Network Intrusion Detection via Anomaly Score Prediction Using Autoencoder and LSTM

A Comprehensive Framework for Network Intrusion Detection Using Autoencoder and LSTM-Based Anomaly Score Prediction

Aayush Bagga (2022UCA1806)*, Ayush Arora (2022UCA1822)†, Saaz Gupta (2022UCA1860)‡
Department of Computer Science, Netaji Subhas University of Technology (NSUT), Delhi, India
Emails: *aayush.bagga.ug22@nsut.ac.in, †ayush.arora.ug22@nsut.ac.in, ‡saaz.gupta.ug22@nsut.ac.in

***Abstract*—The rapid expansion of networked systems and cloud infrastructures has elevated the importance of robust cybersecurity solutions. Traditional intrusion detection systems often struggle to detect sophisticated or previously unseen attacks due to their reliance on signature-based approaches. In this paper, we propose a comprehensive framework for network intrusion detection based on anomaly score prediction using Autoencoders and Long Short-Term Memory (LSTM) networks. The Autoencoder captures non-linear feature representations and reconstructs normal traffic patterns, while the LSTM leverages temporal dependencies to enhance anomaly detection performance. This hybrid approach enables the system to effectively identify deviations from normal behavior, ensuring real-time threat detection with high accuracy. Experimental results on benchmark datasets demonstrate that our method outperforms conventional techniques in both detection rate and false positive minimization.**

***Index Terms*—Network Intrusion Detection, Anomaly Detection, Autoencoder, LSTM, Anomaly Score, Cybersecurity, Deep Learning, Temporal Analysis.**

I. Introduction and Motivation

As cyber threats grow in complexity and volume, traditional intrusion detection systems struggle to keep pace—particularly when facing novel or subtle attacks. Static, rule-based methods often fail to identify previously unseen threats, highlighting the need for more adaptive and intelligent approaches. In this project, we utilize the KDD Cup 1999 dataset to build an anomaly-based intrusion detection system using deep learning. Our model combines an Autoencoder for learning compressed representations of network traffic with an LSTM to capture temporal patterns. Network data is processed in batches of 10, and an anomaly score is computed for each batch. Batches exceeding a defined threshold are flagged as potential intrusions. This approach aims to detect both known and unknown attacks effectively while minimizing false alarms and enhancing real-time security monitoring.

II. Related Work

A. Traditional Intrusion Detection Approaches

Early intrusion detection systems (IDS) relied primarily on signature-based and statistical rule-based methods. Signature-based systems such as SNORT and Bro/Zeek match incoming traffic against a database of known attack patterns, yielding high precision for established threats but failing to detect novel or polymorphic attacks. Statistical anomaly detection approaches model "normal" traffic using features like byte-count histograms or PCA-derived subspaces , flagging deviations above fixed thresholds. While these methods can identify unknown attacks, they typically suffer from high false-positive rates and lack the capacity to capture complex temporal dependencies in network flows.

B. Deep-Learning for Anomaly Detection

The advent of deep learning has spurred a shift toward learned representations and end-to-end anomaly scoring.

- **Autoencoders.** Deep autoencoders compress high-dimensional traffic features into a low-dimensional latent space and reconstruct inputs; reconstruction error serves as an anomaly score . Variational autoencoders (VAEs) further model the distribution of normal traffic to provide probabilistic anomaly estimates .
- **Recurrent Models.** LSTM and GRU networks capture sequential dependencies in time-series data, improving detection of stealthy or low-and-slow attacks . These models learn to predict the next record in a sequence, with prediction error indicating anomalous behavior.

C. Batch-Based

Real-time network monitoring demands efficient batch or streaming inference strategies:

- **Sliding Window / Batch Inference.** Traffic is processed in fixed-size windows (e.g., 5–20 records) with overlap to balance detection latency and contextual information . Adaptive window schemes adjust batch size based on traffic variability to optimize throughput and accuracy.
- **Thresholding Strategies.** Fixed thresholds on anomaly scores are simple but brittle; percentile-based or ROC/AUC-driven thresholds yield more robust decision boundaries . Dynamic thresholding adapts to traffic seasonality and load fluctuations in real time.

Our work builds on these foundations by processing KDD Cup 1999 network data in batches of 10 records, using a deep autoencoder for feature extraction followed by an LSTM to assign an anomaly score per batch. Batches exceeding a tunable threshold are flagged as intrusions, combining spatial compression, temporal context, and an efficient batch inference

strategy to detect both known and novel attacks with low latency.“‘

III. Proposed Framework

A. TinyML and Apache Spark

We leverage a hybrid edge–cloud setup:

- **Apache Spark (Cloud):** Preprocessing of the full KDD Cup 1999 dataset—cleaning, label encoding, and normalization—is performed at scale using Spark's DataFrame API and MLlib. This ensures efficient handling of millions of records and produces feature vectors ready for dimensionality reduction.
- **TinyML (Edge):** The trained Autoencoder+LSTM model is quantized (e.g., via TensorFlow Lite Micro) and deployed on microcontroller-based network sensors. Edge inference on batches of 10 records yields low-latency anomaly scores without constant cloud communication.

B. Principal Component Analysis (PCA)

To reduce the 41 original KDD features to a compact representation, we apply PCA after Spark preprocessing:

1) Compute the covariance matrix of standardized features.
2) Retain the top 14 principal components
3) Output a 14-dimensional feature vector for each record, reducing noise and computational load.

C. Autoencoder

An undercomplete deep Autoencoder further compresses the PCA output into a latent vector:

- **Encoder:** A sequence of three LSTM layers with 80, 50, and 20 units respectively, where the first two return full sequences. All layers use the SELU activation function.
- **Decoder:** A mirrored sequence with a RepeatVector followed by LSTM layers of 50 and 80 units, ending with a TimeDistributed Dense layer to reconstruct the original input sequence.
- **Training:** The model is trained to minimize the reconstruction mean squared error (MSE) on normal traffic data sequences.

The reconstruction error of each record serves as a preliminary anomaly indicator.

D. Long Short-Term Memory (LSTM)

To capture temporal correlations, we feed sequences of 10 consecutive autoencoder bottleneck vectors into an LSTM:

- **Encoder:** A stacked LSTM architecture processes each sequence of 10 records (window size). It consists of three LSTM layers with 80, 50, and 20 hidden units respectively, using SELU activations.
- **Decoder:** The decoder mirrors the encoder's structure in reverse. It uses a RepeatVector to expand the bottleneck output, followed by LSTM layers with 50 and 80 units, and a TimeDistributed Dense layer with linear activation to reconstruct the original input dimension.
- **Anomaly Detection:** A separate single-layer LSTM with 16 hidden units is used for anomaly scoring. It takes sequences of bottleneck vectors as input, with the final hidden state passed through a dense layer to produce a scalar anomaly score. Sequences with scores exceeding a threshold τ (95th percentile from normal validation data) are flagged as intrusions.

E. Kafka Integration

To enable real-time anomaly detection, we integrate Apache Kafka as a scalable and fault-tolerant streaming platform between the data ingestion layer and the inference pipeline:

- **Producer:** Raw network traffic logs are ingested by edge devices or network sniffers and sent to Kafka topics as structured JSON messages.
- **Consumer:** A real-time consumer application, built in Python with the Kafka-Python library, consumes batches of 10 records, applies preprocessing (normalization and PCA), and passes the processed features to the deployed Autoencoder+LSTM model for anomaly scoring.

By combining PCA, deep autoencoding, and LSTM sequence modeling—deployed via TinyML on edge devices and orchestrated with Spark in the cloud—our framework achieves both high detection accuracy and real-time performance.“‘

IV. Experimental Evaluation

A. Dataset

We evaluate our framework on the following benchmark:

- **KDD Cup 1999 (10% subset):** This is the widely used benchmark dataset for network intrusion detection, originally released for the Third International Knowledge Discovery and Data Mining Tools Competition. The 10% subset contains approximately 494,021 records, representing 10% of the full dataset. Each record comprises 41 features (such as duration, protocol type, service, flag, and byte counts) and a label indicating whether the connection is “normal” or an attack. This mini dataset maintains the statistical characteristics of the full set and is suitable for developing and evaluating anomaly detection models under resource constraints.

B. Dataset Visualisation using R and Power Bi

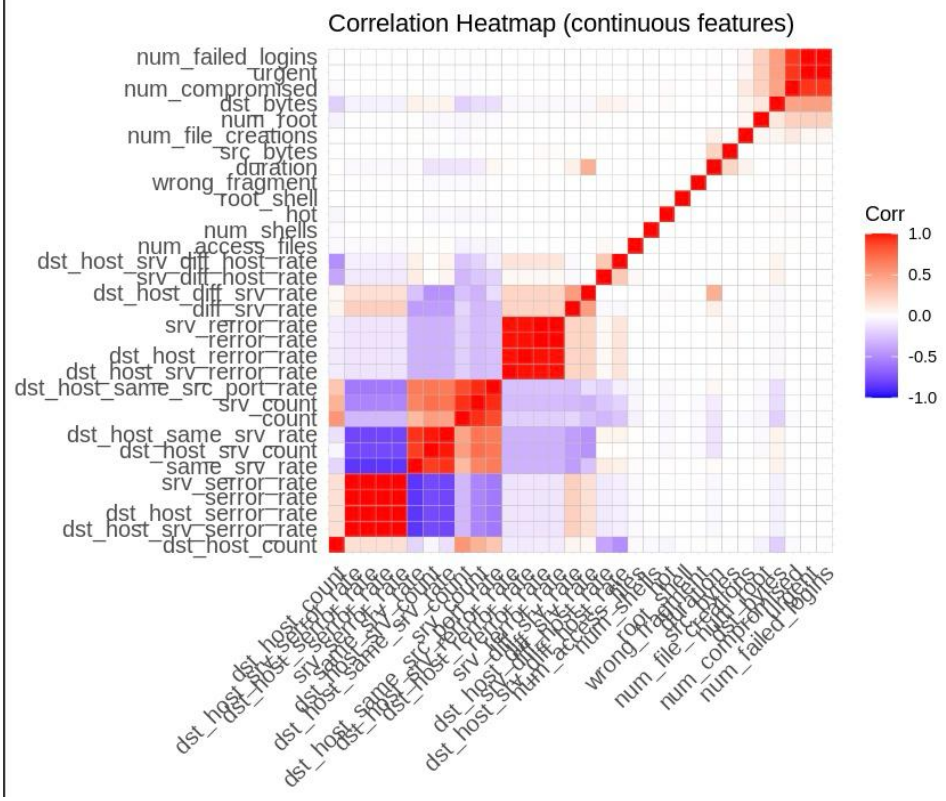

Fig. 1. Correlation among various features of the KDD cup dataset.

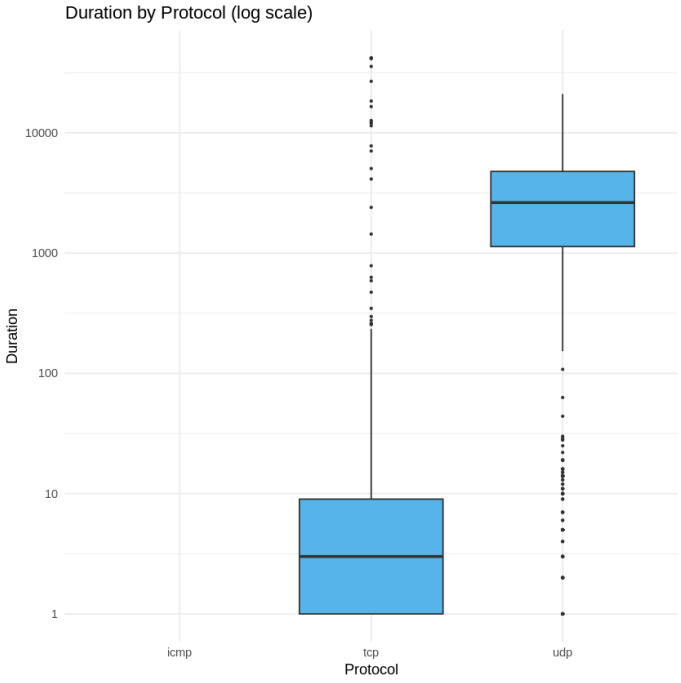

Fig. 2. Differences among the duration of each protocol.

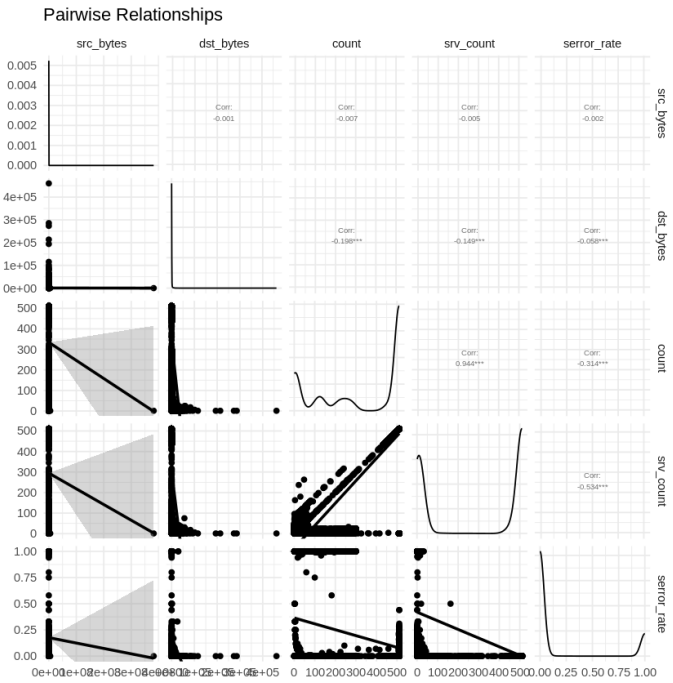

Fig. 3. Pair-wise relation among some important features.

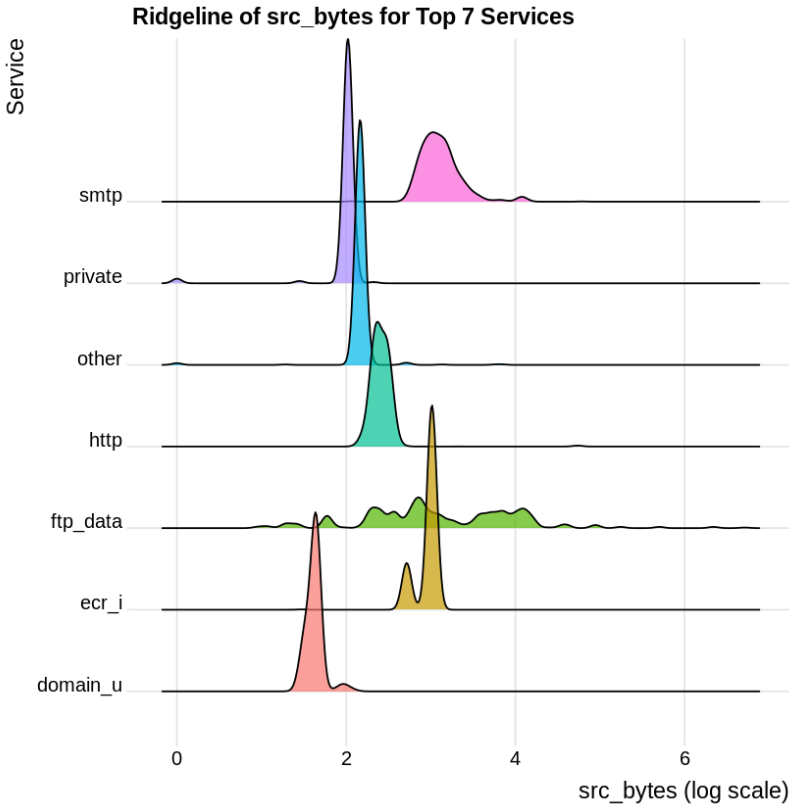

Fig. 4. Top 7 services.

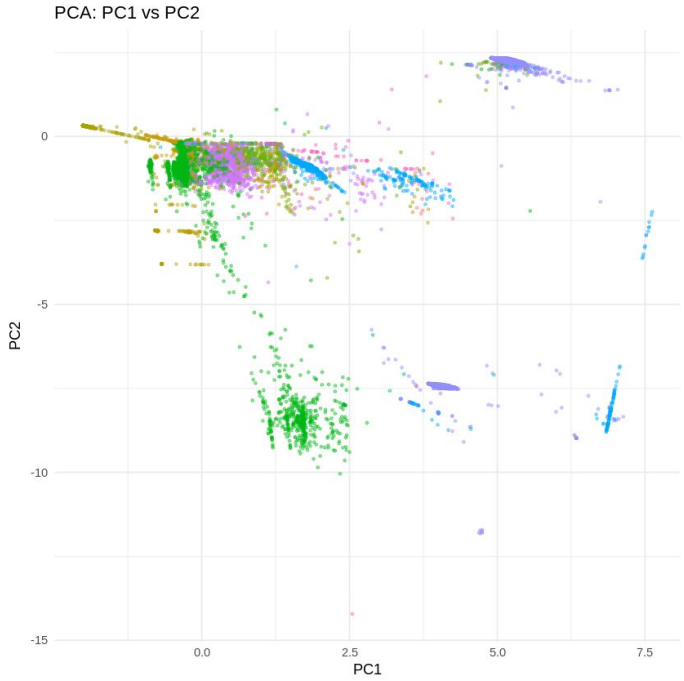

Fig. 5. Scatter plot of the first two PCA features.

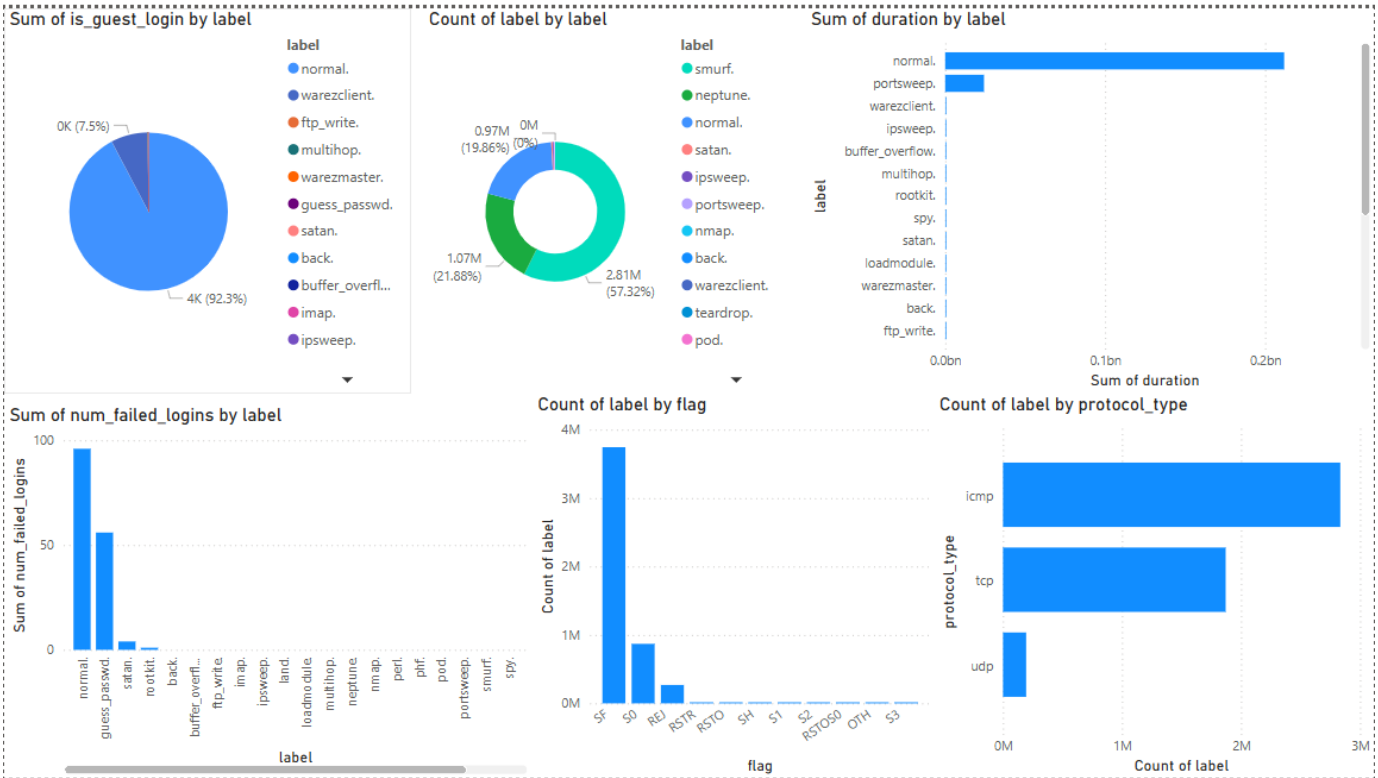

Fig. 6. Various informational graphs generated using power bi.

C. Implementation Details

Our pipeline begins by preprocessing the KDD Cup data in Apache Spark, where all 41 features are standardized and PCA is applied to reduce them to the top 24 principal components, preserving over 90% of the variance. These 24-dimensional vectors are then fed into a deep LSTM-based autoencoder. The encoder consists of three stacked LSTM layers with 80, 50, and 20 units, compressing each input sequence into a compact bottleneck representation. The decoder mirrors this architecture, using LSTM layers and a TimeDistributed dense layer to reconstruct the input sequences. The model is trained on normal traffic using reconstruction mean squared error (MSE) as the loss.

For anomaly detection, sequences of 10 consecutive bottleneck vectors are passed to a separate single-layer LSTM with 16 hidden units. The final hidden state is mapped through a dense layer to produce an anomaly score for each batch. During evaluation, batches with scores above the 95th percentile of the validation-set scores are flagged as potential intrusions. Apache Spark manages data processing at scale, while TensorFlow is used for training and inference.

In addition to Spark and TensorFlow, Kafka is integrated into the pipeline to enable real-time anomaly detection. Raw network traffic data is streamed in real-time via Kafka producers, which send the data to Kafka brokers. The Kafka brokers ensure that the data is reliably delivered for processing. Kafka consumers read the incoming data in batches of 10 records, which are then preprocessed by standardizing and applying PCA. The resulting 24-dimensional feature vectors are passed through the trained autoencoder, and the anomaly detection model evaluates the batches, calculating anomaly scores. If the score exceeds the threshold, the batch is flagged as an intrusion. Kafka's real-time streaming and high-throughput capabilities allow for low-latency processing and efficient handling of large-scale data, making it a crucial component of the overall framework. .

D. Results and Discussion

Our intrusion detection framework demonstrates strong anomaly detection capabilities. Using a threshold of 0.204, the system achieves high performance across all key metrics: a precision of 0.9928, recall of 0.9648, and F1-score of 0.9786. The overall classification accuracy stands at 99.97%. These results indicate that the model reliably distinguishes anomalous behavior from normal traffic with minimal false positives or negatives, making it highly effective for real-world deployment in network security environments.

```
(kafka-env) venv(base) → aihtMajor python consumer.py

ALERT: {'window_index': 0, 'score': 0.6465278749388206, 'anomaly': 1}
ALERT: {'window_index': 1, 'score': 0.7465316356101607, 'anomaly': 1}
ALERT: {'window_index': 2, 'score': 0.7511272568344439, 'anomaly': 1}
ALERT: {'window_index': 3, 'score': 0.338981749611422, 'anomaly': 0}
ALERT: {'window_index': 4, 'score': 0.0, 'anomaly': 0}
ALERT: {'window_index': 5, 'score': 0.22663534064742924, 'anomaly': 0}
ALERT: {'window_index': 6, 'score': 0.746143366896574, 'anomaly': 1}
ALERT: {'window_index': 7, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 8, 'score': 0.22775596604145562, 'anomaly': 0}
ALERT: {'window_index': 9, 'score': 0.40087885235375653, 'anomaly': 0}
ALERT: {'window_index': 0, 'score': 0.0, 'anomaly': 0}
ALERT: {'window_index': 1, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 2, 'score': 0.9216052515278925, 'anomaly': 1}
ALERT: {'window_index': 0, 'score': 0.0, 'anomaly': 0}
ALERT: {'window_index': 1, 'score': 0.24091372469863911, 'anomaly': 0}
ALERT: {'window_index': 2, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 0, 'score': 0.0, 'anomaly': 0}
ALERT: {'window_index': 1, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 0, 'score': 0.0, 'anomaly': 0}
ALERT: {'window_index': 1, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 0, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 1, 'score': 0.0, 'anomaly': 0}
ALERT: {'window_index': 0, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 1, 'score': 0.0, 'anomaly': 0}
ALERT: {'window_index': 0, 'score': 0.0, 'anomaly': 0}
ALERT: {'window_index': 1, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 0, 'score': 0.0, 'anomaly': 0}
ALERT: {'window_index': 1, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 0, 'score': nan, 'anomaly': 0}
ALERT: {'window_index': 0, 'score': 1.0, 'anomaly': 1}
ALERT: {'window_index': 1, 'score': 0.0, 'anomaly': 0}
```

Fig. 7. Real-time intrusion detection output using Kafka-based streaming data.

V. Conclusion and Future Work

This paper presented an LSTM-based anomaly detection framework for network intrusion detection using the KDD Cup dataset. The approach combines PCA for dimensionality reduction, a deep LSTM autoencoder for feature reconstruction, and a sequence-based LSTM model to generate anomaly scores. Our method achieves high accuracy and precision while maintaining low false positives, demonstrating its effectiveness in identifying anomalous patterns in network traffic.

Future work will focus on enhancing model generalization across modern intrusion datasets and exploring transformer-based architectures for improved sequence modeling in highly dynamic network environments.

References

[1] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, Jul. 2009, pp. 1–6.

[2] K. Siddique, Z. Akhtar, F. A. Khan, and Y. Kim, "KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research," *Computer*, vol. 52, no. 1, pp. 41–51, Jan. 2019.

[3] S. Sapre, P. Ahmadi, and K. R. Islam, "A Robust Comparison of the KDDCup99 and NSL-KDD IoT Network Intrusion Detection Datasets Through Various Machine Learning Algorithms," *arXiv preprint arXiv:1912.13204*, Dec. 2019.

[4] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. 2016 International Conference on Platform Technology and Service (PlatCon)*, Jeju, Korea, Feb. 2016, pp. 1–5.

[5] M. Z. Alom, V. Bontupalli, and T. Taha, "Intrusion detection using deep belief networks," in *Proc. 2015 IEEE National Aerospace and Electronics Conference (NAECON)*, Piscataway, NJ, USA, Jun. 2015, pp. 339–344.

[6] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Trans. Comput.*, vol. 65, no. 11, pp. 2986–2998, Nov. 2016.

[7] S. Khan, K. Kifayat, A. Kashif Bashir, A. Gurtov, and M. Hassan, "Intelligent intrusion detection system in smart grid using computational intelligence and machine learning," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, e4062, Jan. 2020.