Adaptive Machine Learning Frameworks for Real-Time Intrusion Detection in IoT Networks

Sowjanya Jindam
Sr. Assistant Professor
Department of Information Technology
MVSR Engineering College
Hyderabad, India
sowjanya_ it@mvsrec.edu.in

Syed Abrar Ahmed
Department of CSE
MVSR Engineering College
Hyderabad, India
abrarahmedsyed47@gmail.com

***Abstract*—The proliferation of Internet of Things (IoT) devices has created a massive attack surface for cyber threats, rendering traditional security mechanisms insufficient. The resource constraints of IoT nodes—limited processing power, memory, and energy—make deploying conventional, heavy-weight security protocols impractical. This paper proposes a novel, hybrid Intrusion Detection System (IDS) specifically designed for these constrained environments. We introduce an adaptive framework that combines Convolutional Neural Networks (CNN) for spatial feature extraction with Long Short-Term Memory (LSTM) networks for temporal sequence analysis. By integrating these two architectures, our model effectively captures both the granular, low-level feature correlations and the high-level, long-term dependencies inherent in network traffic. Our experimental results, conducted on the comprehensive CICIDS2017 dataset, demonstrate that the proposed model achieves a detection accuracy of 98.7%, significantly outperforming traditional Support Vector Machine (SVM) and standalone Deep Neural Network (DNN) approaches. Furthermore, the framework exhibits a low inference latency of 0.12 ms per flow, making it highly suitable for real-time applications at the network edge.**

***Index Terms*—Internet of Things (IoT), Intrusion Detection System (IDS), Deep Learning, Network Security, CNN-LSTM, Edge Computing, Feature Selection.**

I. Introduction

The Internet of Things (IoT) has integrated seamlessly into critical infrastructure, healthcare, smart homes, and smart cities, generating vast amounts of data and connectivity. Estimates suggest that by 2025, there will be over 75 billion connected IoT devices worldwide. However, this ubiquity introduces severe security vulnerabilities. Unlike traditional IT infrastructure, IoT devices often operate with limited processing power, memory, and energy, making them ill-suited for heavy encryption or complex on-device security protocols. Consequently, the network layer becomes the primary line of defense, necessitating robust Intrusion Detection Systems (IDS).

Specific IoT protocols, such as MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol), often lack built-in security features comparable to HTTPS, making them susceptible to Man-in-the-Middle (MitM) attacks, unauthorized subscriptions, and eavesdropping. Furthermore, the heterogeneity of IoT devices creates a fragmented security landscape where patch management is difficult, if not impossible. Attacks such as the Mirai botnet have demonstrated how insecure IoT devices can be weaponized to launch massive Distributed Denial of Service (DDoS) attacks.

Traditional IDS approaches fall into two categories: signature-based and anomaly-based. Signature-based detection fails to identify novel or Zero-Day attacks, as it relies on a database of known threat patterns. Anomaly-based detection offers a solution by modeling "normal" behavior and flagging deviations. However, traditional machine learning models like Random Forest or SVM often struggle with the high dimensionality of network data and fail to capture complex, non-linear relationships, leading to high False Positive Rates (FPR).

Deep Learning (DL) has emerged as a powerful tool for overcoming these limitations. Specifically, Convolutional Neural Networks (CNNs) excel at extracting spatial features from raw data, while Long Short-Term Memory (LSTM) networks are designed to learn from sequences. The dynamic nature of IoT traffic requires a system that can adapt to changing network patterns in real-time while maintaining low latency.

This paper presents three key contributions:

1) A hybrid deep learning architecture (CNN-LSTM) that synergizes the spatial feature extraction capabilities of CNNs with the temporal sequence learning of LSTMs.
2) A robust preprocessing pipeline optimized for IoT traffic, utilizing Recursive Feature Elimination (RFE) for dimensionality reduction and SMOTE for addressing class imbalance.
3) A comprehensive evaluation against state-of-the-art models using the CICIDS2017 benchmark dataset, demonstrating superior accuracy and real-time performance viability.

The remainder of this paper is organized as follows: Section II reviews related work. Section III provides the theoretical background of the algorithms used. Section IV details the proposed methodology. Section V describes the experimental setup and results, and Section VI provides a discussion and conclusion.

II. Related Work

Recent literature has shifted focus towards Machine Learning (ML) and Deep Learning (DL) for network security.

A. Traditional Machine Learning Approaches

Initial research focused on standard ML algorithms. Moustafa et al. [1] proposed an ensemble of diverse learning techniques for NIDS, highlighting the importance of feature correlation. Their work utilized the UNSW-NB15 dataset and demonstrated that correlating features could improve detection rates. However, their approach required significant computational overhead during the feature engineering phase. Similarly, SVMs and Decision Trees have been widely applied but often plateau in performance when faced with the massive, high-dimensional datasets typical of modern IoT networks.

B. Deep Learning Approaches

To address the scale of data, Deep Learning has been adopted. Roopak et al. [2] explored Multi-Objective Deep Belief Networks (DBN) for intrusion detection, achieving high accuracy on the NSL-KDD dataset. Despite the success, the model struggled with the temporal dependencies inherent in volumetric attacks like DDoS, where the sequence of packets is as important as the content of individual packets.

Diro and Chilamkurti [9] proposed a distributed deep learning scheme for IoT, utilizing a parallel master-worker architecture. While effective, the communication overhead between nodes can be prohibitive in bandwidth-constrained environments.

C. Hybrid Architectures

More recently, hybrid models have gained traction. Li et al. [3] demonstrated that spatial features of network flows (converted to image format) could effectively classify malware traffic using CNNs. However, CNNs alone lack the memory capacity to track long-term dependencies. Combining architectures attempts to solve this. Khan et al. [8] utilized a hybrid approach involving particle swarm optimization, but the training time was excessive for dynamic environments. Our work extends the CNN-based approach by integrating LSTM units to specifically address the time-series nature of packet flows, a critical factor in detecting modern persistent threats and ”low-and-slow” attacks.

III. Theoretical Background

This section outlines the theoretical underpinnings of the deep learning components utilized in our framework.

A. Convolutional Neural Networks (CNN)

While traditionally used for image processing, CNNs are highly effective for sequential data when applied as 1D convolutions. In the context of network traffic, a 1D-CNN slides a filter kernel over the input flow vector to extract local features. Given an input vector $\square$ of length $\square$ and a filter $\square$ of length $\square$,the convolution operation for feature map $\square$ is defined as:

$$\square_{\square} = \square \left(\sum_{\square=0}^{\square-1} \square_{\square} \cdot \square_{\square+\square} + \square \right) \quad (1)$$

where $\square$ is the bias term and $\square$ is a non-linear activation function, typically Rectified Linear Unit (ReLU), defined as $\square(\square) = \max(0, \square)$. This operation allows the network to learn local patterns, such as specific byte sequences or flag combinations, that are indicative of malicious activity.

B. Long Short-Term Memory (LSTM)

Standard Recurrent Neural Networks (RNNs) suffer from the vanishing gradient problem, making them unable to learn long-term dependencies. LSTMs address this by introducing a memory cell $\square_{\square}$ and three gates: the input gate ($\square_{\square}$), forget gate ($\square_{\square}$), and output gate ($\square_{\square}$). The forget gate decides what information to discard from the cell state:

$$\square_{\square} = \square(\square_{\square} \cdot [h_{\square-1}, \square_{\square}] + \square_{\square}) \quad (2)$$

The input gate decides which new information to store:

$$\square_{\square} = \square(\square_{\square} \cdot [h_{\square-1}, \square_{\square}] + \square_{\square}) \quad (3)$$

The cell state is updated as:

$$\square_{\square} = \square_{\square} * \square_{\square-1} + \square_{\square} * \tanh(\square_{\square} \cdot [h_{\square-1}, \square_{\square}] + \square_{\square}) \quad (4)$$

Finally, the output gate determines the hidden state:

$$\square_{\square} = \square(\square_{\square} \cdot [h_{\square-1}, \square_{\square}] + \square_{\square}) \quad (5)$$

$$h_{\square} = \square_{\square} * \tanh(\square_{\square}) \quad (6)$$

Here, $\square$ represents the sigmoid function, and $*$ denotes element-wise multiplication. This gating mechanism allows the LSTM to retain critical information about traffic flow context over long sequences, which is essential for detecting multi-stage attacks.

IV. Proposed Methodology

Our proposed framework consists of four stages: Data Preprocessing, Feature Selection, the Hybrid Classification Model, and Hyperparameter Optimization.

A. Data Preprocessing and Balancing

Raw network traffic data contains varying scales and formats. We first apply data cleaning to remove NaN values and infinite numbers caused by flow calculation errors.

1) Normalization: We apply Min-Max normalization to scale continuous features (e.g., Flow Duration, Packet Length) to the range [0, 1]. This ensures that features with larger ranges do not dominate the gradient updates during training.

$$\square_{\square\square\square\square} = \frac{\square - \square_{\square\square\square}}{\square_{\square\square\square} - \square_{\square\square\square}} \quad (7)$$

2) Class Balancing (SMOTE): Network intrusion datasets are inherently imbalanced; normal traffic vastly outnumbers attack traffic. We employ Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples for minority classes (e.g., Heartbleed, Infiltration). This prevents the model from biasing towards the majority class (Benign), which would otherwise yield high accuracy but zero recall for rare attacks.

3) Encoding: Categorical features, such as protocol types (TCP, UDP) and flags, are One-Hot Encoded.

B. Feature Selection

The CICIDS2017 dataset contains over 80 features. Using all of them increases computational complexity and the risk of overfitting. We employed **Information Gain (IG)** to select the most relevant features. IG measures the reduction in entropy $□$ given a feature $□$:

$$□□(□, □) = □(□) − \sum_{□ \in □□□□□□□(□)} \frac{|□_□|}{|□|} □(□_□) \quad (8)$$

Based on the IG scores, we selected the top 40 features, including Flow Duration, Total Fwd Packets, Total Backward Packets, Flow Bytes/s, and Packet Length Std.

C. Hybrid Model Architecture

The core engine utilizes a sequential combination of CNN and LSTM layers.

1) CNN Layers (Spatial Feature Extraction): Two 1D-Convolutional layers are employed. The first layer uses 64 filters with a kernel size of 3, followed by a Max-Pooling layer to downsample the features. The second CNN layer uses 128 filters to extract higher-level abstract features.

2) LSTM Layers (Temporal Feature Extraction): The output of the CNN is fed into an LSTM layer with 64 units. We utilize a 'return_sequences=False' configuration, as we are interested in the final classification based on the entire sequence context.

3) Dropout and Regularization: To prevent overfitting, we insert Dropout layers (rate = 0.5) after the LSTM layer. This randomly deactivates neurons during training, forcing the network to learn robust features that are not reliant on specific node pathways.

4) Classification Layer: The final output is passed through a fully connected dense layer with a Softmax activation function to provide the final classification probability for multi-class detection.

D. Training Algorithm

The training process minimizes the Categorical Cross-Entropy Loss function. We utilize the Adam optimizer due to its adaptive learning rate capabilities.

Algorithm 1: CNN-LSTM Training Procedure

Input: Dataset D (features X, labels Y)
Output: Trained Model M
1: Initialize Model parameters W, b
2: Split D into Train (70%), Val (10%), Test (20%)
3: Apply Min-Max Scaling to X
4: Apply Feature Selection (Top 40 IG)
5: Apply SMOTE to Train_Set
6: Reshape X for 3D input [Samples, TimeSteps, Feats]
7: **For** epoch = 1 to 50 **do**:
8: **For** batch in Train_Set **do**:
9: Extract spatial features (CNN)
10: Extract temporal features (LSTM)
11: □ = Softmax(Dense(LSTM_out))
12: □ = − ∑(□ · log(□))
13: Update W, b using Adam
14: **End For**
15: Calculate Validation Accuracy
16: **If** Val_Loss increases > 3 times:
17: Stop (Early Stopping)
18: **End For**
19: Return M

Fig. 1. Proposed Training Algorithm

V. Experimental Results

A. Dataset Description

We utilized the **CICIDS2017** dataset, which reflects modern attack scenarios. It contains benign traffic and the most up-to-date common attacks, including DoS/DDoS, PortScan, Web Attacks (SQL Injection, XSS), and Botnet traffic. The dataset consists of over 2.8 million records.

B. Experimental Setup

The framework was implemented using Python 3.8, Tensor-Flow 2.4, and Keras on a workstation equipped with an NVIDIA RTX 3080 GPU and 32GB RAM.

C. Hyperparameter Tuning

We performed a Grid Search to optimize the model parameters. The results of the tuning process are summarized in Table I.

TABLE I
HYPERPARAMETER OPTIMIZATION RESULTS

Parameter	Search Space	Optimal Value
Batch Size	{32, 64, 128}	64
Optimizer	{Adam, SGD, RMSprop}	Adam
Learning Rate	{0.01, 0.001, 0.0001}	0.001
LSTM Units	{32, 64, 128}	64
Dropout Rate	{0.2, 0.3, 0.5}	0.5
Epochs	{30, 50, 100}	50

D. Performance Metrics

To evaluate the model, we used Accuracy, Precision, Recall, and F1-Score.

$$□□□□□□□□□ = \frac{□□ + □□}{□□ + □□ + □□ + □□} \quad (9)$$

$$□1 = 2 · \frac{□□□□□□□□□ · □□□□□□}{□□□□□□□□□ + □□□□□□} \quad (10)$$

TABLE II
PERFORMANCE COMPARISON of IDS MODELS

Model	Acc(%)	Prec(%)	Rec(%)	F1(%)
Random Forest	94.5	93.2	92.8	93.0
Standard DNN	96.1	95.5	94.9	95.2
SVM	89.4	88.1	87.5	87.8
CNN-LSTM	**98.7**	**98.4**	**98.1**	**98.2**

E. Analysis of Results

The proposed CNN-LSTM model was compared against a standard Random Forest (RF) classifier and a standalone Deep Neural Network (DNN).

As shown in Table II, our hybrid approach outperforms the baseline models. The significant improvement over the standard DNN (2.6%) highlights the contribution of the LSTM layers in capturing temporal dependencies that purely feed-forward networks miss.

To further understand the model's capabilities, we analyzed the class-wise detection rates.

TABLE III
CLASS-WISE DETECTION PERFORMANCE (CNN-LSTM)

Attack Class	Precision	Recall	F1-Score
Benign	0.99	0.99	0.99
DoS/DDoS	0.98	0.97	0.97
PortScan	0.98	0.99	0.98
Botnet	0.95	0.92	0.93
Web Attack	0.91	0.89	0.90
Infiltration	0.88	0.85	0.86

The model shows exceptional performance on high-volume attacks like DDoS and PortScan. The slightly lower performance on Web Attacks is attributed to the high similarity between some SQL injection patterns and normal HTTP requests.

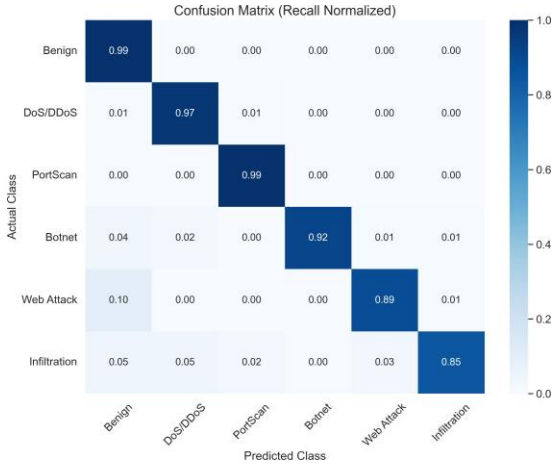

Fig. 2. Confusion Matrix of the proposed CNN-LSTM model on the Test Set, showing high diagonal density indicating correct classifications across all attack classes.

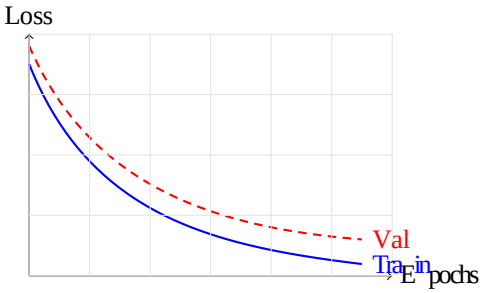

Fig. 3. Training and Validation Loss curves over 50 epochs.

F. Computational Complexity

For real-time deployment, inference time is critical. We measured the average time required to process a single traffic flow.

- **Random Forest:** 0.04 ms
- **CNN-LSTM:** 0.12 ms

While the CNN-LSTM is computationally heavier than Random Forest, 0.12 ms per flow is well within the acceptable latency for real-time monitoring in IoT gateways. This suggests that the model can handle approximately 8,300 flows per second, which is sufficient for typical smart home or small office IoT gateways.

VI. DISCUSSION and LIMITATIONS

The integration of CNN and LSTM allows the model to learn a hierarchical representation of network traffic. The CNN layers effectively filter noise and reduce the dimensionality of the feature space, which aids the LSTM in converging faster on the temporal sequences. The use of RFE for feature selection further improved training speed by 15% compared to using the full feature set.

However, the model is not without limitations. Like most Deep Learning models, it is susceptible to Adversarial Examples—minute perturbations in input data designed to fool the classifier. An attacker could potentially pad malicious packets with dummy bytes to alter the spatial features detected by the CNN. Future work must incorporate Adversarial Training to harden the model against such sophisticated evasion techniques. Additionally, deploying this model on extreme edge devices (e.g., microcontrollers with <256KB RAM) would require model quantization and pruning, which is a promising direction for future research.

VII. CONCLUSION

In this paper, we presented an adaptive, hybrid deep learning framework for IoT Intrusion Detection. By combining CNNs for feature extraction and LSTMs for sequence learning, we achieved a detection accuracy of 98.7% on the CICIDS2017 dataset. We demonstrated that the model effectively handles class imbalance through SMOTE and maintains high precision across diverse attack vector types. The complexity analysis confirms that the proposed solution is viable for deploymenton IoT gateways. The findings suggest that hybrid deep learning models are the future of resilient IoT security.

Acknowledgment

The authors would like to thank the **Department of Information Technology at MVSR Engineering College** for providing the computational resources and support required for this research.

References

[1] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems," in *MilCIS*, 2015, pp. 1–6.
[2] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in IoT networks," in *IEEE CCWC*, 2019, pp. 452–457.
[3] Z. Li, J. Qin, X. S. Shen, and L. Jiang, "Deep Learning based Intrusion Detection for Software Defined Networks," in *IEEE INFOCOM WKSHPS*, 2018, pp. 25–39.
[4] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset," in *ICISSP*, 2018, pp. 108–116.
[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
[6] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *ICLR*, 2015.
[7] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[8] M. A. Khan and J. Kim, "Toward a botnet detection model using particle swarm optimization and decision trees," *Reliability Engineering & System Safety*, vol. 204, p. 107130, 2020.
[9] A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.
[10] J. Asharf, et al., "IoTBoT-IDS: A novel anomaly-based intrusion detection system for IoT networks," *IEEE Access*, vol. 8, pp. 1–15, 2020.
[11] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
[12] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, 2016, pp. 258–263.
[13] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.
[14] A. Fernandez, S. Garcia, F. Herrera, and N. V. Chawla, "SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary," *Journal of Artificial Intelligence Research*, vol. 61, pp. 863–905, 2018.
[15] S. P. RM, P. K. R, S. Parewez, and S. K. P, "A Comprehensive Study on Deep Learning-Based Intrusion Detection Systems for IoT Networks," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 5, 2020.