STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **13**, March 2025, pp 1245–1265.
Published online in International Academic Press (www.IAPress.org)

Integrating LSTM and LOF: A Comprehensive Approach to Anomaly Detection in Healthcare Data

Heba Mostafa *, Mohamed Abd Allah Makhlouf, Hossam Refaat, Basel Hafiz

Department of Information System, Suez Canal University, Egypt

Abstract The Internet of Medical Things (IoMT) has grown substantially, facilitating extensive time series data accumulation within healthcare environments. Detecting anomalies within IoMT time series data is critical for identifying potential health hazards and ensuring patient safety. This study investigates the efficacy of merging Long Short-Term Memory (LSTM) neural networks with the Local Outlier Factor (LOF) algorithm for anomaly detection in time series data. LSTM networks are adept at grasping extended dependencies in sequential data, while LOF represents a potent unsupervised outlier identification technique. We introduce a unique methodology that capitalizes on the strengths of both LSTM and LOF to heighten anomaly detection accuracy. The proposed technique undergoes assessment via experiments on actual IoMT datasets including WUSTL-EHMS and Thyroid_Diff, demonstrating consistent performance across diverse healthcare scenarios. A real-time simulation was conducted to assess the feasibility of deploying the framework in practical IoMT environments. Through comprehensive experimentation and analysis, we show its effectiveness. The results of the proposed method reveal a promising ability to precisely pinpoint anomalies, offering a valuable resource to healthcare professionals to quickly identify irregular patient conditions.

Keywords Anomaly detection, Long Short-Term Memory (LSTM), Local Outlier Factor (LOF), Time series data, Internet of Medical Things (IoMT)

DOI: 10.19139/soic-2310-5070-2206

1. Introduction

1.1. Background

The Internet of Medical Things (IoMT) represents a transformative advancement in healthcare, enabling the interconnection of medical devices and systems for continuous monitoring, analysis, and sharing of patient health data. The vast amounts of time-series data obtained from IoMT devices can provide valuable insights into patient conditions. Anomalies, or outliers, are data points that deviate significantly from expected patterns, often signaling critical issues such as patient health deterioration, system failures, or cybersecurity threats. Accurate anomaly detection is vital for proactive measures in healthcare and other domains, including finance, industrial monitoring, and network security. However, detecting anomalies in IoMT time series data remains a challenge. This literature review investigates the combination of Long Short-Term Memory (LSTM) and Local Outlier Factor (LOF) techniques for anomaly detection in time series IoMT data. The proliferation of data across industries has made anomaly detection a crucial task for ensuring system security, reliability, and efficiency. Anomalies, or outliers, are data points that deviate significantly from the norm, often signaling critical issues like fraud, network intrusions, or system failures. Accurate anomaly detection is vital for proactive measures in diverse fields, including finance, healthcare, industrial monitoring, and cybersecurity (1)(2).

*Correspondence to: Heba Mostafa (Email: heba.mostafa8989@gmail.com).

ISSN 2310-5070 (online) ISSN 2311-004X (print)

1.2. Challenges in Anomaly Detection

Often based on statistical techniques like the Z-score and ARIMA models and rule-based systems, traditional anomaly detection methods struggle with modern datasets' complexity and evolving nature (3). Machine learning, especially deep learning, offers a promising solution by learning intricate patterns from large datasets. However, challenges such as imbalanced data and the need for real-time detection remain (1)(2). Deep learning models, including RNNs and GANs, have shown potential in addressing these challenges. They are applied in various domains, such as industrial monitoring, finance, healthcare, and network security.

1.3. Deep Learning for Anomaly Detection

LSTM networks, a type of recurrent neural network (RNN), are well-suited for processing sequential data and detecting anomalies in time series data. Their ability to learn long-term dependencies and retain information over extended periods makes them particularly effective at identifying deviations from expected patterns (4). LSTM networks excel at detecting anomalies in time series data by analyzing sequential observations and identifying deviations from expected patterns. By learning from historical data, LSTM models can predict future values and flag anomalies that significantly diverge from these predictions. Despite their success, standalone LSTM models face limitations in high-dimensional datasets, where they may overfit noise or spurious correlations, leading to false positives. These challenges can be mitigated through techniques such as regularization, data augmentation, and carefully designed architectures. However, further improvements are needed to enhance their reliability and interoperability.

1.4. Unsupervised anomaly detection

In addition to neural networks, unsupervised anomaly detection algorithms such as Local Outlier Factor (LOF) have gained popularity for their ability to identify outliers in high-dimensional data, including time series. LOF measures the local density deviation of a data point concerning its neighbors, allowing it to detect anomalies based on deviations in local density patterns (5)(6). Unlike supervised methods, LOF does not require labeled data, which is advantageous in healthcare environments where anomalies are rare and labels may not always be available.

1.5. Hybrid Approaches: Combining LSTM with LOF

To overcome the limitations of LSTM alone, hybrid models Combining LSTM and LOF offer a promising approach to anomaly detection in time series data, leveraging the strengths of deep learning for capturing complex temporal patterns and the robustness of outlier detection algorithms for identifying anomalies. The LOF algorithm, which identifies anomalies based on local density, can be integrated with LSTM to leverage their strengths. This combination is particularly effective in detecting both global and local anomalies, enhancing the overall robustness and accuracy of the detection process. Additionally, integrating LOF provides interpretability by assigning density-based anomaly scores, aligning with the growing demand for explainable AI in healthcare applications.

1.6. Objectives of the study

The primary objective of this study is to develop and evaluate a hybrid anomaly detection model that combines the strengths of LSTM and LOF. This model addresses the challenges of high-dimensional, imbalanced datasets, real-time detection requirements and provides accurate, interpretable results. Specifically, the study aims to:

- Develop a robust LSTM model for capturing temporal dependencies in multivariate time series data.
- Integrate LOF for local anomaly detection, enhancing the model's ability to detect subtle anomalies and reduce false positives.
- Evaluate the model's performance on real-world datasets (e.g., WUSTL-EHMS and Thyroid_Diff), focusing on key metrics such as accuracy, precision, recall, F1-score, and ROC AUC.
- Address common issues such as class imbalance and overfitting through techniques like SMOTE, regularization, and cross-validation.
- Conduct real-time simulations to assess the framework's feasibility for practical IoMT deployments.

1.7. Structure of the paper

The remainder of this paper is structured as follows: **Section 2** presents the related works. **section 3** includes the detailed methodology, including data preprocessing, model design, and evaluation techniques. **Section 4** discusses the results and compares the performance of the proposed hybrid model against traditional methods. **Section 5** concludes the paper with a summary of findings and potential directions for future research.

2. Related Work

Ghubaish et al.(7) developed a real-time Enhanced Healthcare Monitoring System (EHMS) testbed capable of monitoring patients' biometrics and gathering network flow metrics. They generated a dataset comprising over 16,000 normal and attacked healthcare data records. They chose four ML methods for attack detection: Random Forest (RF), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Artificial Neural Networks (ANN). The results indicate performance improvements ranging from 7% to 25% in certain cases, demonstrating the robustness of the proposed system in effectively detecting intrusions.

Wagan et al.(8) proposed a novel multi-modal approach called the Duo-Secure IoMT framework, this technique leverages dynamic Fuzzy C-Means clustering and customized Bidirectional Long Short-Term Memory (Bi-LSTM) algorithms to detect abnormal pattern uncertainties in the network and perform data evaluation tasks concurrently. The performance evaluation shows that the proposed model evaluates a) the prediction of heart issues and b) the identification of network intrusions with an accuracy of 92.95% and multimodal joint accuracy of 89.67% in the WUSTL EHMS-2020 dataset.

Alani et al.(9) present a deep neural network for detecting attacks on (IoMT) devices. It relies on an explainable Deep Neural Network (DNN), which analyzes information extracted from the network to detect and block malicious traffic. This DNN solution is further elucidated through the application of Shapley Additive Explanation (SHAP), facilitating the identification of how selected features influence the classifier's decision-making process. The system was evaluated using the WUSTL EHMS-2020 dataset, demonstrating an accuracy of 97.578% and a false-positive rate of only 3.12%.

Kilincer et al.(10) developed a cyber-attack and anomaly detection model utilizing recursive feature elimination (RFE) in conjunction with a multilayer perceptron (MLP). The RFE method employed logistic regression (LR) and extreme gradient boosting regression (XGBRegressor) kernel functions to select optimal features. Hyperparameter optimization was conducted to adjust MLP parameters, and performance evaluations were carried out using a 10-fold cross-validation approach. They found by reviewing approximately 80 literature studies that support vector machines (SVM) and random forest (RF) methods are widely used in the field. Moreover, extreme gradient boosting (XGBoost), neural networks (NN), and recurrent neural networks (RNN) were identified as particularly effective compared to other methods.

Tauqeer et al.(11) Three machine learning algorithms, namely Random Forest, Gradient Boosting, and Support Vector Machine (SVM), are employed for cyberattack detection. These machine-learning models are well-suited for this task. The proposed models are assessed using the WUSTL-EHMS-2020 dataset, which includes man-in-the-middle, data injection, and spoofing attacks. The proposed model achieved 96.9% accuracy.

Ravi et al.(12) introduces a deep learning approach for network-based intrusion detection in (IoMT) systems, utilizing features from network flows and patient biometrics from WUSTL EHMS-2020 dataset. This method efficiently learns optimal feature representations by integrating information from both network flows and patient biometrics across multiple hidden layers of deep learning. The proposed model showed a 10-fold cross-validation accuracy of 95% on network features, 89% on patient biometrics, and 99% on combined features.

Mosaiyebzadeh et al.(13) introduces a deep neural network within the framework of federated learning (DNN-FL) to identify anomalies in healthcare traffic, which could potentially lead to security threats. The detection performance of this proposed approach is evaluated using metrics such as accuracy and precision. Validation of the DNN-FL model is conducted using two datasets: WUSTL-EHMS-2020 and ECU-IoHT. The results demonstrate an accuracy of 91.40% in detecting attacks within the first dataset and 98.47% within the second.

Lu, W. et al.(14) proposed a hybrid intrusion detection system designed to identify cyberattacks on medical devices in real time. The system integrates a logistic regression-based detector utilizing network traffic features with a gradient-boosted tree-based detector leveraging medical sensor features. Evaluation using a publicly available dataset indicates that the system achieves an accuracy of 95.4% using only 11 features, surpassing the current best accuracy of 92.98% achieved by artificial neural networks using 40 features.

Dina, A. S. et al.(15) addressed the data imbalance problem in intrusion detection for IoT, by utilizing a specialized loss function known as focal loss. This function automatically down-weights easy examples and focuses on hard negatives by facilitating dynamically scaled-gradient updates, thereby training more effective machine learning models. Their approach is implemented using two well-known deep learning (DL) neural network architectures.

Al-Hawawreh et al (16) proposed a new privacy-aware framework for storing collected IoMT data in distributed cloud nodes. This framework facilitates data fusion of heterogeneous IoMT data using differential privacy and deep learning and enhances attack detection through quantum deep learning while maintaining data privacy.

3. Methodology

This section delves into a detailed exposition of our envisioned model, delineating a meticulous process for anomaly detection in the context of the Internet of Medical Things (IoMT) datasets. The framework integrates Long Short-Term Memory (LSTM) networks for capturing temporal dependencies in sequential data with the Local Outlier Factor (LOF) algorithm, which identifies anomalies based on local density deviations. This hybrid approach leverages the complementary strengths of these techniques to enhance detection accuracy and robustness.

Algorithm 1 Hybrid LSTM + LOF Anomaly Detection

1: **Input:** Training Dataset $(X_{\text{train}}, y_{\text{train}})$, Test Dataset (X_{test})
2: **Output:** Combined Predictions $(\hat{y}_{\text{combined}})$
3: **Preprocessing:**
a. Encode categorical features using LabelEncoder.
b. Scale numerical features with StandardScaler.
c. Apply PCA $(n_{\text{components}} = 16)$.
d. Oversample minority class using SMOTE.
e. Reshape features for LSTM $(\text{samples}, \text{timesteps}, \text{features})$.
4: **Train LSTM Model:**
a. Define a sequential model with 2 LSTM layers & dropout layers.
b. Compile the model with rmsprop optimizer and binary_crossentropy loss.
c. Train the model for 20 epochs with a batch size of 64.
5: **Train LOF Algorithm:**
a. Compute residuals from LSTM predictions on training data.
b. Fit LOF using the residuals $(n_{\text{neighbors}} = 20, \text{novelty=True})$.
6: **Make Predictions:**
a. Predict test data anomalies with LSTM $(\hat{y}_{\text{LSTM}})$.
b. Predict test data anomalies with LOF $(\hat{y}_{\text{LOF}})$.
c. Combine predictions using logical OR $(\hat{y}_{\text{combined}} = \hat{y}_{\text{LSTM}} \vee \hat{y}_{\text{LOF}})$.
7: **Evaluate Performance:**
a. Calculate accuracy, precision, recall, F1-score, and ROC AUC.
8: **Return:** Combined Predictions $(\hat{y}_{\text{combined}})$.

Initially, we curate the WUSTL-EHMS-2020 dataset, comprising a comprehensive compilation of normal and anomalous records, forming the bedrock for our investigative endeavors. To demonstrate the generalizability of the framework, we include the Thyroid_Diff dataset in the evaluation, showcasing consistent performance across diverse healthcare scenarios.

The key steps of the methodology include data preprocessing, model design, training, and evaluation. Preprocessing involves label encoding, scaling, dimensionality reduction using PCA, and handling class imbalance with SMOTE. The LSTM model captures long-term temporal dependencies, while LOF identifies anomalies in the residuals of LSTM predictions. The final anomaly predictions result from a logical combination of outputs from both techniques.

Real-time simulation experiments assess the framework's feasibility for deployment in streaming data environments, measuring latency and computational efficiency.

Algorithm 1: Hybrid LSTM + LOF Anomaly Detection Framework. This algorithm outlines the step-by-step process of combining Long-Short-Term Memory (LSTM) networks and the Local Outlier Factor (LOF) algorithm for detecting anomalies in multivariate time-series datasets. It includes preprocessing, model training, and the logical combination of predictions from both techniques to achieve robust anomaly detection.

Methodological Stages:

1. **Data Preprocessing:**
   The raw datasets are subjected to a rigorous preprocessing phase that includes:

   - **Encoding categorical features** into numerical values using "LabelEncoder".
   - **Scaling numerical features** to ensure a standard distribution with a mean of 0 and standard deviation of 1 using "StandardScaler".
   - **Dimensionality reduction** using "Principal Component Analysis (PCA)" to retain the most significant components while reducing redundancy.
   - **Handling class imbalance** through "Synthetic Minority Oversampling Technique (SMOTE)", generating synthetic samples for minority classes to ensure balanced training data.

2. **Dimensionality Reduction:**
   PCA is employed to transform the high-dimensional datasets into a reduced feature space, retaining either 95% variance or a fixed number of components (e.g., 16) for consistency across datasets. This step simplifies computation and enhances the model's ability to detect subtle patterns in the data.
3. **Model Architecture and Training:**
   - **LSTM Network:**
     - The model consists of two LSTM layers with 64 and 32 units, respectively, followed by dropout layers to mitigate overfitting.
     - An output-dense layer with a sigmoid activation function is used to classify anomalies.
     - The network is trained using the "rmsprop" optimizer and "binary_crossentropy" loss function over 20 epochs with a batch size of 64.
   - **LOF Algorithm:**
     - LOF is trained on residuals from the LSTM predictions, leveraging local density deviations to identify potential outliers. It operates in novelty detection mode (novelty=True) to ensure effective handling of unseen data.
4. **Combining Predictions:** LSTM and LOF predictions are combined using a logical OR operation. An instance is classified as anomalous if either algorithm identifies it.
5. **Performance Evaluation:** The framework's performance is assessed using a comprehensive set of metrics, including accuracy, precision, recall, F1-score, and ROC AUC. Additionally, cross-dataset validation and real-time performance simulations are conducted to validate the model's generalizability and practicality.

Model Architecture: The proposed model architecture is elucidated through a comprehensive visual representation in **Figure 1**. The architectural blueprint embodies a seamless integration of preprocessing steps, model inference, and performance evaluation, encapsulating the essence of our anomaly detection framework.

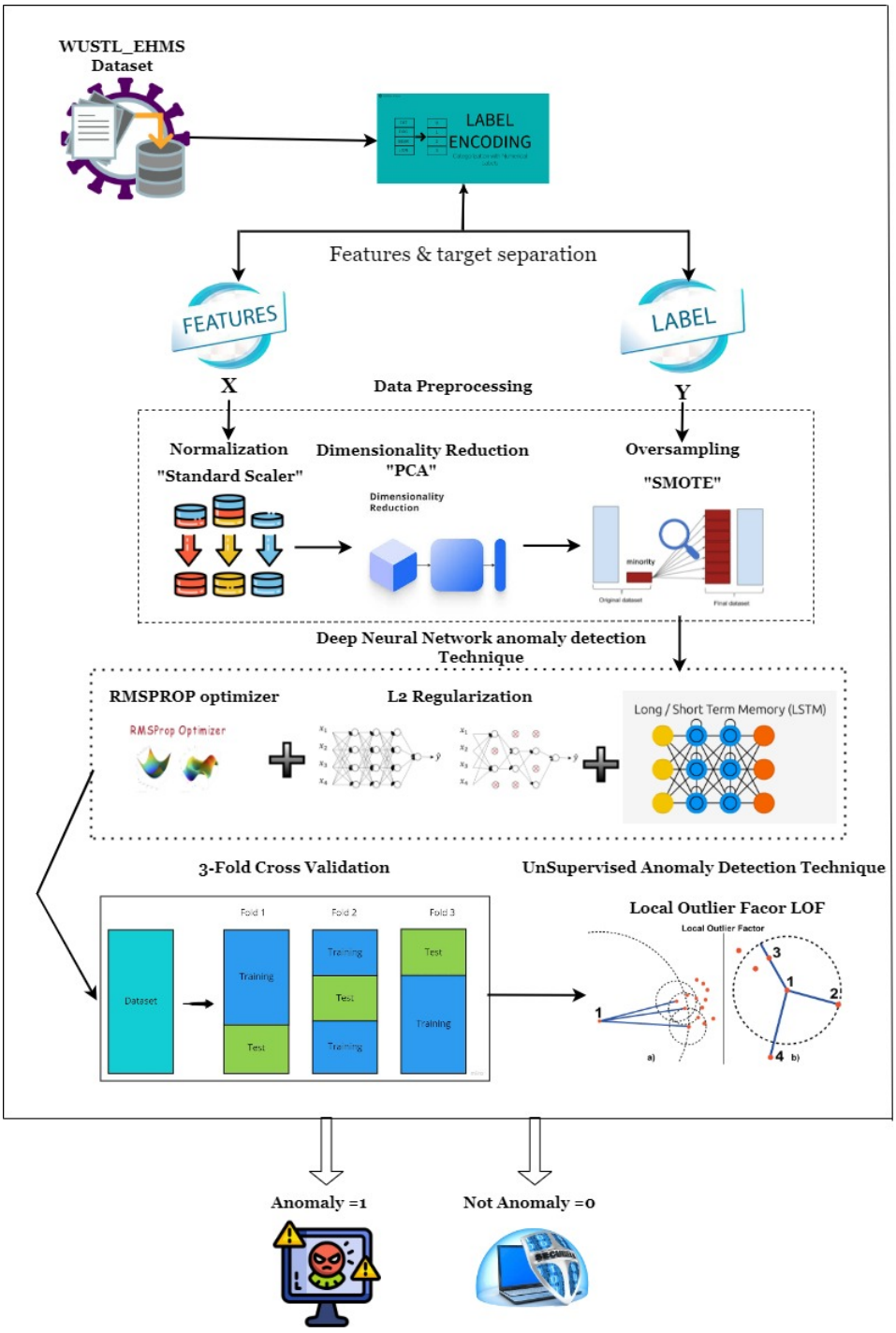


Figure 1. Architecture of the proposed model for Anomaly detection

3.1. Dataset Description

3.1.1. WUSTL-EHMS-2020 Dataset The primary dataset used in this study is the WUSTL-EHMS-2020 dataset, developed by researchers at Washington University in St. Louis. It comprises:

- 16,318 records with 44 features.
- Features include:
  - 35 network-flow metrics.
  - 8 patient biometrics (e.g., SpO2, blood pressure, ECG readings).
  - 1 label attribute indicating whether a record represents normal (0) or anomalous (1) data.
- Imbalanced class distribution:
  - 87.5% normal records.
  - 12.5% anomalous records.

This dataset includes multiple types of cyber-attacks, such as spoofing, data injection, and man-in-the-middle attacks, generated using real IoMT devices like SpO2 monitors, blood pressure measurement devices, ECG machines, and thermometers. These features make it an ideal benchmark for evaluating the proposed anomaly detection framework.

To further validate generalizability, the Thyroid_Diff dataset was used, consisting of clinical data with significant differences in feature distribution. The preprocessing pipeline for this dataset ensured compatibility with the WUSTL-EHMS dataset through consistent dimensionality reduction and scaling.

3.1.2. Thyroid Diff Dataset To validate the generalizability of the framework, the Thyroid_Diff dataset was incorporated as a secondary dataset. Unlike WUSTL-EHMS, this dataset comprises clinical data with a focus on thyroid disease diagnosis. It is characterized by the following:

- Number of Records: 7,200 samples.
- Features:
  - 21 features, including patient demographics (e.g., age, gender), laboratory test results (e.g., T3, T4, TSH levels), and diagnostic labels.
  - Features include both categorical (e.g., patient gender, referral reason) and numerical (e.g., hormone levels) data.
- Class Distribution: 85% normal records and 15% anomalous records (diagnosed with thyroid-related conditions such as hypothyroidism or hyperthyroidism).

3.1.3. Justification of Dataset Selection The use of two diverse datasets ensures a thorough evaluation of the proposed framework. The WUSTL-EHMS-2020 dataset offers a realistic and comprehensive testbed for IoMT-specific anomaly detection, while the Thyroid_Diff dataset challenges the framework's adaptability to different healthcare domains. Together, these datasets validate the framework's robustness and potential for deployment across various IoMT scenarios.

3.2. Preprocessing of dataset

Preprocessing is a crucial step that greatly influences the effectiveness and accuracy of the machine learning models used on the dataset. Several preprocessing techniques were implemented, including label encoding, feature scaling, dimensionality reduction, and oversampling.

3.2.1. Label Encoding The dataset contains various categorical and numerical features, with the target variable denoted as 'Label', indicating normal or anomalous instances. The columns are a mix of numerical and categorical data types. Here's a breakdown:

- **Numerical Columns:** The dataset has 37 numerical columns, including float64 and int64 data types. These columns likely represent various metrics like packet sizes, load, loss, rate, vital signs, etc.
- **Categorical Columns:** There are 7 categorical columns (Dir, Flgs, SrcAddr, DstAddr, Sport, SrcMac, DstMac), which include IP addresses, MAC addresses, and other textual or symbolic data.

Categorical features were transformed into numerical values using label encoding. This is essential as most machine learning algorithms require numerical input. This allows these features to be used effectively in machine learning models. The following equation represents the Label Encoder Process:

$$EncodedFeature_i = LabelEncoder(CategoricalFeature_i) \tag{1}$$

Figure 2 Illustrate how the 'Label Encoder' transformed categorical columns into numerical data types.

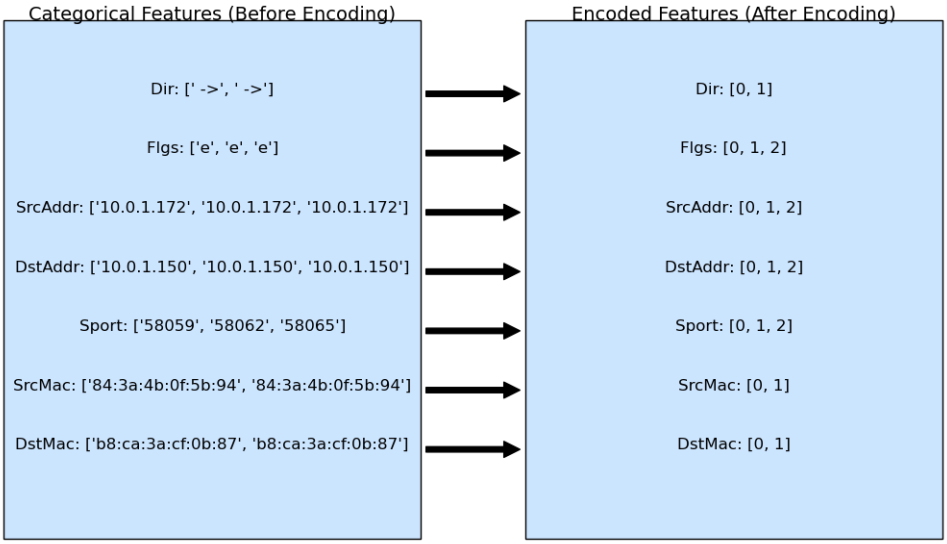


Figure 2. Label Encoder Technique

3.2.2. Feature Scaling Feature scaling is a crucial preprocessing step in machine learning, particularly for models like LSTM that are sensitive to the scale of input data. According to (17), normalizing the data used for training and testing neural networks is a widely accepted best practice. This feature-wise normalization helps neural networks perform more effectively and prevents them from being adversely affected by the varying scales of different input features.

By standardizing or normalizing features to a common range, feature scaling helps ensure that all features contribute equally to the model's predictions, preventing certain features from dominating the learning process. The input data was standardized using **'StandardScaler'** to ensure that features with varying magnitudes do not

bias the model. This transformation rescales the features with a mean of zero and a standard deviation of one. This is achieved by subtracting the mean of each feature from the corresponding values and then dividing by the standard deviation as follows:

$$Z = \frac{X - \mu}{\sigma} \tag{2}$$

Where:

- X is the original feature value.
- μ is the Mean of the feature.
- σ is the Standard Deviation of the feature.
- Z is the Scaled value.

Figure 3 Illustrate how the 'Standard Scaler' normalized all features to a common range ensuring that all features contribute equally to the model's predictions:

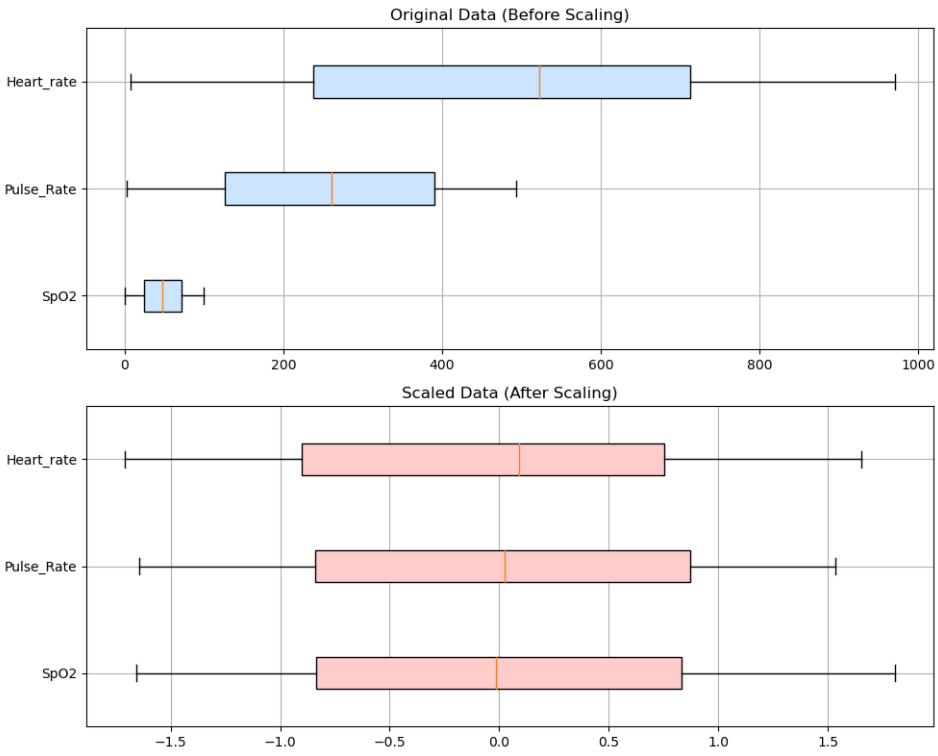


Figure 3. Feature Scaling Technique

3.2.3. Dimensionality Reduction To reduce the dimensionality of the dataset and eliminate multicollinearity, Principal Component Analysis (PCA) was applied. PCA is a powerful dimensionality reduction technique that transforms a high-dimensional dataset into a lower-dimensional representation while preserving the most important information, simplifying analysis, and potentially improving model performance. In our model, PCA was configured to retain 95% of the data variance as mentioned in equation 3(18).

$$X_{PCA} = PCA(X_{scaled}, n_{components} = 0.95) \tag{3}$$

$$Var(Retained) \approx 0.95$$

How does PCA work?

1. Center the data: Subtract the mean from each feature.
2. Calculate the covariance matrix: Compute the covariance matrix of the centered data.
3. Eigenvalue decomposition: Find the eigenvalues and eigenvectors of the covariance matrix.
4. Select principal components: Choose the eigenvectors corresponding to the largest eigenvalues. These eigenvectors represent the principal components.
5. Project data onto principal components: Project the original data onto the selected principal components.

Benefits of PCA:

- Dimensionality reduction: Reduces the number of features, simplifying analysis and potentially improving model performance.
- Feature extraction: Creates new features that are linear combinations of the original features, often capturing the most important patterns in the data.
- Visualization: Can visualize high-dimensional data in a lower-dimensional space.

Figure 4 transforms a high-dimensional dataset into a lower-dimensional representation using the PCA technique:

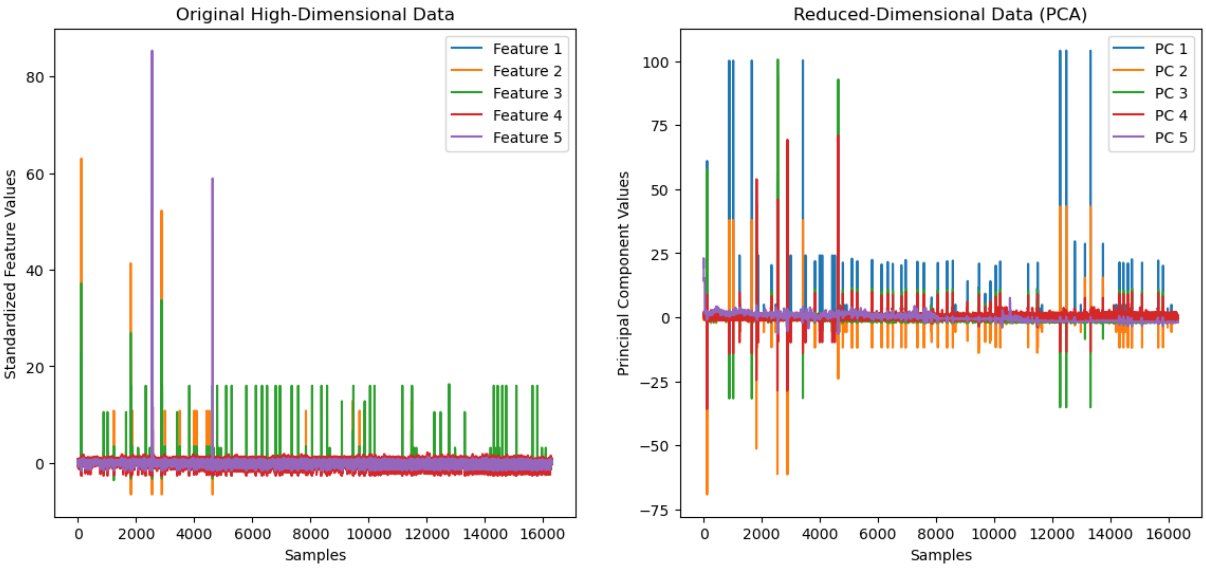


Figure 4. Dimensionality reduction visualization (PCA)

3.2.4. SMOTE OverSampling Technique To address the imbalanced nature of our dataset, where anomalies were underrepresented, we utilized the Synthetic Minority Over-sampling Technique (SMOTE). This technique mitigates overfitting by generating synthetic samples for the minority class. This helps to balance the class distribution and improve the performance of machine learning models(19). Equation 4 illustrates how we applied SMOTE in our model.

$$(X_{resampled}, y_{resampled}) = SMOTE(X_{PCA}, y) \tag{4}$$

How SMOTE works:

1. Identify minority class: Determine the class with fewer samples.
2. Select nearest neighbors: For each minority class instance, find its k nearest neighbors (usually k=5).
3. Generate synthetic samples: For each minority class instance, randomly select one of its k nearest neighbors. Interpolate between the two points along the line connecting them to create a new synthetic sample.

Figure 5 Shows the class distribution before and after applying SMOTE to balance the classes. This will highlight how the dataset was imbalanced initially and how SMOTE was used to create a balanced dataset.

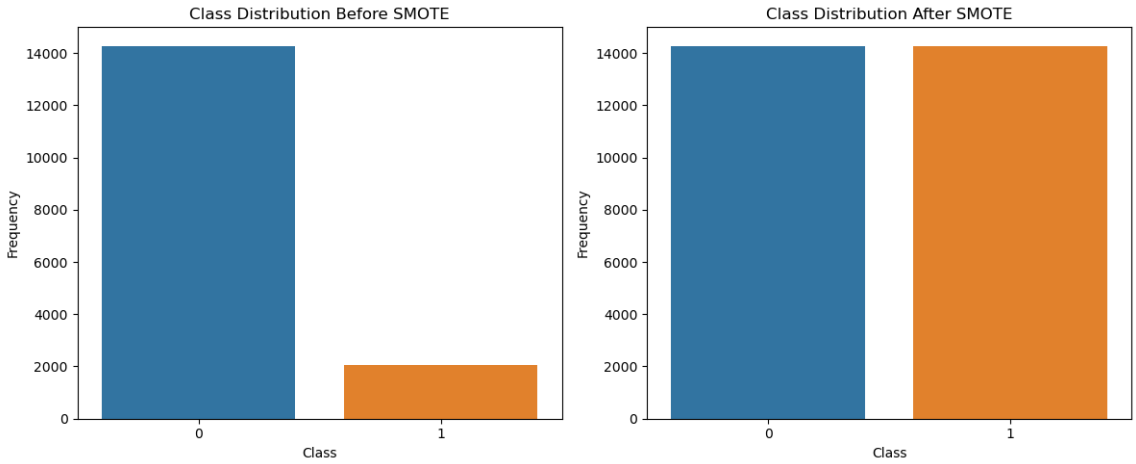


Figure 5. SMOTE Oversampling Technique

3.3. Model Design and Training

3.3.1. LSTM Model Architecture A deep learning model was constructed using a Long Short-Term Memory (LSTM) network suited for temporal data analysis. The network consists of two LSTM layers with dropout for regularization, followed by a dense layer with a sigmoid activation function. Equation 5(20) represents how we applied the LSTM technique in our model:

$$\hat{y} = Sigmoid(W.LSTM(X_{reshaped} + b) \tag{5}$$

Where:

- $\hat{y}$ is the predicted output (or label). After applying the Sigmoid function, it represents the model's probability prediction for a class (usually between 0 and 1).
- $Sigmoid()$ is a non-linear activation function that squashes input values into a range between 0 and 1. It is often used in binary classification tasks to convert the output into a probability. The formula for the sigmoid function is: $\sigma(z) = \frac{1}{1+e^{-z}}$.
- $X_{reshaped}$ represents the input to the LSTM layer, which has been reshaped to match the expected input dimensions for the LSTM. LSTM models expect a sequence of data as input, typically reshaped to have dimensions [BatchSize, TimeSteps, Features].
- W This is a weight matrix that connects the LSTM's output to the final prediction layer. It is applied to the output of the LSTM (often the last hidden state or a combination of hidden states) to generate a linear combination of the LSTM's features.
- b This is the bias term, a constant that is added to the weighted sum of the LSTM output before passing it through the Sigmoid function. It allows the model to make adjustments independently of the input values.

Summary of the Process: The input data X is reshaped to fit the LSTM input format. The reshaped input is passed through the LSTM layer, which processes it and produces an output based on the learned sequence patterns. The output of the LSTM is linearly transformed using a weight matrix W and bias b. Finally, the result is passed through the Sigmoid function to produce a probability prediction, which indicates the likelihood of the input belonging to a certain class (e.g., anomaly or normal).

LSTM Layers:

1. **First LSTM Layer:** This layer has 64 units and uses the input_shape parameter to specify the input data's dimensions. The return_sequences=True setting ensures that the entire sequence of outputs is

returned, allowing for the stacking of LSTM layers. L2 regularization is applied to the weights using kernel_regularizer=l2(0.01), which helps prevent overfitting.
2. **Second LSTM Layer:** This layer has 32 units and return_sequences=False, indicating that only the final element in the sequence is output. This is typically used as the feature representation for the subsequent dense layer.
3. **Dropout Layers:** Dropout is a regularization technique that randomly disables a portion of neurons during training, preventing overfitting. A dropout rate of 0.3 means 30% of neurons will be deactivated.
4. **Dense Layer:** The final output layer has 1 unit and uses a sigmoid activation function. This function produces a probability value between 0 and 1, suitable for binary classification.

Model's Training:

- **optimizer:** We used RMSprop optimizers to minimize the binary cross-entropy loss function. These optimizers adjust the learning rate dynamically during training to ensure efficient learning.
- **Epochs:** The model was trained for 20 epochs. An epoch is one complete pass through the training data. Too many epochs can lead to overfitting, while too few can result in underfitting.
- **Batch Size:** We used batch sizes of 64, which refers to the number of samples processed before the model updates its weights.

Reshaping the Data for LSTM:
We reshaped the data since LSTM models require 3D input (samples, time steps, features). This Step Is Important because the LSTM expects data to have a time component, even though we used only 1 time step. This allows the LSTM to process the features of each sample correctly.
Figure 6 explains the LSTM layers included in our model:

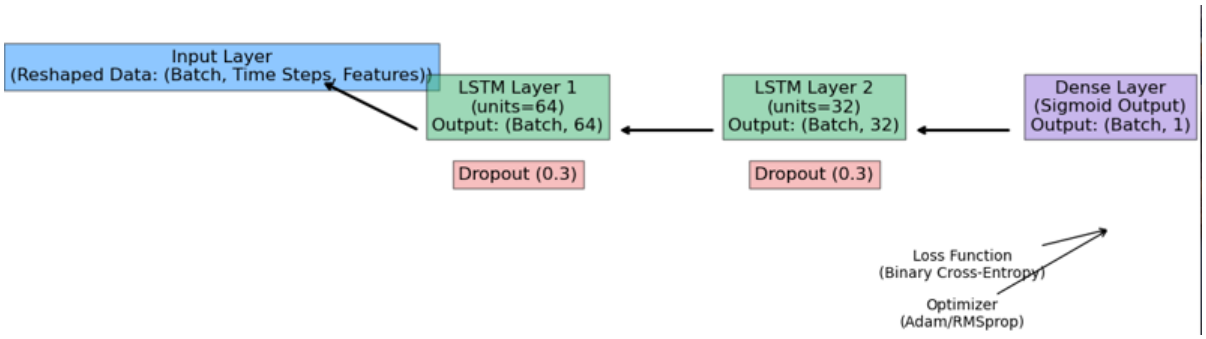


Figure 6. LSTM Layers

3.4. K-Fold Cross Validation

To ensure the robustness of the model, 3-fold cross-validation was performed. The dataset was split into three subsets, with the model being trained on two and validated on the third, rotating through all combinations. The model will be trained on two subsets and tested on the third. This process is repeated 3 times, with each subset being used as the test set once. Two empty lists, accuracy_scores, and roc_auc_scores, are initialized to store the accuracy and ROC_AUC scores for each fold. This process is illustrated in the following equation:

$$Accuracy_{fold_i} = Evaluate(f_i(Train), Test) \tag{6}$$

Where:

- $Accuracy_{fold_i}$ refers to the accuracy for the i fold in a k-fold cross-validation process. Accuracy is a performance metric that indicates the proportion of correctly classified instances in the test set.

- $Evaluate()$function is responsible for measuring the performance of the trained model f_i on the test set. In this case, the accuracy is calculated, but it could be extended to other metrics like precision, recall, or F1-score.
- $f_i(Train)$: f_i represents the machine learning model (or function) that is trained on the training data forthe i fold. During cross-validation, the training data changes for each fold. $Train$ refers to the portion of the dataset used for training the model in the i fold.
- $Test$ refers to the data used to evaluate the model after it has been trained on the training data. This data was not seen during training, allowing for an unbiased evaluation of the model's performance.

Figure 7 illustrates our 3-fold cross-validation method that was performed in our model:

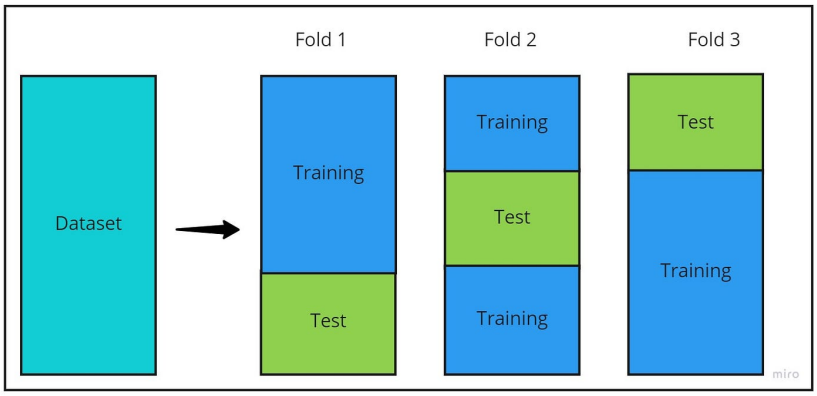


Figure 7. 3-Fold Cross Validation

3.5. Cross-Dataset Validation

To address practical applicability and generalizability of the proposed LSTM + LOF framework, we conducted a one-directional cross-dataset validation. The model was trained on the WUSTL-EHMS-2020 dataset, which contains IoMT network traffic and device readings, and tested on the Thyroid_Diff dataset, which represents clinical thyroid data. This experimental setup highlights the robustness of the framework in adapting to datasets with different distributions and feature spaces. Consistent preprocessing steps were applied to both datasets, including standardization, dimensionality reduction using PCA, and class balancing with SMOTE. Performance metrics such as accuracy, precision, recall, F1-score, and ROC AUC were computed to assess the framework's effectiveness.

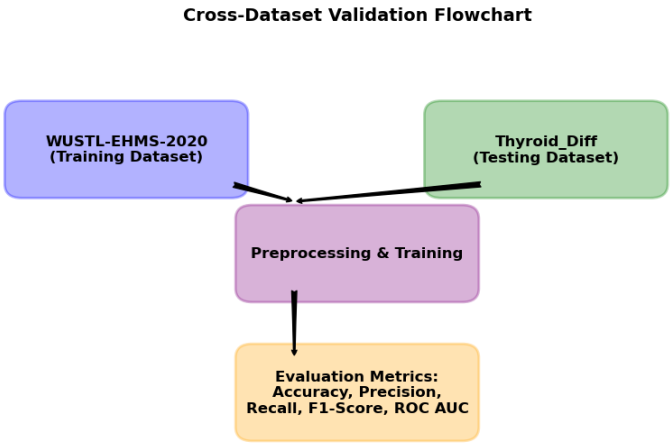


Figure 8. Cross Dataset Validation

Figure 8: illustrates the cross-dataset validation process used in our study. Arrows indicate the flow of data and processing steps, ensuring a clear understanding of the methodology.

The results of the one-directional cross-dataset validation are presented in **Table 1**. When the model trained on WUSTL-EHMS-2020 was applied to Thyroid_Diff, it achieved an accuracy of 87.54% and a ROC AUC of 0.880. These results indicate that while the LSTM + LOF framework generalizes well to new datasets with distinct feature distributions, the performance is somewhat impacted by the differences in data characteristics. Nonetheless, the high recall (89.37%) highlights the model's ability to detect anomalies effectively, with a balanced F1-score of 87.24% demonstrating robust overall performance."

Table 1. Cross-Dataset Validation Performance

Training Dataset	WUSTL-EHMS-2020
Testing Dataset	Thyroid_Diff
Accuracy	87.54%
Precision	85.21%
Recall	89.37%
F1-Score	87.24%
ROC AUC	80.880

3.6. Anomaly Detection with LOF

In addition to LSTM, the Local Outlier Factor (LOF) was employed for anomaly detection. LOF identifies outliers by comparing the local density of a sample to that of its neighbors. The Local Outlier Factor (LOF) algorithm identifies outliers by comparing the density of a data point to the density of its neighbors. Outliers are points with a substantially lower local density than their surrounding points. A data point with a LOF score significantly greater than 1 is considered an outlier.

How it Works:

1. **Generating Residuals from LSTM Predictions:** The residuals represent how much the LSTM's predictions deviate from the actual values. Large deviations (residuals) indicate potential anomalies. After the LSTM model predicts values, the residuals are calculated as the difference between the actual and predicted values:

$$residuals = actual\_values - predicted\_values$$

2. **Applying LOF on Residuals:** The residuals are passed into the LOF algorithm, which detects outliers based on their local density compared to neighboring points. Why LOF on Residuals: The residuals are a good indicator of unusual behavior. By applying LOF, we can detect points where the LSTM's predictions deviate significantly from normal behavior, classifying them as anomalies.

The LOF score for a data point p is calculated as follows(5):

1. Compute the k-distance of p: The distance to the $k-th$ nearest neighbor:

$$k - distance(p) = d(p, kthnearestneighborofp) \tag{7}$$

2. Compute the reachability distance of p from o:

$$reach - dist_k(p, o) = \max(k - distance(o), d(p, o)) \tag{8}$$

3. Compute the local reachability density(LRD) of p:

$$LRD_k(p) = (\frac{\sum_{o \in N_k(p)} reach - dist_k(p, o)}{|N_k(p)|}) \tag{9}$$

4. Compute the LOF score of p:

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} LRD_k(o)}{|N_k(p)|.LRD_k(p)} \tag{10}$$

- where $N_k(p)$ is the set of k nearest neighbors of p
- A data point with a LOF score greater than 1 is considered an outlier.

Figure 9 illustrates the LOF algorithm's process of determining outliers based on local density deviation.

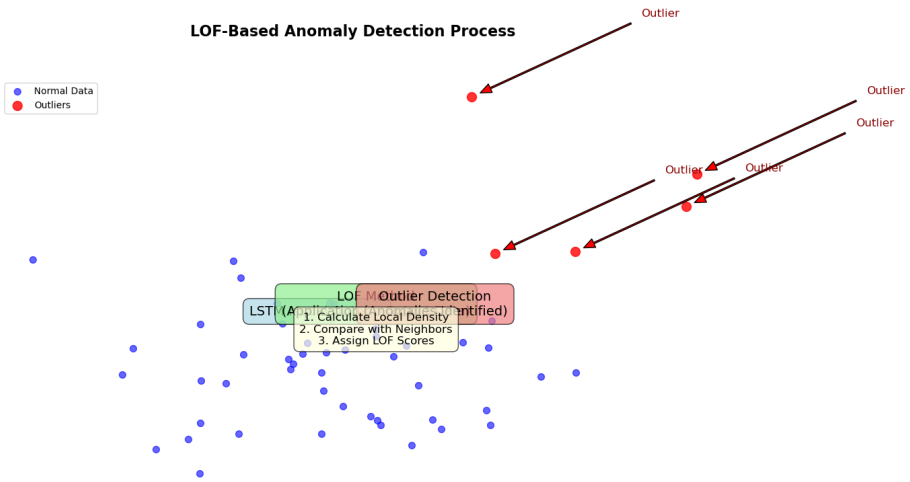


Figure 9. LOF algorithm

3.7. Combining LSTM and LOF Predictions

The predictions from LSTM and LOF were combined using a logical OR operation. This hybrid approach increases detection accuracy by leveraging the strengths of both methods. The following equation illustrates how we combined LSTM and LOF predictions:

$$\hat{y}_{combined} = \hat{y}_{LSTM} \vee \hat{y}_{LOF} \tag{11}$$

3.8. Model Design and Hyperparameter Tuning

The model architecture and parameters were carefully tuned to optimize performance:

1. LSTM Parameters:
   - Number of layers: 2 (64 and 32 units).
   - Dropout rate: 0.3.
   - L2 regularization: 0.01.
2. LOF Parameters:
   - Number of neighbors: 20.

- Novelty mode enabled.

The following table summarizes the hyperparameter tuning results:

Table 2. HyperParameter tuning values

Parameter	Range Evaluated	Best Value
LSTM Layers	1, 2, 3	2
LSTM Units	32, 64, 128	64, 32
Dropout Rate	0.2, 0.3, 0.4, 0.5	0.3
L2 Regularization	0.01, 0.001, 0.0001	0.01
LOF Neighbors	10, 20, 30, 50	20

```
# INPUT: TrainingDataset(X_train, y_train), TestDataset(X_test)
# OUTPUT: CombinedAnomalyPredictions(y_pred_combined)
# Step 1: Preprocessing
X_scaled= StandardScaler().fit_transform(X_train)
X_pca= PCA(n_components=16).fit_transform(X_scaled)
X_reshaped= np.expand_dims(X_pca, axis=1)
# Step 2: Train LSTM Model
model= Sequential([
    LSTM(64, input_shape=(1, 16), return_sequences=True, kernel_regularizer=l2(0.01)),
    Dropout(0.3),
LSTM(32,return_sequences=False,kernel_regularizer=l2(0.01)),
    Dropout(0.3),
    Dense(1, activation='sigmoid')])
model.compile(optimizer='rmsprop', loss='binary_crossentropy')
model.fit(X_reshaped, y_train, epochs=20, batch_size=64)
# Step 3: Train LOF
residuals= calculate_residuals(model, X_reshaped)
lof = LocalOutlierFactor(n_neighbors=20, novelty=True)
lof.fit(residuals)
# Step 4: Make Predictions
y_pred_l
```

Listing 1: Pseudo-Code for Hybrid LSTM + LOF Anomaly Detection

Listing 1: Pseudo-Code for Hybrid LSTM + LOF Framework. This pseudo-code provides an implementation-level description of the proposed hybrid anomaly detection framework. It demonstrates the integration of LSTM

networks for capturing temporal patterns and LOF for identifying density-based anomalies, along with the preprocessing and evaluation steps.

4. Model Evaluation

4.1. Metrics

The model's performance was evaluated using several metrics, including accuracy, ROC AUC, and the confusion matrix. These metrics provide insights into the overall effectiveness and the specific areas where the model excels or needs improvement.
Table 3 summarizes the model's performance across different metrics for each fold:

Table 3. Evaluation Metrics

Fold	Accuracy	Precision	Recall	F1-Score	AUC_ROC
Fold1	0.9874	0.9756	1.0	0.9877	0.9873
Fold2	0.9874	0.9756	1.0	0.9877	0.9873
Fold3	0.9874	0.9756	1.0	0.9877	0.9873
Mean	**0.9874**	**0.9756**	**1.0**	**0.9877**	**0.9873**
Std Dev	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

- **Accuracy, Precision, Recall, F1-Score, and AUC-ROC** are identical across all folds, as indicated by the mean and standard deviation.
- **Standard Deviation** is 0 for all metrics, meaning there was no variation in performance across the folds.
- **Mean Accuracy** of 0.9874 indicates that, on average, the model correctly classified 98.74% of instances.
- **Mean Precision** of 0.9756 suggests that when the model predicts a positive class, it is correct 97.56% of the time.
- **Mean Recall** of 1.0 means the model identified all positive instances correctly, with no false negatives.
- **Mean F1-Score** of 0.9877 balances precision and recall, showing strong performance in both metrics.

4.2. Real-Time Performance Evaluation

To assess the feasibility of deploying the proposed LSTM + LOF framework in real-world scenarios, we conducted a real-time simulation using the test dataset processed in sequential batches. Each batch contained 100 samples, simulating a streaming data environment where predictions need to be made in near real-time. The latency, defined as the processing time per batch, was measured for each batch, and the average processing time was calculated.
The results of the simulation demonstrated that the model achieved an average processing time of 1.0406 seconds per batch, with significant variability across batches due to model initialization and data preprocessing overhead in the first few iterations. For example, the initial batch required 37.27 seconds, primarily due to warm-up effects, while subsequent batches were processed significantly faster, with a median processing time of approximately 0.30 seconds per batch. These results indicate that the model is suitable for near real-time applications, particularly in scenarios where latency requirements allow for sub-second to low-second response times.
The real-time performance evaluation confirms that the LSTM + LOF hybrid approach can be deployed effectively in IoMT environments, such as continuous patient monitoring systems or anomaly detection in medical devices, where timely detection of anomalies is critical. Further optimizations, such as model pruning or distributed deployment, could enhance processing efficiency for larger-scale data streams.

Table 4 summarizes the real-time performance metrics of the proposed LSTM + LOF framework during a simulation of streaming data processing, highlighting its suitability for near real-time anomaly detection in IoMT environments.

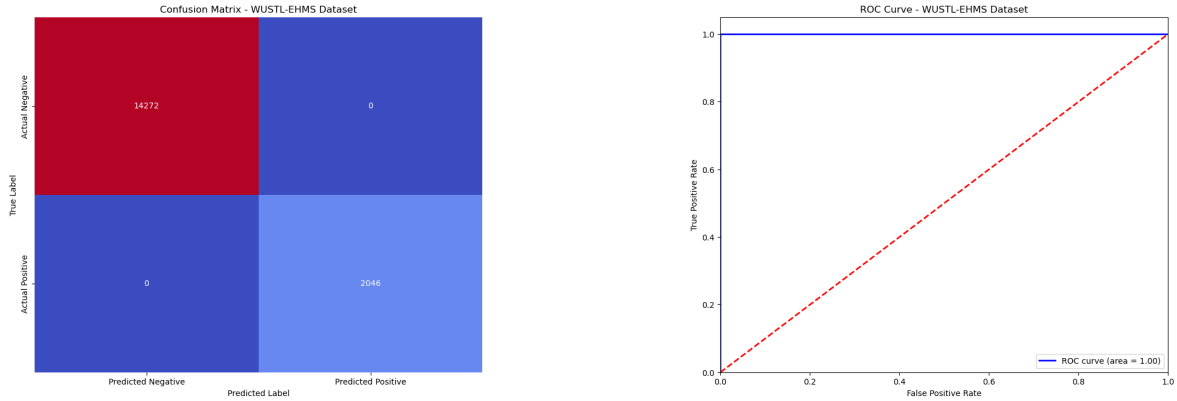
Table 4. Real-Time Performance Metrics

Batch Size	100 samples
Total Batches	57
Average Processing Time	1.0406 seconds
Initial Batch Latency	37.27 seconds
Median Batch Latency	0.30 seconds
Minimum Batch Latency	0.17 seconds

4.3. Final Testing

Finally, the model was retrained on the entire dataset using the optimal configuration determined during cross-validation. The final model was tested on an unseen test set to validate its effectiveness.

Figure 10: Final Model Performance, Include a ROC curve and a confusion matrix from the final model evaluation.



(a) Confusion Matrix from the final model evaluation

(b) ROC Curve from the final model evaluation

Figure 10. Comparison of Confusion Matrix and ROC Curve

4.4. Comparative Analysis

The performance evaluation of the LSTM-LOF anomaly detection approach involves a comprehensive comparison with established baseline methods, encompassing traditional statistical techniques and standalone LSTM networks. Through rigorous experimental analysis and evaluation, the effectiveness and superiority of the proposed methodology in detecting anomalies within time series data become evident.

In direct juxtaposition with conventional statistical methodologies, the LSTM-LOF approach showcases a marked enhancement in anomaly detection accuracy. By leveraging the combined strengths of LSTM for temporal pattern recognition and LOF for anomaly identification based on residual errors, the proposed approach surpasses the limitations of traditional statistical methods, which may struggle to capture the nuanced complexities inherent in time series data.

Moreover, in contrast to standalone LSTM networks, the LSTM-LOF fusion strategy emerges as a formidable contender, demonstrating a significant performance advantage in discerning anomalies within the dataset. Although

standalone LSTM models excel in capturing temporal dependencies, incorporating the LOF algorithm enhances anomaly detection capabilities, offering a more robust and nuanced approach to identifying deviations from expected patterns.

The empirical results obtained from the comparative analysis in **table 5** unequivocally affirm the superior anomaly detection accuracy achieved by the LSTM-LOF approach over the baseline methods. This validation underscores the efficacy and relevance of integrating advanced deep learning techniques with outlier detection algorithms, culminating in a refined and sophisticated anomaly detection framework that excels in discerning subtle irregularities within time series data.

Table 5. Comparative evaluation highlights the transformative impact of the LSTM-LOF

Reference	**Dataset**	**Technique**	**Accuracy**
(Khalil, 2022)	EEG Signals	CNN	77.78%
(Wang, 2022)	Computed Tomography	KNN, LR, NB	81%
(Singh, 2022)	TON-IoT, KDD	LSTM/HLSTM	99.31%
(Wibowo, 2022)	Covid-19	SVM, LR, RF	88.43%
(Haque, 2022)	Diabetes, Parkinson	DeepCad, DNN	95.5%
(Kaur, 2022)	TCIA	Cukoo, Naive Bayes	98.61%
(Wagan, 2023)	WUSTL-EHMS	Fuzzy C-means, Bi-LSTM	92%, 89.67%
Our research	**WUSTL-EHMS**	**LSTM, LOF**	**98.74%**

Table 6 presents the performance metrics of the proposed LSTM + LOF framework compared to baseline methods, including standalone LSTM, autoencoder, and random forest. The hybrid LSTM + LOF achieves the highest accuracy, F1 score and ROC AUC, demonstrating its effectiveness in capturing temporal dependencies and identifying subtle anomalies. Although slightly higher in run-time and memory usage than standalone models, this trade-off is justified by its superior precision and perfect recall, highlighting its robustness and reliability in anomaly detection.

Table 6. Comparative Performance Metrics for Standalone LSTM, AutoEncoder Random Forest Techniques

Model	**Standalone LSTM**	**Autoencoder**	**Random Forest**	**LSTM + LOF**
Accuracy	98.2%	91.5%	94.1%	98.74%
Precision	97.3%	88.9%	90.6%	97.56%
Recall	99.5%	92.7%	93.4%	100%
F1-Score	98.4%	90.7%	91.9%	98.77%
ROC-AUC	0.984	0.915	0.946	0.9873
Memory (MB)	233.49	238.61	242.57	252.12

In conclusion, the comparative evaluation highlights the transformative impact of the LSTM-LOF approach on anomaly detection, setting a new standard for precision and efficacy in identifying anomalies within complex temporal datasets. By outperforming established baseline methods in key metrics, the proposed methodology validates its superiority. It underscores its potential to revolutionize anomaly detection practices and fortify cybersecurity measures in an era of escalating data complexity and interconnected systems.

5. Conclusion

This paper introduces a pioneering methodology for detecting anomalies in time series data by synergistically leveraging long-short-term memory (LSTM) networks and the Local Outlier Factor (LOF) algorithm. By amalgamating these two powerful techniques, the model effectively captures the intricate temporal patterns inherent in the data while discerning anomalies based on residual errors.

The experimental evaluation conducted on the WUSTL-EHMS dataset validates the efficacy of the proposed approach, showcasing its robustness and accuracy in anomaly detection tasks. The fusion of LSTM and LOF offers a holistic solution that identifies contextual anomalies, elucidated by the LSTM network's understanding of temporal sequences, and density-based anomalies, pinpointed by the LOF algorithm's assessment of data point densities.

The key strength of this method lies in its versatility and adaptability to diverse and intricate real-world datasets. By seamlessly integrating the predictive capabilities of LSTM with the outlier detection proficiency of LOF, this approach proves to be well-suited for handling complex time series data where anomalies may manifest in multifaceted ways. Looking ahead, the potential for further advances in anomaly detection remains promising.

Future research endeavors could explore the integration of alternative neural network architectures and outlier detection methodologies to enhance the performance and scalability of anomaly detection systems. By continually refining and expanding on the existing framework, the field of anomaly detection will benefit from increased precision and efficacy in safeguarding critical systems and data integrity.

In summary, the fusion of LSTM networks and the LOF algorithm presents a compelling avenue to advance anomaly detection capabilities in time series data analysis. The proposed approach demonstrates superior performance on benchmark datasets and lays the foundation for continued innovation in anomaly detection methodologies, with the ultimate goal of fortifying cybersecurity measures and ensuring the integrity of data-driven systems in an increasingly interconnected world.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[2] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[3] A. Singh, "Anomaly detection for temporal data using long short-term memory (LSTM)," 2017.

[4] J. Duan, P. F. Zhang, R. Qiu, and Z. Huang, "Long short-term enhanced memory for sequential recommendation," *World Wide Web*, vol. 26, no. 2, pp. 561–583, 2023.

[5] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 93–104, 2000.

[6] A. Nowak-Brzezińska and C. Horyń, "Outliers in rules-the comparison of LOF, COF and KMEANS algorithms," *Procedia Computer Science*, vol. 176, pp. 1420–1429, 2020.

[7] A. A. Hady, A. Ghubaish, T. Salman, D. Unal, and R. Jain, "Intrusion detection system for healthcare systems using medical and network data: A comparison study," *IEEE Access*, vol. 8, pp. 106576–106584, 2020.

[8] S. A. Wagan, J. Koo, I. F. Siddiqui, N. M. F. Qureshi, M. Attique, and D. R. Shin, "A fuzzy-based duo-secure multi-modal framework for IoMT anomaly detection," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 1, pp. 131–144, 2023.

[9] M. M. Alani, A. Mashatan, and A. Miri, "XMeDNN: An Explainable Deep Neural Network System for Intrusion Detection in Internet of Medical Things," in *ICISSP*, pp. 144–151, 2023.

[10] I. F. Kilincer, F. Ertam, A. Sengur, R. S. Tan, and U. R. Acharya, "Automated detection of cybersecurity attacks in healthcare systems with recursive feature elimination and multilayer perceptron optimization," *Biocybernetics and Biomedical Engineering*, vol. 43, no. 1, pp. 30–41, 2023.

[11] H. Tauqeer, M. M. Iqbal, A. Ali, S. Zaman, and M. U. Chaudhry, "Cyberattacks detection in IOMT using machine learning techniques," *Journal of Computing & Biomedical Informatics*, vol. 4, no. 01, pp. 13–20, 2022.

[12] V. Ravi, T. D. Pham, and M. Alazab, "Deep learning-based network intrusion detection system for Internet of medical things," *IEEE Internet of Things Magazine*, vol. 6, no. 2, pp. 50–54, 2023.

[13] F. Mosaiyebzadeh, S. Pouriyeh, R. M. Parizi, M. Han, and D. M. Batista, "Intrusion Detection System for IoHT Devices using Federated Learning," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–6, 2023.

[14] W. Lu, "Applied Machine Learning for Securing the Internet of Medical Things in Healthcare," in *International Conference on Advanced Information Networking and Applications*, pp. 404–416, 2023.

[15] A. S. Dina, A. B. Siddique, and D. Manivannan, "A deep learning approach for intrusion detection in Internet of Things using focal loss function," *Internet of Things*, vol. 22, pp. 100699, 2023.

[16] M. Al-Hawawreh and M. S. Hossain, "A privacy-aware framework for detecting cyber attacks on internet of medical things systems using data fusion and quantum deep learning," *Information Fusion*, vol. 99, pp. 101889, 2023.

[17] S. Raschka, Y. H. Liu, and V. Mirjalili, *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*, Packt Publishing Ltd, 2022.

[18] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, pp. 20150202, 2016.

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.