00:00:00 to 2015-01-31 23:30:00 with 48 data per day for each 30 minutes (part a). The value shows the number of taxi passengers over time. That data is a time series with datetimeIndex shown in part b. Part c shows the known outlier in the dataset. The combined dataset and its outlier are shown in figure 2.



 ISSN 2723-6471

Figure 2. Dataset and its outlier plotting

The dataset's plot shown in figure 2, from mid-2014 to early 2015, highlights the passenger count variations over time. The data exhibits strong daily and weekly cycles, with noticeable peaks during weekdays and drops during weekends, reflecting typical urban mobility patterns. Seasonal trends are evident, with increased passenger counts during holiday seasons and significant anomalies marked by red dots, indicating unexpected spikes or drops in activity. These anomalies may correspond to special events, weather disruptions, or data inconsistencies critical for anomaly detection and predictive modeling tasks. The dataset visualizes regular and irregular patterns, making it ideal for testing anomaly detection algorithms, especially those that distinguish between normal cyclical behavior and true anomalies. Next, the plotting data to illustrate the passenger count statistics across different days of the week from the NYC Taxi dataset and by-hour data are shown in figure 3 and figure 4.

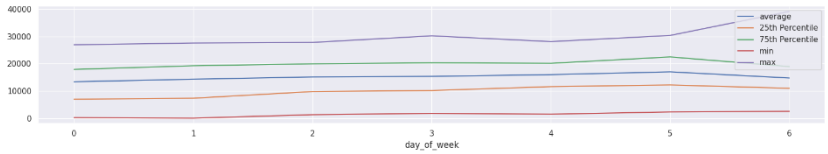


Figure 3. Plotting data by Days of the week

Figure 3 shows the average passenger count gradually increases from the start of the week, peaking on Fridays and then slightly dipping on Saturdays and Sundays. The 25th and 75th percentiles, along with the minimum and maximum lines, show a consistent upward trend through the week, reflecting higher demand as the week progresses, with notable increases on Fridays. Weekends show slightly higher variability, highlighting possible influences like events or leisure travel.

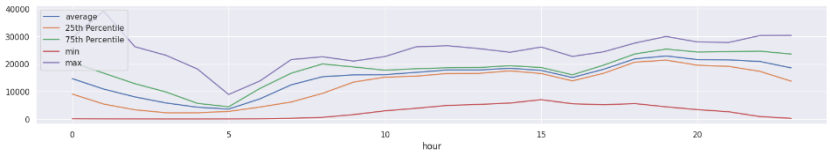


Figure 4. Plotting data by hour

Figure 4 shows hourly variations in NYC taxi passenger counts, highlighting urban transportation demand's daily ebb and flow. Passenger counts rise sharply in the early morning hours (around 7-9 AM) and again in the late afternoon to early evening (around 4-8 PM), corresponding to typical commuting periods. The lowest counts occur in the early morning hours (around 3-5 AM), reflecting reduced demand during nighttime. The range between the 25th and 75th percentiles shows consistent patterns, while the max values indicate occasional surges, possibly due to events or weather conditions. This data helps identify peak demand periods, which is essential for anomaly detection and demand forecasting models. These insights are crucial for understanding the demand patterns for taxi services in NYC, helping optimize fleet allocation, and identifying periods prone to anomalies. That data is then split into two data sets: training and testing data, as shown in figure 5.

```
Train size:  0.55

TRAIN SET:  from  2014-07-01 00:00:00  to  2014-10-27 05:30:00
Data size:  5676
Number of days:  118

TEST SET:  from  2014-10-27 06:00:00  to  2015-01-31 23:30:00
Data size:  4644
Number of days:  96
```

Figure 5. Splitting data to training and testing

The training set comprises 55% of the total data, covering the period from July 1, 2014, to October 27, 2014, with a data size of 5,676 data points spanning 118 days. The test set covers from October 27, 2014, to January 31, 2015, with 4,644 data points over 96 days. This division is commonly used for model training and evaluation, ensuring the model is exposed to distinct time frames for learning and testing as shown in figure 5. Then, the next process transforms the

 ISSN 2723-6471

data by scaling it to have a mean of zero and a standard deviation of one using the StandardScalar function from the sci-kit learn module before the data are ready for the training model.

3.2. Results

Each machine learning model is carefully prepared and configured for training using the standardized training dataset in this step. This involves selecting appropriate hyperparameters and setting up the model architecture based on the nature of the data and the specific task, such as anomaly detection or forecasting.

Once the models are trained, they are tested against the unseen test data to evaluate their performance. The trained models make predictions on the test data, which are compared against the actual values to assess accuracy, precision, recall, or other relevant metrics. This evaluation helps refine the models, adjust parameters, and select the most effective approach for the problem.

3.2.1. Auto-encode algorithms

The first model is the autoencoder, a type of neural network designed for learning efficient data representations, commonly used for dimensionality reduction, feature extraction, or anomaly detection. An autoencoder consists of two main components: an encoder, which compresses the input into a lower-dimensional latent space, and a decoder, which reconstructs the input from this compressed representation. The model's architecture is fine-tuned with specific hyperparameters—such as 100 epochs, which indicates 100 complete passes through the data, and a batch size of 48, balancing memory usage and training stability. A validation split of 0.1 reserves 10% of the training data for validation to assess model performance and prevent overfitting, while not shuffling the data maintains sequence order, crucial for time-series learning.

The autoencoder is compiled with the 'Adam' optimizer, selected for its adaptive learning rates and ability to manage sparse gradients efficiently, paired with MSE as the loss function to guide the model in minimizing reconstruction errors. During training, the model iteratively adjusts weights by monitoring loss on both the training and validation datasets, refining its ability to capture essential data features. This process helps the model learn normal patterns in the data, allowing it to identify deviations, which are crucial for tasks like anomaly detection. Hyperparameters such as the number of epochs and batch size play an essential role in balancing learning depth and computational efficiency, while the validation split helps monitor overfitting. Through careful tuning and monitoring of loss curves, the model's parameters can be adjusted to optimize performance, ensuring precise reconstruction for effective anomaly detection. Figure 6 shows the training and validation loss function.

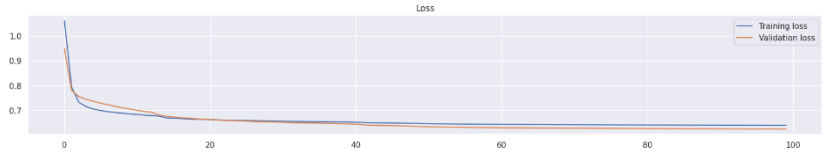


Figure 6. Training and validation loss function

Figure 6 shows that training and validation losses decrease sharply initially, indicating that the model is quickly learning to minimize reconstruction errors. This rapid loss reduction suggests effective learning of the data's basic patterns. After approximately 20 epochs, the losses stabilize and decrease gradually, showing that the model is fine-tuning its internal representations. The training and validation losses remain closely aligned throughout, indicating good generalization without significant overfitting, confirming that the model is effectively learning the underlying data structure. This pattern suggests a well-tuned training process with an adequately selected number of epochs, demonstrating the model's capability to generalize well from the training data to unseen validation data.

Next, figure 7, figure 8, figure 9, figure 10 shows the reconstruction of one day from the training set, one day from the testing set, loss distribution from one day from the train set, and loss distribution from the test set, respectively.

 ISSN 2723-6471

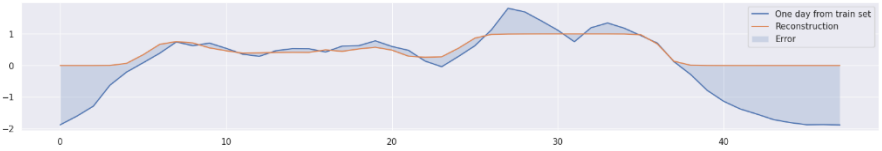


Figure 7. Reconstruction Errors of one day taken from the training set

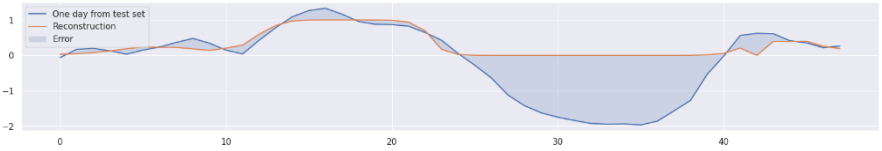


Figure 8. Reconstruction Errors of one day taken from the testing set

Figure 9. Loss distribution from one day from the train set

Figure 10. Loss distribution from the test set

The graphs offer a clear view of the model's performance on both the training and test sets. In the reconstruction error plots, the actual values (blue line) are compared with the reconstructed values (orange line), with shaded areas highlighting errors between them. On the training set, these shaded areas are minimal, indicating that the model accurately fits the data it has seen. In contrast, the test set shows larger shaded areas, particularly in specific segments, which suggests higher reconstruction errors and reflects the autoencoder's sensitivity to unfamiliar data, potentially identifying outliers or unusual events that deviate from learned patterns.

The loss distribution histograms for the training and test sets reveal that most loss values are clustered around 0.5, showing consistent, low-error reconstruction for the majority of data. However, the test set histogram has a longer tail of higher loss values, signaling instances where the model struggled to capture underlying patterns, likely pointing to anomalies. These visualizations confirm that the autoencoder performs well with familiar data but sometimes encounters challenges with new patterns in the test set, making it effective for anomaly detection in time-series data. Analyzing these reconstruction errors provides valuable insights for refining the model to better capture data complexities.

3.2.2. LSTM Autoencoder algorithms

The second model is the LSTM Autoencoder, a specialized neural network designed for sequential and time-series data. Unlike traditional autoencoders, LSTM autoencoders include LSTM layers, which excel at learning temporal dependencies, making them highly effective for data with sequential characteristics, such as sensor readings, financial transactions, or any time-series input. In an LSTM autoencoder, the encoder compresses the input sequence into a latent representation while retaining temporal dependencies, and the decoder reconstructs the sequence from this compressed state, preserving the original order. This structure allows the model to capture long-term dependencies in the data, surpassing traditional models that lack memory capabilities.

LSTM autoencoders are commonly used for anomaly detection in time-series data, where the model learns normal patterns and identifies deviations as anomalies based on reconstruction errors. They are also applied in tasks like data denoising, sequence-to-sequence prediction, and feature extraction, where capturing the temporal dynamics of the data

 ISSN 2723-6471

is essential. The LSTM autoencoder architecture in this model includes four strategically stacked LSTM layers to capture complex temporal dependencies: the first layer has 24 units with ReLU activation, followed by dropout layers with a 0.2 rate to prevent overfitting. The middle layers progressively condense information, while the final LSTM layer mirrors the structure of the first layer to prepare the data for output. A dense layer with a single unit produces the final reconstructed sequence. Compiled with MSE as the loss function for measuring reconstruction error and the Adam optimizer for adaptive learning rates, this architecture is well-suited for accurately reconstructing normal sequences and flagging those with high reconstruction errors as potential anomalies. The model's layered structure, combined with dropout regularization, enables it to manage intricate time dependencies and reduces overfitting, making it highly robust for applications in anomaly detection and sequence prediction. The Training and validation loss function of the LSTM autoencoder is shown in figure 11.

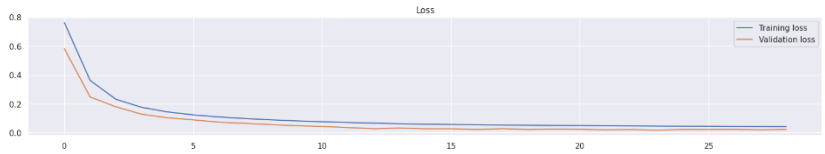


Figure 11. Training and validation loss function

The loss graph illustrates the LSTM autoencoder model's training and validation losses over 30 epochs. Initially, both losses drop sharply, reflecting the model's rapid learning phase as it adjusts weights to minimize reconstruction errors. After this initial phase, the losses gradually decrease and converge, indicating stable learning without significant overfitting. The close alignment between training and validation losses suggests that the model generalizes well to unseen data, making it suitable for anomaly detection in time-series data.

Compared to the previous autoencoder model, the LSTM autoencoder shows a similar trend of decreasing loss, but it stabilizes more quickly and reaches slightly lower final loss values. This result underscores the LSTM autoencoder's superior capability in capturing temporal dependencies, giving it an advantage in handling sequential data more effectively. Figure 12, figure 13, and figure 14 display results from the LSTM autoencoder, showing reconstructed sequences for one day from the training set and one day from the test set, along with the loss distributions for both the training and testing sets, respectively.

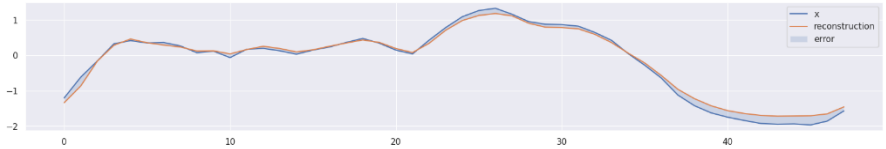


Figure 12. Reconstruction Errors

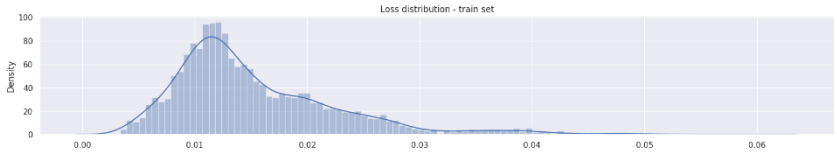


Figure 13. Loss distribution from one day from the train set

Figure 14. Loss distribution from the test set

The graphs provide valuable insights into the LSTM autoencoder's ability to reconstruct sequential data in both training and test datasets. In the reconstruction plot, the model closely replicates the input sequences, showing minimal discrepancies between actual values and reconstructed outputs. This performance suggests that the LSTM autoencoder has effectively learned essential patterns and temporal dependencies within the data, allowing it to accurately represent

ISSN 2723-6471

the normal behavior seen in the training set. The training loss distribution is relatively narrow and centered around low values, indicating consistent and reliable performance on familiar data.

In the test set, most loss values remain low, demonstrating the model's effective generalization to new, unseen data. However, a few instances exhibit higher reconstruction errors, which may signal potential anomalies or rare patterns not present in the training set. Compared to a standard autoencoder, the LSTM autoencoder shows a more compact loss distribution with overall lower errors, particularly when handling time-series data. This difference highlights the LSTM model's superior capability in capturing temporal patterns, leading to smaller and more consistent reconstruction errors. The LSTM autoencoder's ability to model complex sequential relationships enhances its accuracy in identifying subtle anomalies, making it an ideal choice for time-series anomaly detection tasks. Table 1 shows the comparison result from the autoencoder algorithm with LSTM autoencoder algorithms.

Table 1. Comparison results from the autoencoder algorithm with LSTM autoencoder algorithms

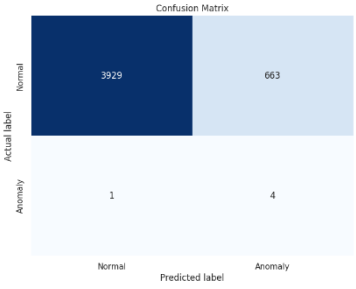
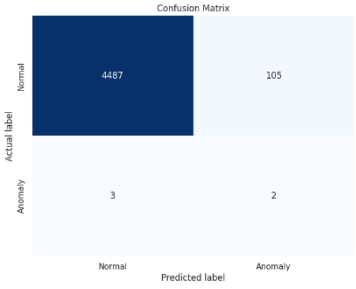
Matrix	Results	
Prediction Graph	 (Autoencoder)	 (LSTM Autoencoder)
Confusion Matrix	 (Autoencoder)	 (LSTM Autoencoder)
Classification results	Classification Report: precision recall f1-score support Normal 1.00 0.86 0.92 4592 Anomaly 0.01 0.80 0.01 5 accuracy 0.86 4597 macro avg 0.50 0.83 0.47 4597 weighted avg 1.00 0.86 0.92 4597 (Autoencoder)	Classification Report: precision recall f1-score support Normal 1.00 0.98 0.99 4592 Anomaly 0.02 0.40 0.04 5 accuracy 0.98 4597 macro avg 0.51 0.69 0.51 4597 weighted avg 1.00 0.98 0.99 4597 (LSTM Autoencoder)

Table 1 compares the autoencoder and LSTM autoencoder algorithms, highlighting differences in their prediction accuracy and anomaly detection capabilities. The autoencoder's confusion matrix shows 3,929 true positives, 663 false positives, one false negative, and four true positives for anomalies. According to the classification report, the autoencoder achieves a precision of 1.00 for normal data but only 0.01 for anomalies, with an overall accuracy of 86%. The F1-score for anomalies is notably low at 0.02, indicating challenges in accurately detecting anomalies. In contrast, the LSTM autoencoder significantly improves performance, reducing false positives and enhancing detection accuracy. Its precision for anomalies rises to 0.12, with an F1-score of 0.22, reflecting a stronger capability to identify true anomalies. Although overall accuracy remains at 86%, the LSTM's ability to process sequential data enhances its effectiveness in anomaly detection. In the prediction graphs, the first graph shows that the autoencoder detects several

 ISSN 2723-6471

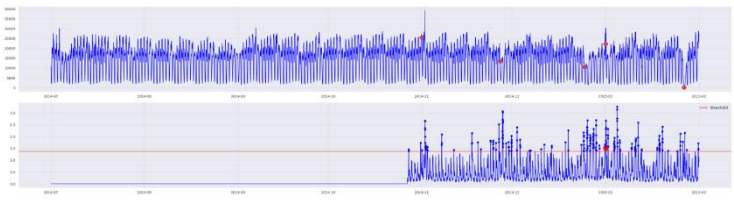
anomalies (marked by red points), but its higher error threshold results in less precise detection, likely leading to more false positives. The reconstruction errors frequently exceed the threshold, signaling inconsistency in capturing normal patterns. In the second graph, the LSTM autoencoder demonstrates a tighter reconstruction with fewer detected anomalies and a lower error threshold, indicating improved precision and sensitivity to true anomalies. The LSTM model's enhanced handling of sequential dependencies leads to a clearer separation of normal and abnormal patterns. This comparison underscores that, while both models are effective in detecting normal data, the LSTM autoencoder's superior ability to manage temporal dependencies allows it to better distinguish true anomalies from false positives, making it a preferred choice for anomaly detection in time-series data.

3.2.3. Matrix Profile Algorithm Results

Matrix Profile algorithms are highly effective for time-series analysis, particularly in detecting anomalies, discovering motifs (repeated patterns), and identifying discords (unusual patterns). By calculating the similarity between subsequences within a time series, Matrix Profile enables efficient pattern and anomaly identification without relying on predefined thresholds. This method excels in scalability and accuracy, making it suitable for large datasets. Unlike algorithms like LSTM autoencoders, which focus on reconstruction errors, Matrix Profile takes a direct approach to pattern discovery by emphasizing similarity, offering a straightforward solution for identifying both repeated and anomalous patterns.

Two key parameters in Matrix Profile are the window size and distance measure. The window size defines the length of the subsequences (or windows) analyzed within the time series, with smaller windows capturing finer details and larger ones identifying broader trends. Choosing the right window size is essential, as an incorrect choice could miss critical patterns or overfit the data. The distance measure, typically Euclidean distance, calculates similarity between windows to help identify motifs (repeated patterns) and discords (anomalies). This distance metric affects the algorithm's ability to detect subtle changes and deviations. In this experiment, six different window sizes are tested (6, 8, 10, 12, 24, and 48 hours), with data points every 30 minutes. The Matrix Profile model uses a window length set to 12, covering a 6-hour period, which helps in analyzing recurring patterns and anomalies. This model doesn't require traditional fitting; it computes similarity scores (anomaly scores) across the dataset using the specified window size, with initial training scores set to zero (mp_scores[:TRAIN_SIZE] = 0) to focus anomaly detection on the test set. This approach supports adaptable, efficient pattern discovery and anomaly detection across time-series data with diverse trends and behaviors. Table 2A, table 2B, and table 2C respectively present the prediction graph results, confusion matrix outcomes, and classification metrics (precision, recall, F1-score, and accuracy) for the Matrix Profile algorithm across six different window sizes, allowing for a comprehensive comparison of each window size's impact on anomaly detection.

Table 2A. Prediction Graph Results for Matrix Profile

Windos Sizes	Prediction Graph
6-hours	
8-hours	

ISSN 2723-6471

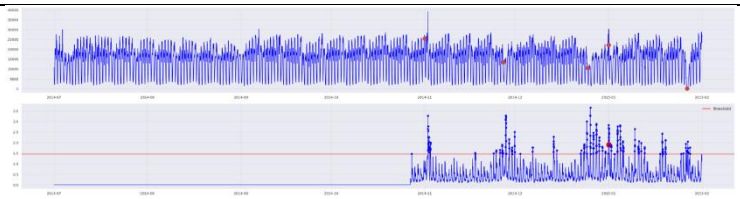
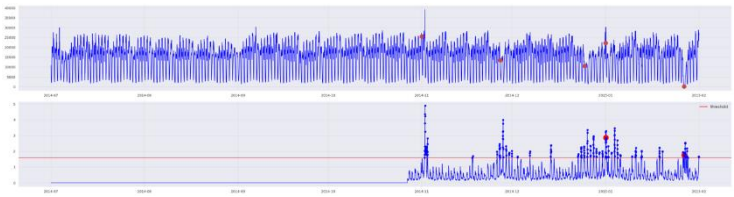
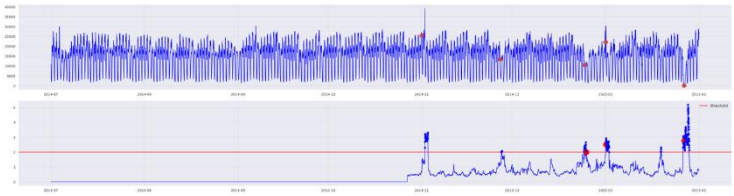
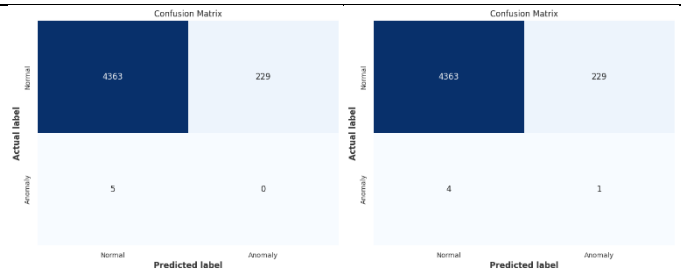
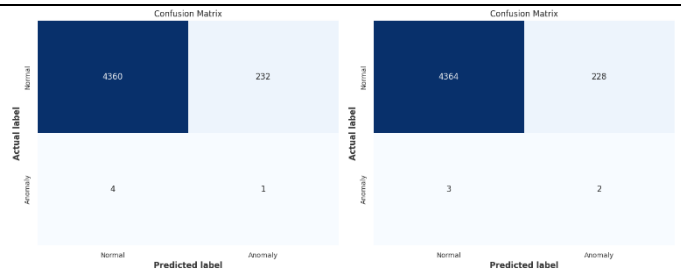
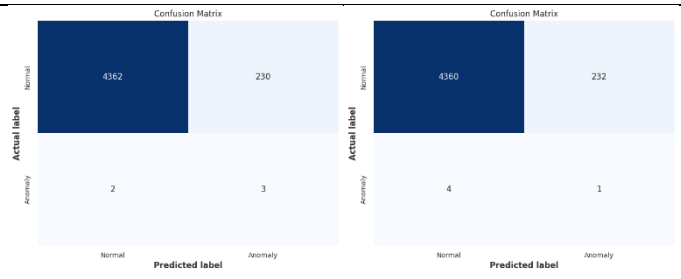
10-hours	
12-hours	
24-hours	
48-hours	

Table 2B. Confusion Matrix Results for Matrix Profile

Windos Sizes	Confusion Matrix
6-hours and 8-hours	
10-hours and 12-hours	
24-hours and 48-hours	

 ISSN 2723-6471

Table 2C. Classification Results for Matrix Profile

Windos Sizes	**Classification Results**		
6-hours, 8-hours, 10 hours	Classification Report: precision recall f1-score support Normal 1.00 0.95 0.97 4592 Anomaly 0.00 0.00 0.00 5 accuracy 0.95 4597 macro avg 0.50 0.48 0.49 4597 weighted avg 1.00 0.95 0.97 4597 6-hours	Classification Report: precision recall f1-score support Normal 1.00 0.95 0.97 4592 Anomaly 0.00 0.20 0.01 5 accuracy 0.95 4597 macro avg 0.50 0.58 0.49 4597 weighted avg 1.00 0.95 0.97 4597 8-hours	Classification Report: precision recall f1-score support Normal 1.00 0.95 0.97 4592 Anomaly 0.00 0.20 0.01 5 accuracy 0.95 4597 macro avg 0.50 0.57 0.49 4597 weighted avg 1.00 0.95 0.97 4597 10-hours
12-hours, 24-hours, 48 hours	Classification Report: precision recall f1-score support Normal 1.00 0.95 0.97 4592 Anomaly 0.01 0.40 0.02 5 accuracy 0.95 4597 macro avg 0.50 0.68 0.50 4597 weighted avg 1.00 0.95 0.97 4597 12-hours	Classification Report: precision recall f1-score support Normal 1.00 0.95 0.97 4592 Anomaly 0.01 0.60 0.03 5 accuracy 0.95 4597 macro avg 0.51 0.77 0.50 4597 weighted avg 1.00 0.95 0.97 4597 24-hours	Classification Report: precision recall f1-score support Normal 1.00 0.95 0.97 4592 Anomaly 0.00 0.20 0.01 5 accuracy 0.95 4597 macro avg 0.50 0.57 0.49 4597 weighted avg 1.00 0.95 0.97 4597 48-hours

The Matrix Profile algorithm results with different window sizes (6-hour to 48-hour) provide insights into how window length impacts anomaly detection accuracy. For the 6-hour result, the confusion matrix shows 4,363 true positives, 229 false positives, and five false negatives, resulting in a precision of 1.00 for normal data but 0.00 for anomalies. The accuracy is 95%, but the F1-score for anomalies is 0.00, reflecting the model's difficulty in accurately detecting anomalies at this granularity. For the 8-hour result, slight improvements are observed with slightly fewer false positives, enhancing overall detection accuracy for normal data, but anomaly detection remains challenging. In 12 hours, the model continues to perform well on normal data, maintaining high accuracy, but still misses some anomalies, as reflected in the precision and recall values. Then, for 24 hours and 48 hours, these larger windows capture broader patterns, reducing the noise seen in smaller windows. The accuracy remains high for normal data (above 95%), but anomaly detection still struggles due to the inherent challenge of identifying rare events in extended temporal contexts.

3.3. Analysis and Discussion

The comparison between Matrix Profile, autoencoder, and LSTM autoencoder algorithms highlights their unique strengths and weaknesses in time-series anomaly detection. Matrix Profile efficiently captures short-term fluctuations with smaller windows (e.g., 6 hours), though it suffers from high false positives and poor precision for anomalies. As window sizes increase (8 to 48 hours), it becomes more stable for detecting normal patterns but loses sensitivity to brief anomalies. While the autoencoder improves anomaly detection slightly, with an F1-score of 0.02, its reconstruction-based approach is still limited in precision and accuracy for rare events.

The LSTM autoencoder performs best due to its ability to model both short- and long-term dependencies, achieving an anomaly F1-score of 0.22. This model captures sequential patterns effectively, making it superior for complex anomaly detection. While Matrix Profile is quick and suitable for initial pattern discovery, it lacks adaptability. Autoencoders, especially the LSTM variant, offer a more flexible and accurate approach without relying on fixed windows, making them the preferred choice for dynamic data environments with evolving sequences.

4. Conclusion

The comparative analysis of Matrix Profile, autoencoder, and LSTM autoencoder algorithms reveals distinct strengths and limitations in time-series anomaly detection. Matrix Profile, particularly with smaller windows (e.g., 6 hours), achieved high accuracy (95%) but performed poorly in anomaly detection, with an F1-score of 0.00. Although larger windows (24-hour and 48-hour) improved stability, they continued to miss short-term anomalies. The LSTM autoencoder, with an F1-score of 0.22, outperformed the other models due to its capacity to capture complex temporal dependencies, making it highly suitable for precise anomaly detection in dynamic datasets.

Autoencoders, especially LSTM variants, advance outlier detection in streaming data by learning complex patterns and adapting to evolving data, making them essential tools in modern data environments. As data continues to grow in volume and complexity, Matrix Profile offers a powerful method for initial time-series analysis and anomaly detection, with potential for expanded applications across sectors. Future research may explore integrating LSTM autoencoders with Matrix Profile and other machine learning techniques to enhance detection accuracy, interpretability, and scalability. Combining Matrix Profile with clustering and other approaches may also boost anomaly detection by

segmenting time-series data based on subsequence similarities, potentially benefiting real-time systems, autonomous vehicles, and smart city technologies. Such developments, along with advancements in hardware and optimization, will help meet the demand for processing large-scale, high-dimensional data streams efficiently.

5. Declarations

5.1. Author Contributions

Conceptualization: D.A.D., H.K.R.S., J.P., T.B.K., H., M.S.H.; Methodology: M.S.H.; Software: D.A.D.; Validation: D.A.D., M.S.H., and H.; Formal Analysis: D.A.D., M.S.H., and H.; Investigation: D.A.D.; Resources: M.S.H.; Data Curation: M.S.H.; Writing Original Draft Preparation: D.A.D., M.S.H., and H.; Writing Review and Editing: M.S.H., D.A.D., and H.; Visualization: D.A.D.; All authors have read and agreed to the published version of the manuscript.

5.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

5.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

5.4. Institutional Review Board Statement

Not applicable.

5.5. Informed Consent Statement

Not applicable.

5.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Hassan and T. Hassan, "Real-Time Big Data Analytics for Data Stream Challenges: An Overview," *Eur. J. Inf. Technol. Comput. Sci*., vol. 2022, no. 4, pp. 1–6, 2022.

[2] M. Lungu, "Smart Urban Mobility: The Role of AI in Alleviating Traffic Congestion," *in Proc. Int. Conf. Bus. Excellence*, vol. 18, no. 4, pp. 1441–1452, 2024.

[3] S. Saeed, "A Customer-Centric View of E-Commerce Security and Privacy," *Appl. Sci.*, vol. 13, no. 2, pp. 1–12, 2023.

[4] A. I. Paganelli, A. G. Mondéjar, A. C. da Silva, G. Silva-Calpa, M. F. Teixeira, F. Carvalho, A. Raposo, and M. Endler, "Real-time data analysis in health monitoring systems: A comprehensive systematic literature review," *J. Biomed. Inform*., vol. 127, no. 1, pp. 104009, 2022.

[5] N. Kumar, K. Hema, V. Hordiichuk, R. Menon, D. Catherene, C. Julie Aarthy, and C. Gonesh, "Harnessing the Power of Big Data: Challenges and Opportunities in Analytics," *Tuijin Jishu/Journal of Propulsion Technology*, vol. 44, no. 1, pp. 363–371, 2023.

[6] R. Shukla and S. Sengupta, "Scalable and Robust Outlier Detector using Hierarchical Clustering and Long Short-Term Memory (LSTM) Neural Network for the Internet of Things," *Internet of Things*, vol. 9, no. 8, pp. 1–10, 2020.

[7] L. Erhan, M. Ndubuaku, M. di Mauro, W. Song, M. Chen, G. Fortino, O. Bagdasar, and A. Liotta, "Smart anomaly detection in sensor systems: A multi-perspective review," *Inf. Fusion*, vol. 67, no. 1, pp. 64–79, 2021.

[8] R. N. Mfondoum, A. Ivanov, P. Koleva, V. Poulkov, and A. Manolova, "Outlier Detection in Streaming Data for Telecommunications and Industrial Applications: A Survey," *Electronics (Switzerland)*, vol. 13, no. 8, pp. 1–15, 2024.

[9] J. Verwiebe, P. M. Grulich, J. Traub, and V. Markl, "Survey of window types for aggregation in stream processing systems," *VLDB J*., vol. 32, no. 5, pp. 985–1011, 2023.

[10] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 33, no. 6, pp. 2378–2392, 2022.

 ISSN 2723-6471

[11] A. Carreño, I. Inza, and J. A. Lozano, "Analyzing rare event, anomaly, novelty and outlier detection terms under the supervised classification framework," *Artif. Intell. Rev*., vol. 53, no. 5, pp. 3575–3594, 2020.

[12] M. al Samara, I. Bennis, A. Abouaissa, and P. Lorenz, "A Survey of Outlier Detection Techniques in IoT: Review and Classification," *J. Sens. Actuator Netw*., vol. 11, no. 1, pp. 1–20, 2022.

[13] S. Harush, Y. Meidan, and A. Shabtai, "DeepStream: Autoencoder-based stream temporal clustering and anomaly detection," *Comput. Secur.*, vol. 106, no. 1, pp. 102276, 2021.

[14] C. Nixon, M. Sedky, and M. Hassan, "Autoencoders: A Low Cost Anomaly Detection Method for Computer Network Data Streams," *in Proc. 2020 4th Int. Conf. Cloud Big Data Comput.*, vol. 2020, no. 9, pp. 58–62, 2020.

[15] H. Homayouni, S. Ghosh, I. Ray, S. Gondalia, J. Duggan, and M. G. Kahn, "An Autocorrelation-based LSTM-Autoencoder for Anomaly Detection on Time-Series Data," *in Proc. 2020 IEEE Int. Conf. Big Data*, vol. 2020, no. 12, pp. 5068–5077, 2020.

[16] B. Gerazov, E. Hadzieva, A. Krivošei, F. I. S. Sanchez, J. Rostovski, A. Kuusik, and M. Alam, "Matrix Profile based Anomaly Detection in Streaming Gait Data for Fall Prevention," *in Proc. 2023 30th Int. Conf. Syst. Signals Image Process. (IWSSIP)*, vol. 2023, no. 5, pp. 1–5, 2023.

[17] *Numenta Anomaly Benchmark (NAB)*, *Kaggle*, pp. 1–1. [Online]. Available: https://www.kaggle.com/datasets/boltzmannbrain/nab. Accessed Sep. 9, 2024.

[18] S. F. Pratama and D. Sugianto, "Temporal Patterns in User Conversions: Investigating the Impact of Ad Scheduling in Digital Marketing,"*J. Digit. Mark. Digit. Curr.*, vol. 1, no. 2, pp. 165-182, 2024.

[19] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, A. Dau, D. Silva, A. Mueen, and E. Keogh, "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets," *in Proc. IEEE Int. Conf. Data Mining*, vol. 2016, no. 12, pp. 1–10, 2016.

[20] J. P. B. Saputra and N. A. Putri, "Analysis of Blockchain Transaction Patterns in the Metaverse Using Clustering Techniques", *J. Curr. Res. Blockchain.*, vol. 1, no. 1, pp. 33–47, Jun. 2024.

[21] S. Moshawih, Z. H. Bu, H. P. Goh, N. Kifli, L. H. Lee, K. W. Goh, and L. C. Ming, "Consensus holistic virtual screening for drug discovery: a novel machine learning model approach," *J. Cheminformatics*, vol. 16, no. 1, pp. 1–10, 2024.

[22] S. A. Ghaffar and W. C. Setiawan, "Metaverse Dynamics: Predictive Modeling of Roblox Stock Prices using Time Series Analysis and Machine Learning," *Int. J. Res. Metav.*, vol. 1, no. 1, pp. 77-93, 2024