ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL
JALT

AI-BASED NETWORK TRAFFIC ANOMALY AND INTRUSION DETECTION USING DEEP LEARNING AND HYBRID SPATIO-TEMPORAL MODELING

Shahzaib Hassan,[a] Jazib Arshad,[b] Nouman Ali [c] and Sohail Irshad [d]
Department of Software Engineering, Faculty of Computer Science & IT,
The Superior University, Lahore,
Pakistan
(Dated: 10 February 2026)

Abstract.
Traditional intrusion detection systems that rely on known signatures still work well for previously identified threats. However, they struggle to detect new or evolving attacks. Anomaly-based approaches attempt to solve this issue by identifying unusual patterns in network traffic, but they often generate a high number of false alarms because normal traffic behavior can vary significantly.
In this study, we propose an AI-based framework for detecting network anomalies and intrusions using a hybrid deep learning model that combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The CNN component is used to capture important patterns and relationships within traffic features, while the LSTM component analyzes how these patterns change over time. This combination makes it possible to detect complex, multi-stage attacks as well as slow, evolving threats.
We evaluate the proposed model using well-known benchmark datasets, including UNSW-NB15 and the IoT- focused TII-SSRC-23 dataset. The results show strong performance, with the model achieving 98.2% accuracy, 97.1% precision, and 98.7% recall, along with a low false alarm rate of 1.9%. Overall, the findings suggest that combining spatial and temporal deep learning techniques can significantly improve intrusion detection in modern, complex network environments.

INTRODUCTION

Traditional Intrusion Detection Systems (IDS) primarily rely on signature-based techniques, where incoming traffic is matched against a database of known attack patterns. This method is effective for identifying previously encountered threats, but it struggles with new and unknown attacks, such as zero-day exploits [1] or constantly evolving malware. Attackers can easily modify their techniques—changing payloads, communication patterns, or using obfuscation—to evade detection. At the same time, the sheer volume of modern network traffic makes manual monitoring impractical, leading to gaps in detection and delayed responses.

Artificial Intelligence (AI) and Machine Learning (ML) offer a more adaptive approach to network security. Instead of depending only on predefined signatures, these systems learn normal network behavior from historical data and use that knowledge to identify unusual activity. Once trained, they can detect deviations in traffic patterns, protocol
usage, timing, or communication flows that may signal malicious actions such as reconnaissance, data theft, or lateral movement within a network. This allows them to identify both known and previously unseen threats, even when attackers attempt to hide their presence.

Deep learning takes this a step further by uncovering complex patterns within large and high-dimensional datasets. Models like Convolutional Neural Networks (CNNs) are effective at extracting meaningful features from traffic data, while Long Short-Term Memory (LSTM) networks are well-suited for analyzing how network behavior evolves over time. When combined in hybrid models such as CNN–LSTM architectures, they can capture both spatial and temporal patterns, making them particularly useful for detecting multi-stage attacks that develop gradually. These approaches reduce the need for manual feature design and improve detection accuracy in complex environments.

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL

However, AI-based intrusion detection is not without its challenges. Training these models requires large, high- quality datasets, which are not always easy to obtain or maintain. Issues such as imbalanced data, adversarial attacks designed to mislead models, computational costs, and limited interpretability can affect real-world performance. High false positive rates can overwhelm security teams, while false negatives may result in serious breaches going undetected. For this reason, it is important to combine robust validation methods, continuous model updates, and human expertise to build reliable and effective intrusion detection systems.

a)Corresponding author: first.author@institution.edu b)Electronic mail: second.author@institution.edu c)Electronic mail: third.author@anotherinstitution.edu

Problem Statement

Modern Network Intrusion Detection Systems (NIDS) face significant challenges when deployed in today's complex and rapidly changing network environments. With increasing traffic volumes, diverse device ecosystems, and constantly evolving cyber threats, traditional detection methods are struggling to remain effective. In particular, signature-based systems have difficulty keeping up with new attack patterns, while anomaly-based approaches often produce too many false alerts, reducing their practical usefulness.

One major limitation of conventional NIDS is their inability to detect previously unseen or zero-day attacks. Signature- based techniques depend on known threat patterns, meaning they can only identify attacks that have already been analyzed and documented. However, modern attackers frequently use techniques such as obfuscation, polymorphism, and rapid mutation to modify their exploits. This creates a gap between the emergence of new threats and the development of corresponding signatures, allowing many attacks to go undetected.

On the other hand, anomaly detection methods are designed to identify unusual behavior and, in theory, can detect unknown threats. In practice, however, they often generate a high number of false positives. Normal but irregular activities-such as high-definition video streaming, large data backups, or bursts of IoT traffic-can appear suspicious simply because they differ from typical patterns. Over time, this leads to alert fatigue, where security teams are overwhelmed with warnings and may struggle to distinguish real threats from harmless anomalies.

Another critical issue is the reliance on training datasets. The performance of NIDS models depends heavily on the quality and relevance of the data used during training. Many commonly used datasets are outdated, artificially balanced, or fail to reflect modern network conditions. They often lack features such as encrypted traffic, IoT communication behavior, and cloud-based workloads. As a result, models trained on these datasets may perform well in testing but fail to generalize effectively in real-world deployments.

To address these challenges, this work adopts an AI-based intrusion detection approach trained on more recent and diverse datasets, specifically UNSW-NB15 and TII-SSRC-23. Beyond proposing a hybrid deep learning model, the study also focuses on building a complete and reproducible pipeline. This includes careful data preprocessing-such as handling missing values, encoding features, scaling, and balancing the dataset-along with consistent data splitting and evaluation using metrics that are meaningful in real deployment scenarios, particularly false alarm rates.

The core idea behind this approach is that combining spatial feature extraction through Convolutional Neural Networks (CNNs) with temporal modeling using Long Short-Term Memory (LSTM) networks can provide stronger and more reliable detection. Such a hybrid model is better suited to identifying complex, multi-stage attacks as well as slow, stealthy intrusions that may not be captured by traditional static classifiers.

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL

Role of AI in Network Security

AI has significantly improved network security by making it possible to analyze large and constantly changing data streams quickly and automatically. Unlike traditional approaches that rely on fixed rules, AI—especially Deep Learning (DL)-can work directly with raw network data and uncover patterns that are difficult to identify manually. These models learn underlying representations of traffic behavior, which helps them distinguish between normal activities, such as everyday browsing or application usage, and potentially harmful actions.

Deep learning models, particularly convolutional and recurrent networks, are well-suited for capturing both structural and temporal patterns in network traffic. They can recognize relationships across data flows, detect unusual protocol behavior, and identify subtle signs of malicious activity. This allows them to effectively detect a variety of cyber threats, including Denial-of-Service (DoS) attacks, botnet command-and-control communication, brute-force login attempts, and other attacks that may evolve or change over time.

The main contributions of this work can be summarized as follows:

- A hybrid CNN–LSTM intrusion detection model that captures both feature-level relationships and temporal patterns in network traffic.
- A consistent experimental framework for comparing traditional machine learning methods with deep learning approaches on modern benchmark datasets.
- An analysis of practical deployment challenges, with a focus on reducing false alarms and handling dataset variations across different environments.

LITERATURE REVIEW

Intrusion Detection Systems (IDS) play a central role in modern cybersecurity, acting much like a digital alarm system for networks and information systems. Their primary function is to continuously monitor and analyze network or host activity in order to detect unusual, malicious, or policy-violating behavior.

This section reviews existing research from three key perspectives that strongly influence the effectiveness of IDS in real-world settings: (i) the detection approach, including signature-based and anomaly-based methods, (ii) the type of learning model used, ranging from traditional machine learning to deep learning, and (iii) the realism and relevance of the datasets used for training and evaluation. A common observation across prior work is that strong results on benchmark datasets do not always translate into reliable performance in practice. This is largely due to changes in traffic patterns caused by factors such as encryption, emerging applications, and the growing diversity of IoT devices.

Evolution of Intrusion Detection

The development of intrusion detection systems has closely followed changes in computing environments, the increasing sophistication of cyber threats, and the growing scale and diversity of modern networks. Early IDS solutions were mainly host-based (HIDS), operating at the operating system level. They focused on analyzing audit logs, system calls, file integrity data, and other local activity to detect suspicious behavior. While these systems worked well in controlled or standalone environments, they became less effective as networks grew more complex and distributed.

As internet usage expanded and systems became more interconnected, Network Intrusion Detection Systems (NIDS) emerged to address these challenges. Unlike HIDS, NIDS analyze network traffic, including packet contents and communication patterns, to identify potential threats. Tools such as Snort, Suricata, and Zeek gained widespread adoption due to their flexibility

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL

and extensibility, and they helped establish signature-based detection as a standard approach.

However, as attackers began using more advanced evasion techniques-such as polymorphic malware, encrypted command-and-control channels, and protocol tunneling-signature-based methods started to show clear limitations. The increasing use of encryption, in particular, reduced the visibility of network traffic, making it harder for traditional inspection-based systems to detect malicious activity.

In response, research shifted toward anomaly-based and behavior-driven detection methods. These approaches aim to learn what "normal" network behavior looks like and then flag deviations that may indicate an attack. Early work in this area also introduced machine learning techniques for feature extraction and modeling, laying the foundation for more advanced, data-driven IDS solutions.

Fundamentals of IDS

Intrusion Detection Systems (IDS) are typically classified according to their core detection approach, with the foundational principles of anomaly detection first formalized by Denning [23].

Signature-Based Detection

Signature-based detection depends on a predefined database of known attack patterns, functioning much like traditional antivirus software. Since it uses deterministic, rule-driven matching, this approach is computationally efficient and performs reliably against previously documented threats. However, it struggles to detect zero-day attacks and requires constant updates to remain effective.

Anomaly-Based Detection

Anomaly-based detection systems create a baseline of what is considered "normal" behavior on a network or host. Unlike signature-based methods, they can detect previously unseen or novel activities, making them particularly effective at identifying reconnaissance efforts and lateral movement within a network. However, these systems often generate higher false positive rates.

In practical deployments, anomaly-based IDS must contend with concept drift-changes in legitimate traffic patterns caused by software updates, policy modifications, or the addition of new devices. Without strategies such as periodic retraining, drift-aware thresholds, or continual learning, these detectors can gradually produce an increasing number of alerts, even when the network is operating normally.

Machine Learning in Network Security

Machine Learning (ML) has become a key element in modern network security, enabling the automated creation of adaptive, data-driven intrusion detection models. Two primary paradigms are commonly applied:

- **Supervised Learning:** This approach requires labeled datasets, with popular algorithms including Decision Trees, Random Forests [7], and Support Vector Machines (SVM) [8].
- **Unsupervised Learning:** These methods work without labeled data, using techniques such as K-Means Clustering or Autoencoders to detect inherent structures or unusual deviations in network traffic.

The limitations of the closed-world assumption are discussed in [19]. Classical ML techniques[6] still perform well on tabular flow-based datasets, especially when careful feature selection and normalization are applied. However, they rely heavily on handcrafted or pre-extracted features, which can restrict their ability to generalize when traffic patterns evolve [24]. Additionally, traditional ML models often struggle to capture multi-stage attacks or temporal behaviors that develop across sessions. Comprehensive surveys of ML trends in intrusion detection are available in [20] and [32].

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL
JALT

Deep Learning Approaches

Deep Learning (DL)[9] has emerged as a leading approach in modern intrusion detection research because of its ability to automatically learn hierarchical feature representations from raw network data. The theoretical foundations of deep architectures are discussed in [21], [22], and [29]. Recurrent neural network–based IDS models have been explored in [28] and [34], while some of the earliest DL-based IDS frameworks are presented in [33].

Support Vector Machines (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm commonly used for classification and regression. It works by identifying an optimal hyperplane that maximizes the margin between classes in a high- dimensional feature space.

- For data that are not linearly separable, SVM employs the kernel trick (such as linear, polynomial, or RBF kernels) to map the data into a higher-dimensional space.
- The data points that lie closest to the decision boundary and influence its position are called **support vectors**.
- The main objective of SVM is to maximize the margin between classes while minimizing classification errors.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have been effectively applied to NIDS by representing network flows as matrices or temporal signals [10]. CNNs excel at detecting spatially correlated features, such as repeated byte-level patterns typical of malware or characteristic structures in protocol headers.

One key advantage of CNN-based IDS is their ability to automatically learn local feature interactions without the need for manual feature-cross engineering. This capability is especially valuable when multiple weak indicators— such as unusual combinations of packet rates, flags, and byte distributions—together reveal a strong signature of malicious activity.

Random Forest

Random Forest is an ensemble learning method used for **classification and regression** that builds multiple

decision trees and combines their outputs.

- It uses **bagging (Bootstrap Aggregating)**: each tree is trained on a random subset of data.
- At each split, a random subset of features is considered.
- Final prediction:
  - Classification -> Majority voting
  - Regression -> Average of outputs

It reduces **overfitting** and improves generalization compared to a single decision tree.

CNN (Convolutional Neural Network) Only

A **Convolutional Neural Network (CNN)** is a deep learning model primarily used for **spatial data processing**, such as images.

Key components:

- **Convolutional layers** -> Extract spatial features using filters
- **Pooling layers** -> Reduce spatial dimensions
- **Fully connected layers** -> Perform classification or regression

CNN-only means the architecture consists purely of convolutional and dense layers without recurrent components. It is mainly used for:

- Image classification

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL
JALT

- Object detection
- Image segmentation

CNN–LSTM

CNN–LSTM is a hybrid deep learning architecture that combines:

- **CNN** -> Feature extraction (spatial patterns)
- **LSTM (Long Short-Term Memory)** -> Temporal sequence modeling

LSTM is a type of Recurrent Neural Network (RNN) designed to handle long-term dependencies in sequential data.

Workflow:

1. CNN extracts spatial features.
2. Extracted features are passed to LSTM.
3. LSTM models temporal relationships. Used in:

- Video classification
- Action recognition
- Time-series with spatial structure
- Medical sequence imaging

Recurrent Neural Networks (RNN) and LSTMs

Unlike static data domains, network traffic exhibits sequential and temporal dependencies. Long Short-Term Memory (LSTM) networks are designed to model such temporal dynamics by maintaining hidden states [11]. LSTMs excel at exploiting temporal behaviors such as gradual shifts in traffic patterns or multi-stage intrusions.

For intrusion detection, the practical challenge is defining suitable sequences: flows must be grouped and ordered (e.g., by host, session, or time window). Sequence design impacts both detection latency and accuracy; shorter windows may miss slow attacks, while very long sequences may dilute signal and increase training cost.

Hybrid Models

Recent research favors hybrid architectures integrating CNNs and LSTMs [12]. In these architectures, CNN layers extract local spatial features, while LSTM layers processing the sequence of these features, enabling temporal reasoning.

Hybrid CNN–LSTM models are especially motivated in environments where attackers deliberately evade static signatures by spreading actions across time (e.g., low-rate scans, staged exploitation, and intermittent C2 beacons). In such cases, spatial correlations within a single flow are insufficient; temporal context improves separability between benign bursts and coordinated malicious behavior.

Comparative Summary of Prior Studies

Table I provides a compact comparison of representative studies and resources frequently used in IDS research. The intent is not to present a strict leaderboard (results are not directly comparable across preprocessing pipelines and label taxonomies), but to highlight how method choice, dataset selection, and evaluation scope shape reported outcomes and limitations.

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL

TABLE I. Representative prior work in IDS and related resources (performance summaries are qualitative due to non-uniform experimental protocols).

Study	Method	Dataset	Performance	Limitations
Roesch (1999) [5]	Signature NIDS	Rules ets	High detection for known attacks	
Moustafa and Slay (2015) [2]	Dataset	UNSW-NB15	Modern benchmark dataset for IDS research	
Tavallaee et al. (2009) [13]	Dataset Analysis	NSL-KDD	Improved version of KDD'99 dataset	Cannot detect zero-day attacks; limited under encrypted traffic Contains synthetic traffic; possible deployment shift
Vinayakumar et al. (2019) [10]	CNN / DNN Benchmark	IDS datasets	Strong performance on flow-based features Comprehensive overview of DL-based IDS	Dated traffic; lacks modern IoT behavior
Ferrag et al. (2020) [4]	Survey	Multiple datasets	Comprehensive overview of DL-based IDS	Sensitive to preprocessing and dataset variation. Highlights reproducibility and evaluation protocol gaps
Ullah and Mahmoud (2020) [12]	CNN–LSTM	Benchmark datasets	Demonstrates benefits of spatio-temporal learning	Higher computational cost; sequence design complexity
Shone et al. (2018)	Deep Autoencoder	NSL-KDD	Effective unsupervised feature learning	High computational overhead
Yin et al. (2017)	RNN / LSTM	NSL-KDD	Captures sequential network behaviour	Performance sensitive to hyperparameter tuning
Diro and Chilamkurti (2018)	Distributed Deep Learning	NSL-KDD	High accuracy in distributed IDS environments	Scalability and reproducibility concerns
Lashkari et al. (2017)	Traffic Characterization	ISCX / CIC datasets	Useful for encrypted traffic analysis	Limited attack diversity

Review of Datasets

The quality of an AI model is heavily dependent on the data it is trained on. This paper utilizes two

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL

distinct datasets:

1. **UNSW-NB15**: Created by the Cyber Range Lab of UNSW Canberra [2], containing hybrid traffic and 9 attack families.
2. **TII-SSRC-23**: A novel dataset introduced to address the lack of diversity in existing datasets [3], focusing on IoT environments and modern threats like Mirai botnet [14].

Other widely used datasets include CICIDS2017 [26] and DARPA-based datasets [18]. Overall, the literature suggests that results are strongly driven by dataset characteristics. Many high-accuracy results are reported on older or simplified datasets, whereas modern IoT-oriented datasets introduce traffic diversity that often increases false alarms. Dataset representativeness challenges are discussed extensively in [25]. This motivates evaluating the CNN-LSTM model on both UNSW-NB15 (enterprise-style benchmark) and TII-SSRC-23 (IoT-focused modern traffic) to better reflect realistic variability.

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL

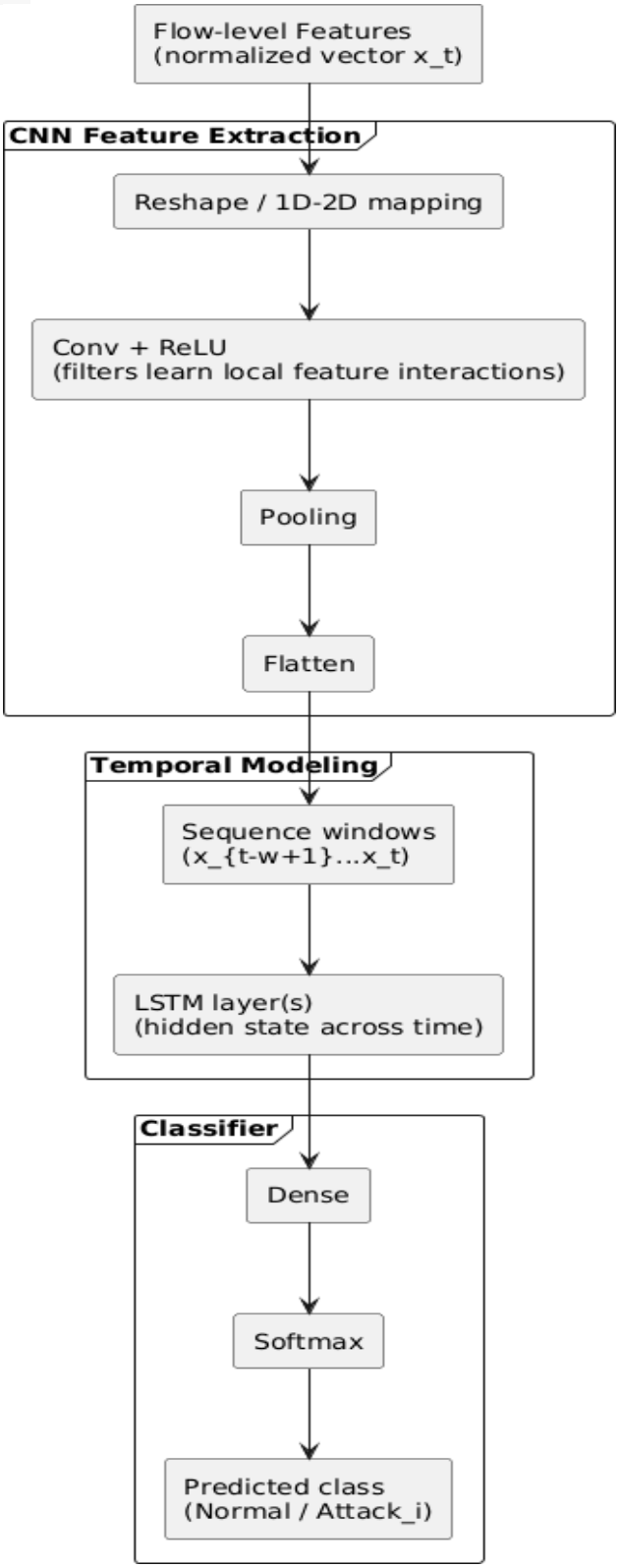


FIGURE 1. Hybrid CNN–LSTM architecture used for intrusion detection.

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL
JALT

METHODOLOGY

Dataset Description

To ensure a comprehensive and reliable evaluation of the CNN-LSTM intrusion detection models, this project utilizes UNSW- NB15 and TII-SSRC-23.

UNSW-NB15: Features 49 attributes spanning numeric, categorical, binary, and nominal types. It includes a multi- class categorization of Normal plus 9 attack types.

TII-SSRC-23: Focuses on traffic diversity, including IoT-specific protocols. It aligns with contemporary network environments, capturing traffic types often missed by older datasets [13].

To improve comparability and mitigate sampling bias, experiments assume a fixed train/validation/test split with stratification by class. For the temporal model, samples are additionally grouped into short sequences (sliding windows) to preserve ordering, ensuring that the LSTM receives meaningful context rather than randomly shuffled flows.

Data Preprocessing

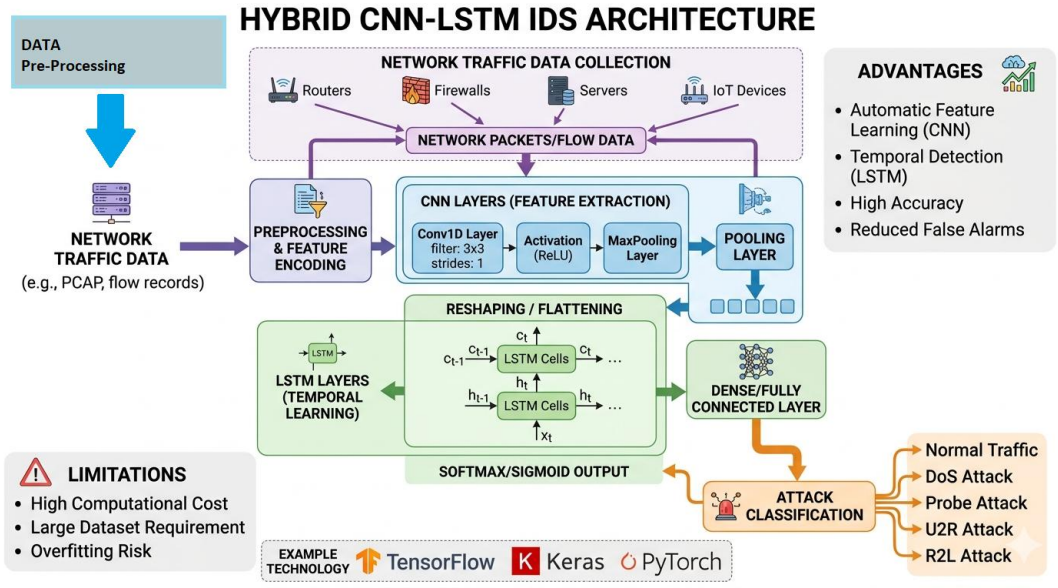
Preprocessing ensures raw traffic features are suitable for deep learning.

- **Data Cleaning**: Handling missing values through imputation and removing duplicate records.
- **Feature Selection**: Using Feature Importance Analysis [15] to eliminate irrelevant features.
- **Encoding**: One-Hot Encoding for categorical features and Label Encoding for binary flags.
- **Normalization**: Min-Max scaling is applied:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- **Class Balancing**: Applying SMOTE [16], [17] to address the high imbalance between benign and malicious traffic.

For CNN processing, normalized feature vectors are reshaped into a 2D representation (or treated as 1D sequences) to allow convolutional filters to learn local interactions among adjacent features. For LSTM processing, flows are ordered by time and grouped into fixed-length windows. In practice, the window length is chosen to balance detection latency and the ability to capture slow-evolving attacks.



ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL
JALT

FIGURE 2. Hybrid CNN–LSTM methodology used for intrusion detection.

Hybrid Architecture

The core contribution of this work is a hybrid spatio-temporal deep learning model.

1. **CNN Feature Extraction**: CNN layers capture spatial relationships between correlated flow attributes. The convolution operation is defined as $h = f(W * X + b)$.
2. **LSTM Temporal Modeling**: LSTM layers model sequential dependencies across time-ordered flows.
3. **Classification Layer**: The final dense layer applies Softmax for multi-class classification:

$$P(y - i) = \frac{e^{zi}}{\sum_{j=1}^{n} e^{zj}}$$

Regularization is applied using dropout and early stopping to reduce overfitting, especially when training on imbalanced classes. Model selection is based on validation-set F1-score and FAR to discourage overly aggressive detection thresholds that would be impractical in real SOC workflows.

SYSTEM ARCHITECTURE AND DESIGN

This section describes how the model fits into an operational intrusion detection pipeline. While model accuracy is important, practical deployment requires reliable telemetry collection, deterministic preprocessing, and alerting workflows that avoid analyst overload.

End-to-End Pipeline

Figure 2 illustrates an end-to-end architecture organized into four layers: (i) network environment and sensors, (ii) flow collection and feature extraction, (iii) preprocessing and inference services, and (iv) operations and response. Traffic is exported as flow records (NetFlow/IPFIX or equivalent). Features are computed consistently during both training and inference to prevent training/serving skew.

Preprocessing and Windowing

Deep models are sensitive to preprocessing choices. Categorical attributes (e.g., protocol, service, connection state) are encoded using one-hot or label encoding. Continuous fields (e.g., duration, bytes, packet counts) are normalized using min–max scaling to prevent high-magnitude features from dominating gradient updates.

Temporal modeling requires constructing sequences from individual flow records. In this work, flows are grouped by entity (e.g., source host or bidirectional session key) and ordered by timestamp. A fixed-length sliding window then converts streams into sequences for the LSTM. Window size is a trade-off: smaller windows yield lower detection latency, while larger windows can capture multi-stage behavior but increase compute and potentially dilute signal.

Online Detection Workflow

In production, inference runs continuously on streamed telemetry. Figure presents a sequence-level view of online detection: sensors export flows, preprocessing constructs windows, the CNN–LSTM produces a score or class distribution, and the decision engine applies thresholds and policy. Alerts are stored with context to support triage and retrospective analysis.

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL
JALT

FEATURE ENGINEERING AND SEQUENCE CONSTRUCTION

Although deep learning reduces reliance on manual feature crafting, feature quality remains a primary driver of IDS performance. Flow-level datasets typically include basic transport statistics (bytes, packets, duration), header and flag information, and derived rates (bytes/sec, packets/sec). Time-based statistical features have also been shown effective for analyzing encrypted traffic such as Tor flows [27]. For IoT settings, periodic telemetry traffic and bursty device synchronization can resemble scanning or flooding if rates are interpreted without context; temporal aggregation and window-based features help disambiguate these cases.

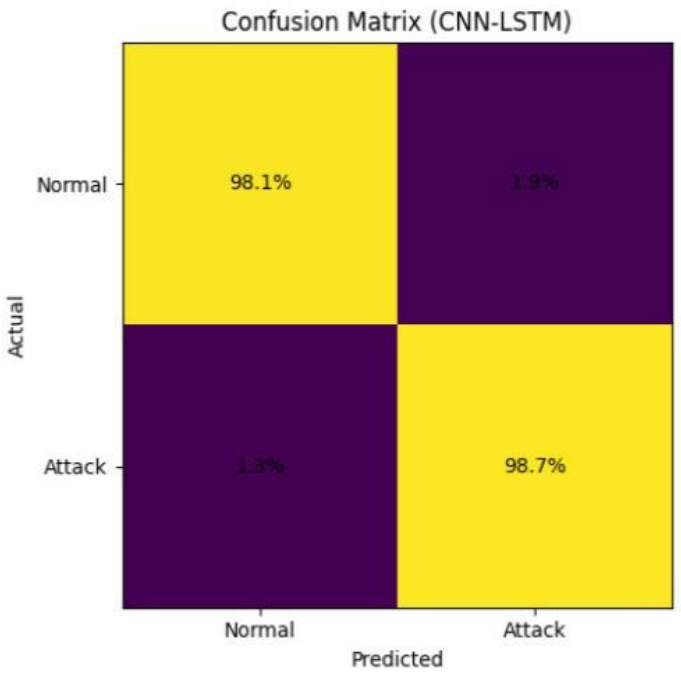
To create sequences, each flow record is mapped to an ordered event stream. Let $xt \in Rd$ denote the feature vector at time step t. A window of length T yields a tensor $X \in RT \times d$ for the LSTM. In practice, T is selected empirically based on validation FAR and detection latency constraints.

In addition to raw features, lightweight derived indicators are often useful: connection fan-out (unique destinations per host per window), failure ratios (failed connections/attempts), and burstiness measures (variance of inter-arrival times). These are compatible with both enterprise and IoT settings and can reduce false alarms caused by benign high-throughput transfers.

EXPERIMENTAL SETUP

Training is performed in Python with TensorFlow on GPU hardware. Key hyperparameters:

- Optimizer: Adam
- Learning rate: 0.001
- Batch size: 128
- Epochs: 30
- Dropout: 0.3



ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL

FIGURE 3. Confusion matrix for the CNN–LSTM intrusion detection model.

Baselines and Evaluation Protocol

To contextualize hybrid model performance, experiments compare against classical ML baselines commonly used for tabular flow data: SVM and Random Forest. All models use the same train/validation/test partitions and identical preprocessing to ensure comparisons reflect modeling capability rather than pipeline differences.

Evaluation focuses on IDS-relevant metrics. ROC and Precision-Recall trade-offs are formally analyzed in [30], [31]. Accuracy and F1-score summarize overall discrimination, while FAR is emphasized due to its operational impact. FAR is computed as

$$FAR = \frac{FP}{FP + TN}$$

Practical Deployment Considerations

Beyond predictive quality, deployments must consider throughput, latency, and maintainability. Flow export and feature extraction add measurement overhead; inference adds compute overhead. A common operational compromise is to perform lightweight inference on edge collectors (for fast response) while preserving richer context centrally for retrospective analysis.

Dataset shift is another key challenge. Real networks evolve due to new services, software updates, policy changes, and onboarding of new IoT devices. Periodic retraining or drift-aware calibration is required to keep FAR bounded over time.

Performance is evaluated using Accuracy, Precision, Recall, F1-score, and False Alarm Rate (FAR).

In addition to aggregate metrics, experiments consider per-class behavior (where multi-class labels are available) to understand whether rare but high-impact attacks are missed. Operationally, FAR is emphasized because even small increases in false alarms can overwhelm analysts at enterprise traffic volumes.

RESULTS AND DISCUSSION

Performance Metrics

Table II summarizes the intrusion detection results compared to baseline models.

TABLE II. Final intrusion detection performance comparison.

Model	Accuracy (%)	Precision (%)	Recall (%)	FAR (%)
SVM	90.4	88.7	91.2	6.8
Random Forest	94.6	93.5	95.1	4.2
CNN Only	96.3	95.7	96.8	2.9
CNN-LSTM	98.2	97.1	98.7	1.9

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL
JALT

The hybrid CNN-LSTM achieves the strongest recall, demonstrating superior detection of attacks with minimal missed intrusions. The low FAR highlights improved operational practicality.

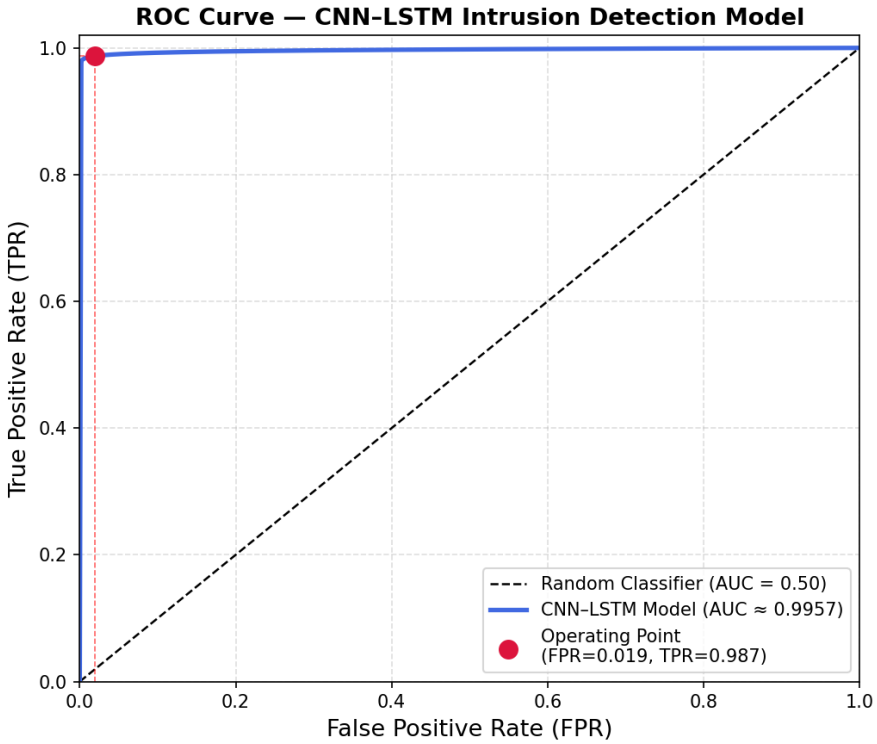


FIGURE 4. Accuracy and false alarm rate comparison across baseline models and the CNN-LSTM model.

Analysis

The Random Forest model performed well on structured features but struggled with complex temporal patterns. The CNN model showed improvement by capturing spatial feature correlations. However, the Hybrid CNN-LSTM out- performed all baselines, particularly in detecting multi-stage attacks present in the TII-SSRC-23 dataset.

These results are consistent with the intuition that temporal context helps disambiguate short benign bursts (e.g., IoT telemetry spikes) from coordinated malicious patterns (e.g., botnet scanning followed by exploitation). While SVM and Random Forest can be tuned for higher recall, doing so typically increases FAR; the hybrid model maintains higher recall with a comparatively lower FAR, suggesting better separation in the learned representation.

Ablation and Sensitivity Discussion

To understand which design choices, contribute most to performance, it is useful to analyze ablations. Removing temporal modeling (CNN-only) typically increases false alarms for bursty but benign traffic because time context is unavailable. Conversely, using sequence-only modeling (LSTM without convolutional feature extraction) can reduce performance when individual time steps contain weak signals that require learning local feature interactions.

Window length T is also a practical sensitivity factor. Very short windows reduce context and can miss slow attacks; very long windows increase compute and may blend benign and malicious

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL

periods. In operational settings, selecting *T* based on FAR constraints and acceptable detection latency is often more meaningful than optimizing accuracy alone.

Operational Tuning and Error Analysis

From an operational perspective, the costliest failure mode is a sustained rise in false positives. In practice, FAR is influenced by (i) threshold choice, (ii) the stability of preprocessing (consistent encoding and scaling), and (iii) the stationarity of the underlying environment. Even with a strong model, introducing a new application (e.g., video conferencing) or onboarding a new IoT device class can shift traffic distributions and cause transient alert spikes.

False positives frequently emerge from "benign anomalies" such as scheduled backups, firmware updates, or telemetry bursts. Temporal modeling helps because these activities often exhibit repetitive, periodic patterns that differ from multi- stage attacks. However, if sequences are constructed without preserving entity boundaries (e.g., mixing flows from multiple hosts into the same window), the temporal signal can be contaminated and misclassified. For false negatives, slow-and-low attacks remain challenging when their observable footprint is intentionally weak per time step. Practical mitigations include increasing the window length for high-risk assets, enriching features with
lightweight graph indicators (fan-out, new-peer rate), and incorporating analyst feedback loops for retraining.

Finally, thresholds should be calibrated to the deployment objective. A SOC may prefer a conservative operating point (lower FAR) during high-volume periods, and a more aggressive point (higher recall) during incident response. This motivates maintaining a calibration procedure rather than relying on a single global threshold.

THREATS TO VALIDITY AND LIMITATIONS

Several threats to validity should be taken into account. First, benchmark datasets may not fully reflect the diversity of traffic present in real-world enterprise or IoT environments, including proprietary protocols and organization-specific policies. Second, label quality is not always perfect; mislabeled or ambiguous flows can artificially inflate false alarm rates or reduce recall. Third, preprocessing decisions—such as encoding schemes and data balancing—can influence class boundaries, so results should always be interpreted within the context of the specific pipeline used.

Additionally, deep learning models introduce operational considerations. They often require greater computational resources, need ongoing monitoring for concept drift, and can be vulnerable to adversarial manipulation. Mitigation strategies, such as calibrated thresholds, rate limiting, and continuous model monitoring, should be incorporated into any deployment plan.

ETHICAL, PRIVACY, AND REPRODUCIBILITY CONSIDERATIONS

Network telemetry often contains sensitive information, including user behavior patterns, device identifiers, and timing data that could potentially be correlated. When deploying learning-based IDS, organizations should establish clear policies for data minimization and retention. Wherever possible, analysis should focus on flow-level features rather than raw payloads to limit exposure of sensitive content.

Model outputs can also affect human decision-making, so alert interfaces should be designed to support verification rather than encourage automation bias. Providing additional context—such as the top contributing features, recent sequence summaries, or comparisons with peer activity—can help analysts validate alerts and avoid unnecessary escalation.

For reproducibility, it is essential to persist and version preprocessing artifacts (e.g., encoders,

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL

scalers, feature order), random seeds, and dataset splits. Without these practices, results may vary between runs, and comparisons between baseline models and deep learning approaches can become unreliable.

CONCLUSION AND FUTURE WORK

Convolutional Neural Networks (CNNs) have been successfully applied to NIDS by treating network flows as matrices or temporal signals [10]. They are particularly effective at capturing spatially correlated features, such as repeated byte-level patterns common in malware or distinctive structures within protocol headers.

A major strength of CNN-based IDS lies in their ability to automatically learn local feature interactions without relying on manual feature-cross engineering. This is especially useful when multiple weak indicators—like unusual combinations of
packet rates, flags, and byte distributions—combine to form a strong signature of malicious activity.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Department of Software Engineering, The Superior University, Lahore.

REFERENCES

1. L. Bilge and T. Dumitras, “Zero-day attack: A survey,” ACM Computing Surveys (CSUR) **2**, 1–30 (2012).
2. N. Moustafa and J. Slay, “Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set),” Military Communications and Information Systems Conference (MilCIS) (2015).
3. F. Al-Ali and et al., “Tii-ssrc-23: A dataset for intrusion detection in iot networks,” IEEE Internet of Things Journal (2023).
4. M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and
comparative study,” Journal of Information Security and Applications **50**, 102419 (2020).
5. M. Roesch, “Snort: Lightweight intrusion detection for networks,” LISA **99**, 229–238 (1999).
6. Nasim, Fawad, Sohail Masood, Arfan Jaffar, Usman Ahmad, and Muhammad Rashid. "Intelligent Sound-Based Early Fault Detection System for Vehicles." Computer Systems Science & Engineering 46, no. 3 (2023).
7. Ramzan, Muhammad Shaharyar, Fawad Nasim, Hafiz Nabeel Ahmed, Umar Farooq, Muhammad Sheraz Nawaz, Syed Krar Haider Bukhari, and Hamayun Khan. "An Innovative Machine Learning based end-to-end Data Security Framework in Emerging Cloud Computing Databases and Integrated Paradigms: Analysis on Taxonomy, challenges, and Opportunities." Spectrum of engineering sciences (2025): 90-125.
8. Aish, Muhammad Abdullah, Amina Abdul Ghafoor, Fawad Nasim, Kiran Irfan Ali, Shamim Akhter, and Sumbul Azeem. "Improving stroke prediction accuracy through machine learning and synthetic minority over-sampling." Journal of Computing & Biomedical Informatics 7, no. 02 (2024).
9. JTaj, F., Muqaddas, R., Yousaf, A., Nasim, M. F., Naeem, M., & Nawaz, E. (2025, October). Ensembled Deep Learning (DL) Methods for Detecting Skin Cancer Using CNN Algorithm with Multi Model. In *2025 International Conference on Engineering and Emerging Technologies (ICEET)* (pp. 1-6). IEEE.
10. R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and J. Ryoo, “A deep learning approach for network intrusion detection
system,” Future Generation Computer Systems **95**, 6–15 (2019).
11. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation **9**, 1735–1780 (1997).
12. I. Ullah and Q. H. Mahmoud, “A hybrid dl-driven intelligent network intrusion detection system for complex attacks,” Computer Networks **1**, 1– 10 (2020).
13. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” IEEE Symposium on Computational Intelligence for Security and Defense Applications (2009).

ISSN E: 2709-8273
ISSN P:2709-8265
JOURNAL OF APPLIED LINGUISTICS AND TESOL
JALT

14. M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," USENIX security symposium (2017).
15. G. Chandrashekar and F. Sahin, "Feature selection for intrusion detection systems," Computers & Electrical Engineering **40**, 16–28 (2014).
16. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," Journal of artificial intelligence research **16**, 321–357 (2002).
17. H. He and E. A. Garcia, "Learning from imbalanced data," IEEE Transactions on Knowledge and Data Engineering **21**, 1263–1284 (2009).
18. K. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," *DARPA Information Survivability Conference and Exposition (DISCEX)* (1999).
19. R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," *IEEE Symposium on Security and Privacy* (2010).
20. A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials* 18, 1153–1176 (2016).
21. Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning* 2, 1–127 (2009).
22. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).
23. D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering* 13, 222–232 (1987).
24. W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Transactions on Information and System Security (TISSEC)* 3, 227–261 (2000).
25. M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security* 86, 147–167 (2019).
26. M. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSP* (CICIDS2017 dataset) (2018).
27. A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time-based features," *ICISSP* (2017).
28. J. Kim, J. Kim, H. L. Kim, and H. K. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," *International Conference on Platform Technology and Service (PlatCon)* (2016).
29. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature* 521, 436–444 (2015).
30. T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters* 27, 861–874 (2006).
31. J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," *ICML* (2006).
32. R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications* 9, 16 (2018).
33. A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," *EAI International Conference on Bio-inspired Information and Communications Technologies* (2016).
34. H. Yin, Z. Qin, G. Wang, X. Luo, and J. Wang, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access* 5, 21954–21961 (2017).