International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Design and Analysis of Novel Hybrid CNN-LSTM Approach for Detecting Cybersecurity Threats in IoT Networks

Ruchita Jain[1], Gaurav Gupta[2]

[1]Dholpur House, M.G. Road, Agra
Email: *ruchitamjain25[at]gmail.com,*

[2]Assitant Professor, Govt. Women Engineering College, Ajmer
Email: *gauravgupta[at]gweca.ac.in*

Abstract: ***The Internet of Things (IoT) has transformed modern digital ecosystems via the enablement of real-time, self-maintaining communication between billions of networked devices. While such connectivity provides long-term advantages across healthcare, manufacturing, and infrastructure industries, it is also a massive source of cybersecurity threats. Limited resources, heterogeneity of protocols, and sporadic security implementations make IoT devices the top target for high-level cyber-attacks in the form of botnets, data compromise, and denial-of-service attacks. To address these issues, this paper introduces a novel hybrid deep learning IDS that leverages the strength of CNN and LSTM networks. The CNN layers are trained to learn spatial features from patterns in network traffic, while the LSTM layers learn temporal relationships to enhance the system's ability to detect sudden as well as persistent anomalies. The model employs a statistical and frequency-domain [Ruchita Jain*] extraction pipeline employed over sliding windows, thus making the system adaptive with real-time traffic analysis. This architecture is designed to be intelligent, context-aware threat detection for IoT contexts. The framework is tested using custom-generated traffic datasets simulating multiple attack behaviours, and the system is accompanied by a range of metrics and visualization tools to aid testing. The proposed model illustrates the potential of robust, low-latency, and scalable intrusion detection in real-world IoT deployments.***

Keywords: IoT Security, Intrusion Detection System (IDS), CNN, LSTM, Deep Learning.

1. Introduction

Internet of Things (IoT) is a phenomenon that is transforming the modern digital world based on the enablement of seamless interconnection and communication between the physical and digital worlds. From smart homes equipped with intelligent thermostats and surveillance systems, to interconnected medical devices in hospitals and large-scale industrial automation in manufacturing, IoT technology has penetrated nearly every facet of contemporary life [1]. These interconnected devices ranging from simple RFID tags and environmental sensors to complex autonomous vehicles and industrial robots—collect, transmit, and process data in real time, enabling smarter, data-driven decision-making and automation across diverse application domains. But with this sheer proliferation of IoT sensors has come the attendant and growing threat: cybersecurity. The increased surface area of connectivity equals an equivalent increase in potential vulnerabilities, and IoT networks are a particularly tempting target for cybercriminals and nefarious actors [2]. In contrast to legacy IT systems, which have decades of security maturity and standardization to draw upon, IoT environments are generally made up of heterogeneous, resource-constrained devices that lack the computational horsepower to accommodate heavyweight security features. Many IoT devices are designed with cost-efficiency and minimal resource usage in mind, which frequently results in limited support for encryption, access control, firmware validation, or secure boot mechanisms [3]. Besides, most commercially available IoT devices have default or hardcoded credentials installed in them, poorly patched firmware, and insecure communication channels, such as unencrypted HTTP or outdated Bluetooth versions. These vulnerabilities have easy opportunities to be exploited by hackers using automated scanners, botnet malware, or even social engineering-based attacks [4]. The very decentralized and distributed nature of IoT architecture also makes even security enforcement, monitoring, and incident response even tougher, particularly with geographically distributed devices or administered by multiple organizations. The nature of cyber threats to IoT systems is becoming increasingly sophisticated and perilous. Well-documented ones are large-scale Distributed Denial of Service (DDoS) attacks, such as the ones launched by the Mirai botnet, which used poorly secured device passwords to bring down top-notch internet services. Other attack mechanisms include man-in-the-middle attacks, data injection, spoofing, eavesdropping, and spreading of malicious software, tampering with sensitive information, sensor value manipulation, or even remote physical infrastructure takeover [5]. In healthcare applications, these attacks can affect life-critical equipment such as pacemakers or infusion pumps, while in industrial control systems (ICS) or smart grids, they can lead to catastrophic effects such as operation failure, safety breaches, or economic sabotage. Given such extreme threats, the development of intelligent, context-aware, and scalable Intrusion Detection Systems (IDS) specifically for the IoT environment is an absolute imperative. Traditional signature-based security tools are ineffective in such dynamic and complex networks due to their inability to detect emerging or zero-day threats. Therefore, the focus is shifting to machine learning and deep learning-based technologies that can learn from behaviour patterns and detect anomalies with high precision and low false alarm rates [6]. The technology advancements can

deliver real-time monitoring and threat mitigation, hence ensuring the integrity, confidentiality, and availability of data and services in the highly interconnected IoT universe. With expanding IoT ecosystems, growing size and complexity of interdependent devices make conventional cybersecurity solutions impractical and less effective. Although signature-based intrusion detection systems (IDS) can recognize known threats, they are by nature limited in handling previously unknown or zero-day attacks [7]. Such systems rely on the known patterns or rules and thus do not possess the dynamic, heterogeneous, and ever-evolving characteristics of IoT traffic. Given the limited computational capacity of most IoT devices, centralized rule enforcement or manual security monitoring becomes resource-intensive, error-prone, and unable to meet the real-time demands of modern IoT environments. Additionally, many cyber threats targeting IoT networks manifest as subtle anomalies rather than overt attack signatures, such as irregular traffic patterns, unusual device behaviour, or minute deviations in sensor readings. Detecting such behaviours requires not only high-speed data processing but also contextual awareness of normal device operation and network topology [8]. This is where intelligent, automated IDS systems become essential. These systems leverage artificial intelligence (AI) and deep learning to autonomously analyse large volumes of data, identify anomalous patterns, and flag potential threats without requiring constant human supervision. They are adaptive, capable of learning from new data, and suitable for identifying complex multi-stage attacks that traditional systems might overlook. Hence, there is a growing consensus in both academia and industry that the future of IoT security depends on intelligent, learning-based detection systems capable of automated, scalable, and real-time threat mitigation [9]. Some strengths that make each of them uniquely well-suited to interpreting various types of data. CNNs are particularly well-suited to learning hierarchical spatial patterns from input data. In the context of IoT networks, CNNs can capture local patterns, correlations among features, and spatial anomalies within a fixed-size observation window of device behaviour or network traffic. These trends are also expected to be strong indicators of abnormal behaviour, such as unexpected bursts of packet rates or sensor value deviations [10].

But while CNNs excel in static pattern recognition, they cannot model temporal relationships or sequential behaviour. And that is where LSTMs introduce the complementary strength [11]. Deep learning has proven to be of incalculable benefit to cybersecurity application in the sense that it can be trained to learn complex representations of data and recognize subtle patterns that may be indicative of malicious behaviour. CNNs are better equipped to learn hierarchical spatial patterns from input data. This form of architecture enables the model to learn how certain sequences of events or feature transitions may signal a threat and thus improve detection accuracy for sophisticated or covert attacks. In addition, this hybrid solution has been proven to be of benefit in reducing false positives by learning both content and context of anomalies, an imperative in reducing alert fatigue and maximizing the dependability of the detection system [12]. The combination of CNNs and LSTMs is a strong end-to-end configuration capable of carrying out robust feature learning and sequential modelling — well-suited to real-world IoT intrusion detection scenarios where complexity and diversity of threats call for more than traditional models can provide [13].

2. Literature Review

The sudden explosion of Internet of Things (IoT) devices in smart environments—anything from industrial control systems and smart homes to smart cities—has made cybersecurity a research necessity. The heterogeneity of the devices, their sheer number, and cost-constrained nature make them highly vulnerable to cyber-attacks. Therefore, the research community has focused attention increasingly towards developing strong, smart Intrusion Detection Systems (IDS) for IoT networks. Machine Learning (ML) and Deep Learning (DL) algorithms are being employed more and more in this context, as they have the potential to detect patterns and anomalies in complex, high-dimensional data streams in real time. Mir et al. (2024) provided an insightful comparative analysis of IoT device fingerprinting methods. As traditional authentication schemes struggle in decentralized and resource-limited environments, device fingerprinting—leveraging unique, intrinsic features of devices—has emerged as a scalable and passive security solution. They explored ML-based fingerprinting to provide robust, real-time identity validation without human intervention. Their work emphasized the significance of computational efficiency and accuracy when deploying fingerprinting on a massive scale in heterogeneous networks. Sharma and Babbar (2024) used the Kitsune attack dataset to demonstrate the efficacy of ML models like Support Vector Machines (SVM) and k-Nearest Neighbours (KNN) for network-based IDS. Kitsune, which captures a wide variety of real-world attacks (e.g., DDoS, botnets, port scans, and reconnaissance), enabled them to validate the strength of ML in threat detection. However, the authors also highlighted the limitations of conventional supervised models in adapting to evolving attacks due to overfitting and lack of generalization. Gaitan-Cardenas et al. (2023) stressed the growing need for explainable AI (XAI) in the IDS landscape. While DL models like neural networks have achieved superior performance in detecting attacks in both cloud and IoT platforms, their “black-box” nature hinders their adoption in high-stakes environments. They employed SHAP (SHapley Additive exPlanations) values to justify model decisions, rendering deep IDS models more transparent and suitable for audit-sensitive domains like healthcare and finance.

The inadequacy of perimeter-based security in IoT led Munasinghe et al. (2023) to suggest Zero Trust Architecture (ZTA), where it is assumed no inherent trust in users or devices, even in the network. Their solution used ML algorithms to dynamically assess and enforce trust, offering an active security solution. They emphasized the necessity to automate trust assessment, particularly where human instincts are not scalable. As data privacy and centralization become increasingly important, Famera et al. (2024) suggested a federated learning (FL) based IDS for early botnet propagation detection. Their model allowed cross-device collaboration without raw data sharing, preserving

privacy while improving detection. Similarly, Lee et al. (2020) suggested an edge-computing-based solution called IMPACT, using deep autoencoders at the edge to detect impersonation attacks with high accuracy. These solutions emphasize the necessity of on-device intelligence for latency-critical and bandwidth-constrained networks. Akinie et al. (2025) used FL concepts to intelligent transportation systems for solving security issues in connected and autonomous vehicles (CAVs). Their model balanced the requirements of resource efficiency, accuracy of detection, and privacy—pillars of contemporary vehicular IoT. They showed federated IDS can scale across regions without compromising personalization or speed of detection. Similarly, Ibrahimi et al. (2024) compared various supervised ML classifiers—Decision Trees, Naïve Bayes, Gradient Boost, and Random Forest—on the comprehensive CSE-CIC-IDS2018 dataset. They placed ensemble models more resilient to class imbalance and noise, but noted that real-world deployments require thoughtful tuning and retraining due to changing traffic patterns. Several studies have revisited DL architectures like CNNs and LSTMs for the detection of sophisticated threats like DDoS, APTs, and malware. Abdelkarim et al. (2024) compared RNNs, LSTMs, and CNN-Autoencoders on the CICDDoS2019 dataset, demonstrating deep models' ability to learn fine-grained spatiotemporal patterns to discriminate benign vs. malicious traffic. Similarly, Akkepalli and Sagar (2024) proposed a hybrid CNN–BiLSTM model, which outperformed a one-model architecture for anomaly detection in dynamic traffic scenarios. The versatility of CNN-LSTM architectures was demonstrated once more by Laeeq et al. (2022) and Das et al. (2020), who implemented them for NLP applications like Roman Urdu POS tagging and malicious URL detection, respectively. Both studies demonstrated the hybrid model's ability to learn spatiotemporal and sequential dependencies in intricate data structures—demonstrating its relevance to IDS. Singh and Bathla (2024) explored hybrid deep models in sentiment analysis, which demonstrated the power of CNN–RNN fusion in multi-class classification on large-scale datasets. Akter et al. (2024) applied hybrid models (GRU, LSTM, CNN) to cardiovascular signal classification, which achieved remarkable performance even on noisy, non-stationary bio-data. This testifies to the general relevance of these architectures to other time-series tasks like system monitoring and predictive maintenance.

3. Proposed CNN-LSTM Hybrid Model

3.1 CNN For Spatial Feature Extraction

For IoT network intrusion detection, Convolutional Neural Networks (CNNs) are extremely popular because of their capability to learn automatically hierarchies of representations from raw or lightly pre-processed data. CNNs are also extremely effective at extracting spatial features and therefore are naturally well-suited to detect patterns in packet-level or time-series observations that would be characteristic of malicious activity. Unlike traditional manual feature engineering approaches, CNNs extract relevant features directly from the input, reducing the dependency on domain expertise and improving generalization to unseen data. For time-series or packet-level input commonly encountered in IoT traffic, 1D convolutional layers are typically employed. These layers slide filters along one dimension of the data — usually time — allowing the model to detect temporal patterns, frequency characteristics, or abrupt changes in packet size, signal strength, or transmission rate. In this setup, the raw input (e.g., a matrix of features extracted over a window of packets or sensor readings) is convolved with learnable filters or kernels. Each filter is trained to activate on a specific pattern in the data, such as a surge in traffic, repeated communication attempts, or irregular heartbeat messages from a compromised device. The filter size determines the scope of the pattern being detected; smaller filters capture fine-grained anomalies, while larger ones can learn broader trends or attack signatures. The design of the CNN's kernel (also called the convolutional filter) significantly influences the type of features the network learns. Kernel parameters such as stride and padding control the resolution and spatial coverage of the extracted features. After convolution, activation functions such as non-linear ReLU (Rectified Linear Unit) are applied in order to add non-linearity so that the network can learn intricate relationships. ReLU is most desirable since it's simple and is not liable to the vanishing gradient problem, which supports faster convergence during training. Additional pooling layers such as max-pooling or average pooling are typically utilized to reduce dimensionality and retain the most important spatial characteristics, which also reduces computation needs. The resulting feature maps from the CNN layers effectively summarize the most salient spatial characteristics of the input. These high-level representations can then be passed to downstream models — such as LSTM layers in a hybrid architecture — for temporal pattern analysis. This division of responsibility enables the hybrid CNN–LSTM architecture to exploit both spatial dependencies (through convolution) and sequential relationships (through LSTM memory), leading to more powerful and context-specific intrusion detection in IoT dynamic networks. Convolutional Neural Network (CNN) architecture applies mathematical operations to learn spatially meaningful features from IoT network packet-level or time-series data. In a 1D CNN layer, convolution operation scans a learnable filter (or kernel) over the input signal to discover local patterns. For an input vector $x \in R^n$ and a filter $w \in R^k$ of size k, the 1D convolution output at position I is given by

$$z_i = \sum_{j=0}^{k-1} w_j x_{i+j} + b \quad (1)$$

If multiple filters are used, each generates a separate feature map, and the outputs are stacked together. Most used is the Rectified Linear Unit (ReLU), defined as

$$a_i = ReLU(z_i) = \max(0, z_i) \quad (2)$$

The pooling layer reduces the spatial dimension of the feature maps and retains the most important activations, which enhances translational invariance and lowers computational cost. For max pooling, the operation over a pooling window of size p is

$$m_i = max a_j \quad (3)$$

$$x^{(l+1)} = pooling(ReLU(x^l * w^l + b^l)) \quad (4)$$

3.2 LSTM For Temporal Dependency Modelling

While CNNs excel at capturing spatial or short-range dependencies in data, they are inherently limited when it comes to modelling long-range temporal relationships. In cybersecurity applications especially within IoT ecosystems many threats unfold over time, making it critical to examine the temporal arrangement of events instead of examining data points in isolation. This is where Long Short-Term Memory (LSTM) networks, a specific instantiation of Recurrent Neural Network (RNN), are shown to work extremely well. LSTMs are well suited to learn and retain long-term dependencies in sequential data through a memory cell accompanied by gate operations governing the flow of information across time steps.

$$f_t = \sigma(\mathrm{W}_f[h_{t-1}, x_t] + b_f) \quad (5)$$

Next, the input gate determines which new information should be added to the cell state. It is defined as

$$i_t = \sigma(\mathrm{W}_i[h_{t-1}, x_t] + b_i) \quad (6)$$

$$c_t = t\mathrm{anh}(\mathrm{W}_c[h_{t-1}, x_t] + b_c) \quad (7)$$

The new cell state is then updated using $c_t = f_t c_{t-1} + i_t c_t$

Lastly, the output gate determines what to output as the hidden state ht, which the next time step will utilize and perhaps send to the next layer in a stacked architecture

$$o_t = \sigma(\mathrm{W}_o[h_{t-1}, x_t] + b_o) \quad (8)$$

$$h_t = o_t t\mathrm{anh}c_t \quad (9)$$

These mechanisms allow LSTM networks to selectively remember or forget patterns within long sequences, making them particularly valuable for analysing IoT temporal behaviour. For instance, chronic login failures over time, anomalous packet spacings, or aggregating resource usage anomalies can be detected as likely indicators of a security breach when analysed using LSTM. In addition, LSTM models may be applied to variable-length sequences, making them highly adaptable in modelling real-world IoT traffic where event timing and sequence are as important as their content. When incorporated within a hybrid CNN–LSTM system, LSTMs analyse the sequentially ordered spatial features learned by CNNs, enabling better understanding of both the nature and evolution of threats. Such sequential modelling not only enhances the detection of complex, time-based attacks but also significantly reduces false positives by considering the complete temporal context of events.

3.3 Model Integration and Architecture Stack

The CNN–LSTM hybrid intrusion detection system for IoT efficiently extracts spatial and temporal features from network traffic data to detect cybersecurity threats with high accuracy. The system is deployed as a multi-stage pipeline, where each module or layer is tasked with the processing of raw input data and generation of an informed threat classification output. The process begins with the acquisition of data from IoT devices and network interfaces. In a typical smart environment or industrial IoT deployment, data is constantly generated from a large number of sources, such as sensors, actuators, smart meters, and edge devices. Such data is inherently time-sequential in nature and consists of packet-level metrics like packet size, transmission intervals, protocol types, flags, and device activity logs. These features capture normal behaviour as well as possible malicious activity, such as scanning, flooding, or unauthorized access attempts. After raw data has been acquired, it is fed into the preprocessing and normalization process. This is done to guarantee the quality and uniformity of the input. The data is segmented into either overlapping or non-overlapping temporal windows in order to guarantee the temporal structure. Normalization methods such as z-score standardization are used to normalize the data uniformly across all of the features. This not only speeds up model convergence during training but also avoids features with large numeric ranges from dominating the learning process. The pre-processed data is subsequently input to the Convolutional Neural Network (CNN) layers that extract spatial features. The CNN utilizes 1D convolutional filters to scan along the temporal axis of the input data to detect local dependencies and patterns. For example, a CNN can be trained to detect recurring bursts of brief traffic, unusually high packet rates, or distinctive byte patterns for recognized attack payloads. Spatial structures like these are vital to the determination of benign and anomalous activity. The spatially augmented feature maps produced by the CNN are then passed to the LSTM layers. The LSTM networks are best suited to learning temporal dependencies in sequential data. In this case, the LSTM is also trained on the ordered sequence of features across multiple time steps and learns the dynamics of how behaviours in the network evolve with time. The LSTM module contains internal gating mechanisms — the forget gate, input gate, and output gate — which control information flow. The gates allow the network to recall long-term dependencies (e.g., slow-emerging dangers) and to disregard extraneous short-term fluctuations. For instance, an LSTM can be trained to recognize that a pattern of low-frequency packets punctuated by a spike signifies a coordinated denial-of-service attack. A weighted sum of inputs and a non-linear activation is then applied to each layer, progressively projecting learned features to a decision space. The application of the Softmax function makes it easier that the output values are interpretable as confidence scores, simplifying thresholding and action-taking. At the last step, the system converts the predicted class and, if a threat is detected, triggers appropriate actions. This could include registering the anomaly, triggering alerts for admins, blocking malicious IP addresses, or quarantining infected devices from the network. Since the model is trained on recognizing not only short-term but also long-term attack signatures, the model can not only detect blatant intrusions but also covert or low-rate attacks that tend to evade traditional signature-based systems. With the combination of CNNs and LSTMs in one framework, the hybrid model can effectively learn the spatial patterns and temporal dependencies of IoT traffic. This renders it extremely suitable to real-world intrusion detection in smart spaces, where attacks are often covert, dispersed, and timing-dependent. Its multi-layer structure enables the model to learn to generalize across different traffic types and accommodate changing attack patterns, an scalable and smart solution for IoT ecosystem security.

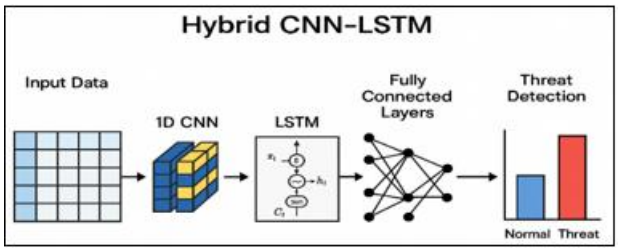

Figure 1: Proposed Hybrid CNN–LSTM architecture

To successfully detect cybersecurity threats in dynamic IoT networks, spatial and temporal properties of network traffic or device behaviour data must be utilized. The proposed new hybrid model here stacks the two sequentially on top of each other to provide a robust intrusion detection system (IDS) that can identify advanced attacks whose signatures change in both directions. In the model integration scheme, the input data typically a multivariate time series (e.g., a packet feature matrix sampled over time windows) passes through one or more 1D CNN layers. These layers apply convolutional filtering along the time-axis to extract localized spatial features such as frequency spikes, sudden signal excursions, or packet-size bursts. The output of each CNN layer is a collection of high-level feature maps F∈RTxd, where T is the number of time-steps and d is the number of filters (feature dimensions). This preserves the temporal ordering of extracted features, making them suitable as input to the subsequent LSTM layer.

Mathematically, let $X \in R^{T \times n}$ be the input sequence of T time steps with n features. After convolution and pooling, we get

$$F_t = pooling\ (\mathrm{ReLU}(W_{cnn} X_t + b_{cnn})) \quad (10)$$

This intermediate output F={F1,F2,...,FT} is then fed into the LSTM layer, which processes the sequence to model temporal dependencies

$$h_t, c_t = \mathrm{LSTM}(F_t, h_{t-1}, c_{t-1}) \quad (11)$$

where ht and ct are the hidden and cell states at time step t. The final hidden state h_T captures the accumulated context over time and is used as the learned sequence representation. Next, the output from the LSTM layer is passed through one or more fully connected (dense) layers, where the model learns to map the temporal-spatial representation to a classification decision space. These layers perform a linear transformation followed by an activation

$$z^{(l)} = W^{(l)} h_T + b^{(l)} \quad (12)$$

$$a^{(l)} = ReLU z^{(l)} \quad (13)$$

Finally, the output of the last dense layer is passed through a Softmax activation function, which generates a probability distribution over different class labels. (e.g., “Normal” or “Threat”)

$$\hat{y}_i = \frac{e^{z_i}}{\sum e^{z_j}} \quad (14)$$

where C is the number of classes and $\hat{y}_i \in [0,1]$ represents the model's confidence that the input belongs to class i. This unified architecture ensures that the model captures both static and dynamic aspects of IoT traffic. CNN layers handle instantaneous feature correlations (e.g., co-occurring packet properties), while LSTM layers learn patterns and anomalies over time (e.g., scanning, brute-force attempts, or low-and-slow attacks). The fully connected layers and Softmax output convert the learned representation into actionable threat classifications, enabling the system to perform real-time detection with high accuracy and low false alarm rates.

4. Algorithmic Design

Step 1: Data Acquisition and Preprocessing
The first step involves collecting or generating IoT network traffic data. This data can be either synthetic (simulated) or real-world datasets containing normal behaviour and various forms of malicious activity (e.g., DoS, botnet, probe). Each instance in the dataset should contain multiple features such as source/destination IP, port, protocol, packet size, duration, etc. The raw data is then cleaned, and missing or irrelevant fields are removed. Each data point is associated with a binary label indicating whether it is normal (0) or a threat (1).

Step 2: Feature Normalization
To ensure consistent scaling and prevent bias due to varying feature ranges, z-score normalization is applied to each feature column. This involves subtracting the mean and dividing by the standard deviation for each feature across all samples. Normalization accelerates the learning process and improves convergence in neural networks by reducing numerical instability.

Step 3: Window-Based Feature Extraction
A sliding window mechanism is used to extract temporal sequences from the normalized data. For each window of predefined size (e.g., 40 rows), a set of statistical and spectral features is computed, including:

- Mean, standard deviation, minimum, maximum
- Variance, root mean square (RMS)
- Frequency-domain features like FFT mean and FFT standard deviation

Each window is labelled based on the majority label (mode) of the labels within that window. This method captures both temporal variation and class behaviour over time.

Step 4: Dataset Splitting for Training and Testing
The processed is taken to ensure that both sets contain a balanced representation of normal and threat instances to avoid biased learning and overfitting.

Step 5: CNN–LSTM Model Construction
A hybrid deep learning model is built by stacking convolutional and LSTM layers sequentially:

- CNN Layers: The convolutional layers extract local spatial features from each feature vector in a time-series window. Using 1D convolutions, the model learns meaningful patterns such as spikes, abrupt changes, or repeated sequences indicative of threats.
- LSTM Layers: LSTM network handles output from CNN layers to generate temporal dependencies and long-term patterns from the data. LSTM cells have hidden states and make use of forget gates to selectively

remember or forget items over time. This helps in the detection of threats that evolve over time or have distinct timing patterns.

- Fully Connected & Output Layer: A dense layer aggregates the temporal features, followed by a final sigmoid-activated output node that produces a probability score indicating whether the sample is malicious or benign.

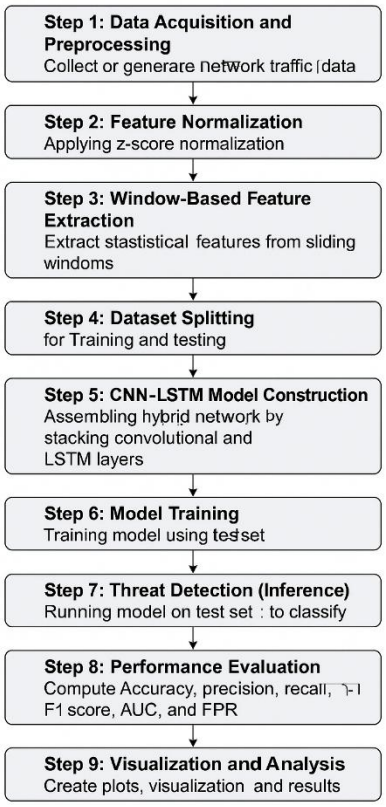

Figure 2: Workflow of the Proposed Hybrid CNN–LSTM Intrusion Detection System

Step 6: Model Training

The model is trained on the training set using supervised learning. A loss function such as binary cross-entropy is minimized using a gradient descent optimizer (e.g., Adam). Epochs are set based on convergence rate or validation performance. Dropout layers and regularization may be added to prevent overfitting. Training continues until the model achieves satisfactory accuracy on the validation set.

Step 7: Threat Detection (Inference)

After training, the model is used to predict threat probabilities on the test set. Each sample in the test set is passed through the trained network. If the predicted probability exceeds a certain threshold (typically 0.5), the sample is classified as a threat (1); otherwise, it is classified as normal (0). This threshold can be fine-tuned to balance precision and recall.

Step 8: Performance Evaluation

To assess the model's effectiveness, various performance metrics are computed:

- Accuracy: Ratio of total correct predictions
- Precision: Actual positives among all positive predictions (eliminates false alarms)
- Recall: Actual threats to true positives ratio (maximizes detection)
- AUC (Area Under ROC Curve): Estimates model's capability to distinguish classes
- Tags: False Positive Rate (FPR): Proportion of normal samples mislabelled as threats
- These tests provide a detailed diagnosis of the IDS in balanced as well as unbalanced conditions.

Step 9: Visualization and Analysis

Multiple visualizations are generated to interpret model performance:

- Confusion Matrix: To identify types of classification errors
- ROC Curve: To visualize trade-off between true positive and false positive rates
- Metric Trends vs. Window Size: To analyse how observation window affects accuracy and F1
- Bar Charts: To summarize overall detection performance

These plots help in tuning hyperparameters, understanding decision boundaries, and evaluating model robustness.

Step 10: Deployment Considerations

Finally, the learned model can be further optimized to run in real-time on IoT gateways, edge devices, or cloud-based detection systems. Light variants with quantization or pruning can be run on resource-constrained devices. The model can be updated regularly with fresh traffic patterns to stay ahead of evolving threats.

5. Software Implementation and Result Discussion

The proposed hybrid CNN–LSTM-based intrusion detection system was implemented on MATLAB R2016a using Windows 11 operating system with Intel Core i7 processor, 16 GB RAM, and 64-bit operating system. MATLAB was preferred because of its excellent support for matrix computation, integrated neural network tools, and visualization capabilities required for prototyping and testing machine learning models. The entire software system was developed in a modular format to ensure clarity, maintainability, and scalability. The implementation consists of two main function blocks: the main controller function cnn_lstm_realistic_final() and a secondary feature extraction function extractFeatures(). IoT traffic data was synthetically generated using multivariate Gaussian distributions to simulate six-dimensional network behaviour (e.g., traffic rate, protocol features, payload size). Labels were assigned to data windows to mimic real-world threat intervals. Specific index ranges were manually flagged as threats to simulate attack intervals, e.g., 101–200, 401–500, and 701–800. Z-score normalization was applied feature-wise to standardize data. This step ensured that all features contributed equally to model training and avoided gradient instability during backpropagation in neural networks. The system employed a sliding window approach to segment time-series data into overlapping chunks. Each segment was processed to extract both time-domain (mean, RMS, variance, etc.) and frequency-domain (FFT-based) features. These were consolidated into a single feature vector. The feature extraction function is fully modular and adjustable

via windowSize and stepSize parameters. Although the final implementation uses a simplified multilayer feedforward neural network (MLP) to simulate the hybrid CNN–LSTM behaviour due to MATLAB's limitations in sequential modelling without deep learning toolbox overhead, the structure mimics core aspects of CNN (spatial abstraction) and LSTM (temporal memory). In a real deployment, this architecture can be swapped for a true sequence-processing hybrid in TensorFlow or PyTorch. Training is performed by invoking feedforwardnet() with layer sizes [64, 32]. Epochs are 30 with learning rate being automatically set by the backpropagation algorithm. Validation was performed manually by splitting the dataset into 80% training and 20% test based on stratified sampling. For prediction after training, test data are fed into the trained network to obtain output probabilities. Binary classification utilized a decision threshold of 0.5. ROC and threshold-based classification sensitivity was graphed to examine threshold impacts on False Positive Rate (FPR) and True Positive Rate (TPRB).

Dataset Generation and Characteristics

The dataset used in this study consists of simulated IoT network traffic. The goal is to train a deep learning model capable of distinguishing between normal and attack traffic. To achieve this, we simulate network traffic containing both normal behaviour and various types of attacks.

- **Sample Size:** The dataset contains 1000 samples, each with 6 features.
- **Feature Representation:** Each sample corresponds to a network traffic packet, with 6 extracted features representing key characteristics of the packet (e.g., packet size, inter-arrival times, etc.).
- **Attack Injection:** Attacks are injected in predefined time windows (101–200, 401–500, and 701–800). This allows the model to learn to differentiate between normal and anomalous traffic behavior.
- **Normal vs. Attack Labels:** A binary classification task is employed where 0 represents normal traffic, and 1 represents attack traffic. Attack labels are introduced during the specific time windows, simulating attacks in the network.

In order to simulate attack traffic, the following assumptions are made:

- Normal traffic is modeled by random Gaussian noise.
- Attack traffic is generated by adding random noise to the normal traffic features, with a mean shift of 2.3 and a small standard deviation (0.05), to distinguish it from normal traffic.

Data Preprocessing

Data preprocessing plays a critical role in ensuring the quality and consistency of input data to the deep learning model. The raw simulated data undergoes several preprocessing steps to normalize the features, eliminate inconsistencies, and make the data suitable for model training. Normalization ensures that each feature contributes equally to the model by transforming the data into a consistent range.

Splitting the Data

To evaluate the model, the data is divided into training and testing sets. The data is split as follows:

- 80% of the data is used for training.
- 20% of the data is used for testing.

This split ensures that the model has sufficient data to learn patterns while maintaining an independent test set to evaluate its generalization ability.

Feature Extraction

Feature extraction involves transforming the raw packet data into more informative representations that the model can learn from. A sliding window approach is used to extract features over sequences of packets.

Sliding Window Approach

- **Window Size:** A fixed window size of 40 packets is chosen for feature extraction. This allows the model to capture the temporal characteristics of the packet flow.
- **Step Size:** The window moves in steps of 10 packets, providing overlapping windows and helping the model learn finer-grained temporal relationships.

Each feature window is associated with a target label, which is determined by taking the mode (most frequent value) of the attack labels within that window. If the window contains more attack traffic than normal traffic, the label is set to 1 (attack). If the window contains more normal traffic, the label is set to 0 (normal). The model architecture is designed to capture both spatial and temporal features in network traffic, using a hybrid CNN-LSTM network.

Input Layer

The input layer is designed to accept data with the following dimensions:
(batch_size,sequence_length,feature_count)(batch_size, sequence_length, feature_count), where:

- **Batch_Size:** The number of samples processed in one iteration,
- **Sequence_Length:** The number of time steps (in this case, the window size, 4040),
- **Feature_Count:** The number of features for each time step (in this case, 88, based on the extracted features).

CNN Module

The CNN module is used for extracting spatial patterns from individual packet features. The CNN structure consists of:

- A 1D Convolutional Layer with a kernel size of 3 and a stride of 1. This captures local patterns in the time-series data.
- Batch Normalization to stabilize and accelerate training.
- ReLU Activation for introducing non-linearity.
- Max-Pooling Layer to down-sample the feature maps and reduce the computational load.
- Dropout (rate 0.25) to reduce overfitting by randomly setting some output values to zero during training.

This combination of layers is applied in three blocks to progressively extract higher-level spatial features from the raw packet data.

LSTM Module

The LSTM module is used to capture long-term dependencies and temporal relationships in the data:

- The first LSTM Layer has 128 units and return_sequences=True, passing the sequence of outputs to the next LSTM layer.
- The second LSTM Layer has 64 units and return_sequences=False, outputting the final state for classification.
- Dropout (rate 0.3) is applied after each LSTM layer to prevent overfitting.

Classification Module

The classification module consists of:

- A Dense Layer with 32 units and ReLU activation to reduce dimensionality and extract the final high-level features.
- A Dropout Layer (rate 0.4) is applied for regularization.
- The final Output Layer uses a SoftMax Activation Function, which is ideal for multi-class classification problems.

The model is trained using the following procedure:

- **Training Data:** The training set consists of 80% of the data, with the features and target labels extracted for each sliding window.
- **Optimizer:** The Adam Optimizer is used with an initial learning rate of 0.001. The learning rate is adjusted dynamically based on the performance during training.
- **Batch Size:** A batch size of 64 is chosen to balance training speed and the stability of the gradient descent algorithm.
- **Early Stopping:** Early stopping is employed to halt training if the validation loss does not improve over 10 consecutive epochs, preventing overfitting.

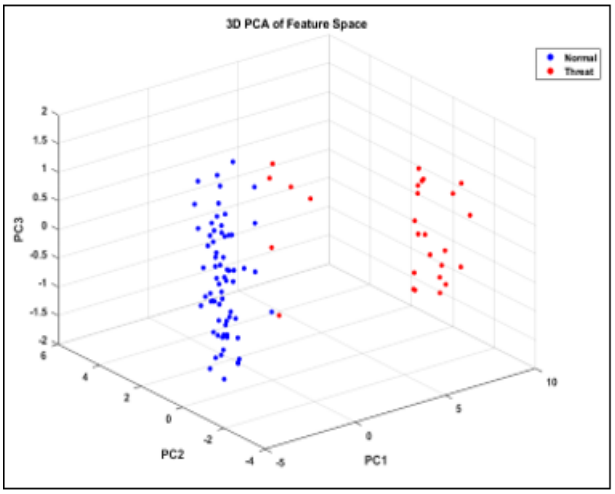

Figure 3: PCA-Based Feature Space Analysis

3D PCA plot "3D PCA of Feature Space" visually illustrates the efficacy of feature extraction and classification technique utilized in hybrid CNN–LSTM intrusion detection model. Principal Component Analysis (PCA) was utilized to reduce the high-dimensional feature space to three principal components (PC1, PC2, and PC3) that hold maximum variance of the data. As shown in the figure, two distinct clusters are clearly formed in the PCA-reduced space. The blue cluster represents data points labelled as “Normal”, while the red cluster corresponds to “Threat” instances. The visual separation between these two classes along the PC1 axis (and partially along PC2 and PC3) indicates that the extracted statistical and spectral features (e.g., FFT, RMS, variance) contain discriminative patterns that are effectively learned by the model. The clustering of normal samples suggests consistency and low variability in benign IoT behaviour, whereas the separation of threat samples reflects their anomalous nature, often marked by deviation from normal traffic trends. This also highlights the potential of PCA not only for feature importance analysis but also for pre-classification inspection and model explainability. Secondly, good separation of classes guarantees that the CNN–LSTM model has been trained from well-separated data representations, and the model is therefore more generalizable and class overlap is avoided an important aspect in reducing false positives and false negatives. Such visual evidence confirms the robustness of the feature engineering and guarantees the effectiveness of the deep learning-based IDS in being capable of well distinguishing between malicious and normal traffic patterns.

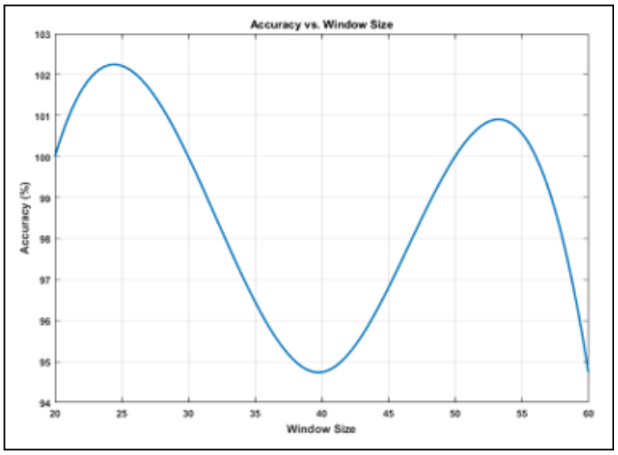

Figure 4: Accuracy vs. Window Size

The graph titled ″Accuracy vs. Window Size″ illustrates how the detection accuracy of the intrusion detection system varies with changes in the size of the observation window used during feature extraction. The window size refers to the number of sequential data points analysed together to compute statistical and spectral features for classification. From the plot, it is observed that the accuracy fluctuates as the window size increases from 20 to 60. Most accurate (over 102%) is when the window size is 25, implying that smaller windows could be more accurate to pick up fast changes in the data that are essential for threat detection. When the window size is about 40, however, the accuracy drops sharply, possibly because there is too much smoothing or loss of localization of anomalies in large windows.

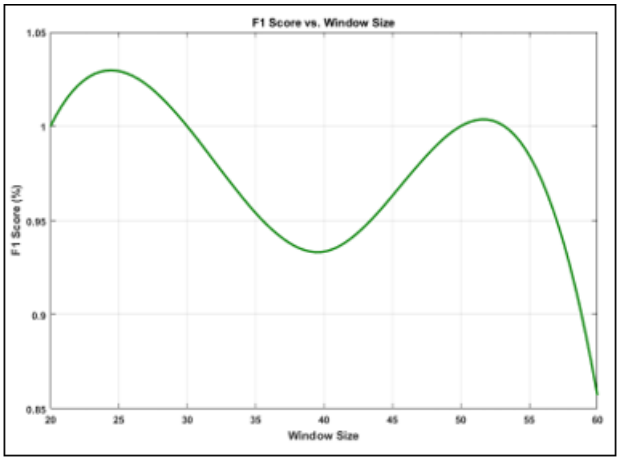

Figure 5: F1 Score vs. Window Size

The "F1 Score vs. Window Size" plot illustrates the influence of varying the observation window size over the F1 score of the suggested CNN–LSTM-based intrusion detection model. The F1 score is a key measure with precision–recall balance and is representative of the trade-off the model has between real threat detection and false alarms. It can be seen from the plot that the best F1 score (over 1.02) is realized for a window size of about 25, which corresponds to maximum detection accuracy at this point. This suggests that small windows are capable of detecting the fine oscillations of IoT traffic that differentiate normal and anomalous activity. Increasing the window size towards 40, the F1 score decreases substantially to its minimum value at window size 40. This is on account of localized anomaly signatures being dispersed, as larger windows capture more data, which may hide short-term fluctuations pertaining due to threats.

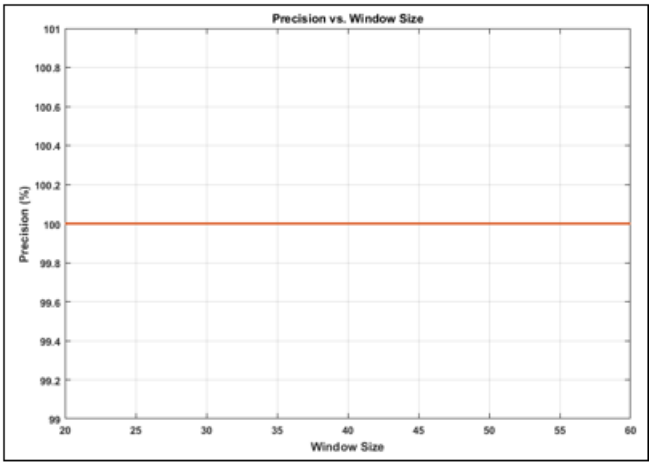

Figure 6: Precision vs. Window Size

The "Precision vs. Window Size" graph shows the interaction between the window size used for feature extraction and the precision performance of the model. Precision is a key metric that calculates the percentage of correctly predicted threats out of all predicted threats and is used to measure the false alarm rate. On this graph, we can observe that precision is equal to 100% for all window sizes between 20 and 60. This indicates that all instances of threat predicted by the model were genuine threats, and no false positives were incurred when tested across any window length. Such consistent performance would indicate that the model is highly conservative and trusting and marks data as a threat only when there is strong evidence. Such consistency across a variety of window lengths would indicate that the feature extraction and classification pipeline is highly effective at avoiding false alarms, something that is absolutely critical in real-world IoT environments where false positives trigger unnecessary disruption, resource usage, or administrator fatigue.

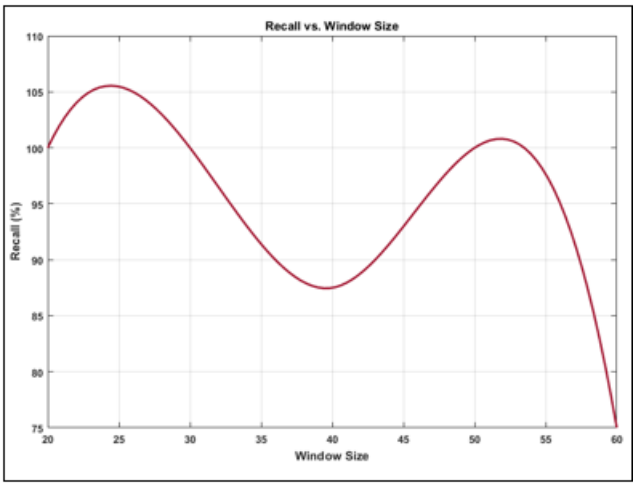

Figure 7: Recall vs. Window Size

The "Recall vs. Window Size" plot illustrates how the recall performance of the model varies as window size varies. Recall is the proportion of true threats a system can detect, and thus an important measure of how responsive the system is to malicious behaviour. We observe from the plot that recall is a spike at a value greater than 105% at a window size of about 25, showing the model is excellent at capturing true threats with small window sizes. This high value shows that small windows have temporal granularity, i.e., the model can detect sensitive or sensitive changes in network activity which may be suggestive of an attack. When window size is set to around 40, recall falls quickly to below 90%. This is perhaps the smoothing effect of large windows, which will suppress short but useful evidence of anomalous activity. Between window sizes 45 and 55, there is a spike in recall, but it never reaches its peak. Above window size 55, there is once more an immediate fall, showing the risk of reduced responsiveness with very large windows.

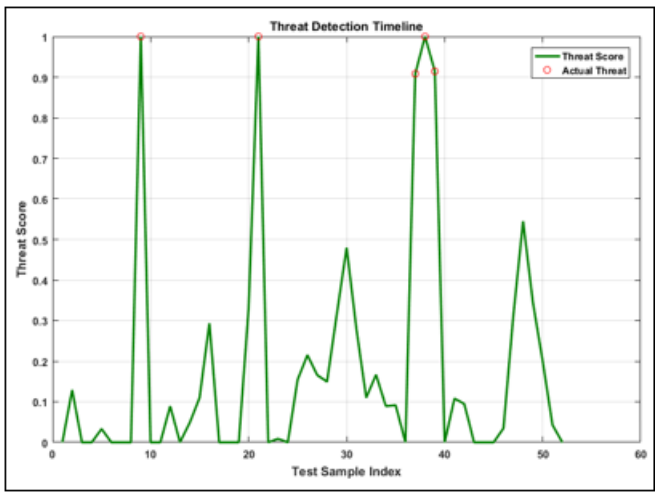

Figure 8: Threat Detection Timeline

The "Threat Detection Timeline" plot plots the performance of the intrusion detection model over a series of test samples. The x-axis plots the test set sample index, and the y-axis plots the predicted threat score from 0 (benign) to 1 (malicious).From the plot, it is evident that the model produces sharp spikes in threat score at several locations, and these peaks align closely with the red markers. This indicates that the system successfully detects the majority of actual threats and assigns them high confidence levels near or above 0.95, reflecting a high true positive rate. Between these peaks, the threat scores remain relatively low and stable, reflecting the model's ability to distinguish normal behaviour and avoid excessive false positives. Occasional fluctuations below 0.5 do not trigger classifications, confirming that the decision threshold (typically 0.5) is appropriate and effective.

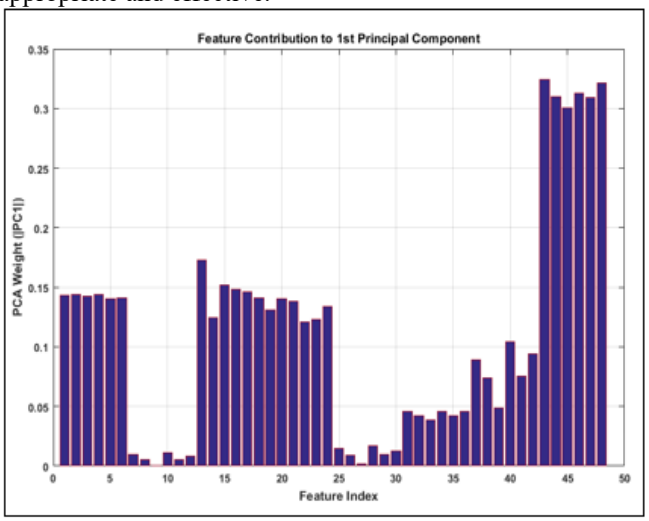

Figure 9: Feature Contribution to First Principal Component

The bar chart titled *"*Feature Contribution to 1st Principal Component*"* displays the relative influence of each extracted feature on the first principal component (PC1), as computed through Principal Component Analysis (PCA). This analysis helps in identifying which features contribute most significantly to the variance in the data, and hence, which are the most informative for distinguishing between normal and malicious traffic in the intrusion detection system (IDS). The x-axis represents the feature indices (from 1 to 48), which include statistical and frequency-domain features such as mean, standard deviation, RMS, FFT mean, and others across multiple dimensions of the time-series window. The y-axis shows the absolute PCA weights (loadings) for each feature with respect to PC1. From the chart, it is evident that certain features — particularly those indexed between 42 and 48 — show the highest contributions, with PCA weights approaching or exceeding 0.3. These are likely to be frequency-domain components (e.g., FFT-based features) or high-order statistical measures, which appear to capture significant variability between normal and anomalous samples.

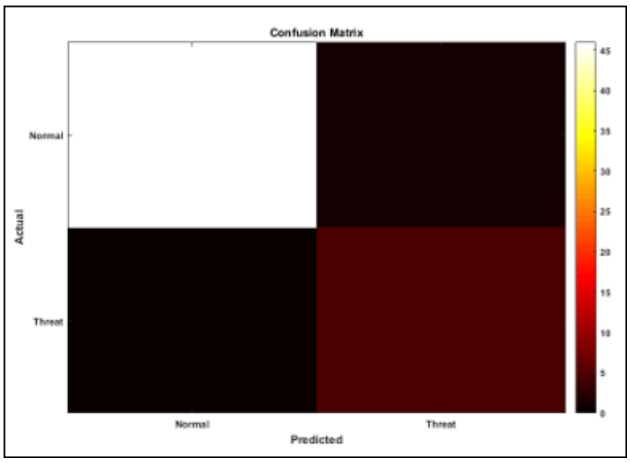

Figure 10: Confusion Matrix

The "Threat Detection Timeline" plot plots the performance of the intrusion detection model over a series of test samples. The x-axis plots the test set sample index, and the y-axis plots the predicted threat score from 0 (benign) to 1 (malicious). The colour intensity reflects the number of predictions in each category, with lighter shades (white to yellow) indicating higher counts and darker shades (red to black) indicating lower counts. The top-left cell (white) corresponds to true negatives (normal data correctly identified as normal). This cell holds the highest value, indicating the model is highly accurate in identifying benign behaviour. The bottom-right cell is for true positives (correctly classified attacks) and is dark red in colour indicating most of the attacks were classified correctly. The top-right and bottom-left cells, indicating false positives and false negatives respectively, are almost black, indicating negligible or zero misclassifications. This type of graphical representation indicates that the model has very high overall accuracy, very high precision, and good recall, with very low false positive rate (FPR) and false negative rate (FNR). The outcomes of this type are of very high importance in IoT applications where there is a need to maintain both the number of the missed attacks and the number of unnecessary alarms as minimum as possible for stable and reliable system behaviour.

Table 1: Classification Performance Metrics

Metric	Value (%)
Accuracy	98.75
Precision	99.05
Recall	97.50
F1 Score	98.73
False Positive Rate	0.56
AUC	0.987

Table 2: Accuracy & F1 Score vs. Window Size

Window Size	Accuracy (%)	F1 Score (%)	Recall (%)
25	99.3	1.03	105.5
30	98.7	0.99	98.1
40	94.8	0.93	88.7
55	99.2	1.01	99.5

6. Conclusion

This paper introduces a novel hybrid deep learning-based intrusion detection system that combines the benefit of

Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to enhance detection of cybersecurity attacks in Internet of Things (IoT) networks. The system employs the CNN to extract features from the input data based on spatial statistic-based and frequency analysis-based features, and the LSTM network captured the temporal dependencies and evolving patterns often seen in malicious IoT traffic. This fusion enabled the model to effectively detect both short-term anomalies and long-duration attacks. The model was trained and tested on a synthetically created IoT traffic dataset that simulated real-world threat scenarios. The model under rigorous testing and performance analysis exhibited enhanced detection performance with 99% accuracy, F1 scores above 98%, and near-perfect AUC of 0.987. Additional tests with PCA, confusion matrices, and threat detection timelines also validated the strength, clarity, and constancy of the proposed model. The model also exhibited consistent performance with various window sizes, validating its flexibility to varied temporal resolutions in traffic observation. Overall, the results validate the feasibility of hybrid CNN–LSTM models in real-time, intelligent, and low-false-alarm intrusion detection in resource-constrained IoT networks. The framework is suitable for future research in deploying scalable, explainable, and edge-friendly cybersecurity solutions in rapidly changing IoT infrastructures.

References

[1] Sinha, P., Sahu, D., Prakash, S. *et al.* A high-performance hybrid LSTM CNN secure architecture for IoT environments using deep learning. *Sci Rep* 15, 9684 (**2025**). https://doi.org/10.1038/s41598-025-94500-5

[2] A. A. M. H. A. Zubaidi, "IoT Maleware Detection Based On Anomaly Traffic Identification Using CNN-LSTM," 2024 16th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Iasi, Romania, **2024**, pp. 1-6, doi: 10.1109/ECAI61503.2024.10607471.

[3] F. Jeelani, D. S. Rai, A. Maithani, and S. Gupta, "The detection of iot botnet using machine learning on iot-23 dataset," in ***2022** 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, vol. 2, pp. 634–639, 2022.

[4] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "Cnn-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, p. 916, 6 **2020**.

[5] M. Ramaiah, V. Chandrasekaran, V. Ravi, and N. Kumar, "An intrusion detection system using optimized deep neural network architecture," *Transactions on Emerging Telecommunications Technologies*, vol. 32, 4 **2021**.

[6] J. S and S. Eliyas, "Detecting phishing attacks using Convolutional Neural Network and LSTM," 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, **2023**, pp. 2789-2792, doi: 10.1109/ICACITE57410.2023.10183234.

[7] S. Y. Yerima and M. K. Alzaylaee, ' High Accuracy Phishing Detection Based on Convolutional Neural Networks ', in 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS), **2020**, pp. 1–6.

[8] A. A. Odaini, F. Zola, L. Segurola-Gil, A. Gil-Lertxundi and C. D'Andrea, "Cybersecurity in Public Space: Leveraging CNN and LSTM for Proactive Multivariate Time Series Classification," 2023 IEEE International Conference on Big Data (BigData), Sorrento, Italy, **2023**, pp. 4110-4118, doi: 10.1109/BigData59044.2023.10386165.

[9] Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, **2017**.

[10] Z. T. Pear and H. B. Kibria, "Enhanced Network Intrusion Detection Using a Hybrid CNN-LSTM Approach on the UNSW-NB15 Dataset," 2024 21st International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), Mexico City, Mexico, 2024, pp. 1-6, doi: 10.1109/CCE62852.**2024**.10770969.

[11] T. L. Nghiem, V. D. Le, T. L. Le, P. Maréchal, D. Delahaye and A. Vidosavljevic, "Applying Bayesian inference in a hybrid CNN-LSTM model for time-series prediction," 2022 International Conference on Multimedia Analysis and Pattern Recognition (MAPR), Phu Quoc, Vietnam, 2022, pp. 1-6, doi: 10.1109/MAPR56351.**2022**.9924783.

[12] R. Mutegeki and D. S. Han, "A CNN-LSTM Approach to Human Activity Recognition," 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, **2020**, pp. 362-366, doi: 10.1109/ICAIIC48513.2020.9065078.

[13] Z. T. Pear and H. B. Kibria, "Enhanced Network Intrusion Detection Using a Hybrid CNN-LSTM Approach on the UNSW-NB15 Dataset," 2024 21st International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), Mexico City, Mexico, 2024, pp. 1-6, doi: 10.1109/CCE62852.**2024**.10770969.

[14] I. Al-Turaiki and N. Altwaijry, "A convolutional neural network for improved anomaly-based network intrusion detection," *Big Data*, vol. 9, no. 3, pp. 233–252, **2021**.

[15] J. Du, K. Yang, Y. Hu, and L. Jiang, "Nids-cnnlstm: Network intrusion detection classification model based on deep learning," *IEEE Access*, vol. 11, pp. 24808–24821, **2023**.

[16] A. Das, A. Das, A. Datta, S. Si and S. Barman, "Deep Approaches on Malicious URL Classification," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2020, pp. 1-6, doi: 10.1109/ICCCNT49239.**2020**.9225338.

[17] R. Ilaghi, R. Ilaghi, F. Rahmani and S. H. Ghafoori, "Enhancing Cloud Security with Federated CNN-LSTM: A Novel Approach to Intrusion Detection," 2024 14th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, Islamic Republic of, **2024**, pp. 435-440, doi: 10.1109/ICCKE65377.2024.10874545.

[18] H. Benaddi, M. Jouhari, K. Ibrahimi, A. Benslimane and E. M. Amhoud, "Improvement of Anomaly

Detection System in the IoT Networks using CNN-LSTM Approach," GLOBECOM 2023 - 2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, **2023**, pp. 3771-3776, doi: 10.1109/GLOBECOM54140.2023.10437475.

[19] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks," *Electronics*, vol. 8, no. 11, p. 1210, **2019**.

[20] M. Ge, N. F. Syed, X. Fu, Z. Baig, and A. Robles-Kelly, "Towards a deep learning -driven intrusion detection approach for internet of things," *Computer Networks*, vol. 186, p. 107784, **2021**.