International Journal of Engineering, Business and Management (IJEBM)
ISSN: 2456-7817
[Vol-10, Issue-2, Apr-Jun, 2026]
Issue DOI: https://dx.doi.org/10.22161/ijebm.10.2
Article Issue DOI: https://dx.doi.org/10.22161/ijebm.10.2.4

Intelligent Threat Detection in Cloud Computing Using Machine Learning Techniques: A Hybrid Intrusion Detection Framework

Rajesh Rajaan[1], Loveleen Kumar[1], Nilam Choudhary[3], Aakriti Sharma[4]

[1]Assistant Professor, Computer Science and Engineering, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur, Rajasthan, India
Rajesh.rajaan@skit.ac.in; ORCID: 0000-0002-0931-6605
[2]Assistant Professor, Computer Science and Engineering, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur, Rajasthan, India
loveleentak@gmail.com; ORCID: 0000-0002-6532-4220
[3]Associate Professor, Department of Computer Science & Engineering, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur, Rajasthan, India
nilam@skit.ac.in; ORCID: 0000-0003-4728-2511
[4]Associate Professor, Department of Computer Science & Engineering, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur, Rajasthan, India
aakriti@skit.ac.in; ORCID: 0000-0003-1218-0136

Received: 14 Apr 2026; Received in revised form: 11 May 2026; Accepted: 16 May 2026; Available online: 20 May 2026

Abstract— *Cloud computing has fundamentally transformed how organizations deploy, scale, and manage computational resources. However, this paradigm shift introduces significant security vulnerabilities arising from multi-tenancy, distributed architectures, and dynamic traffic patterns. Traditional intrusion detection systems struggle to address these challenges due to their reliance on static signature-based methods and inability to adapt to novel attack vectors. This paper presents a hybrid intrusion detection framework that integrates machine learning and deep learning techniques to enhance threat detection accuracy in cloud environments. The proposed framework combines Extreme Gradient Boosting (XGBoost) for feature selection with a Convolutional Neural Network–Long Short-Term Memory (CNN-LSTM) architecture for classification. We evaluate the framework using four benchmark datasets: NSL-KDD, CICIDS2017, CSE-CIC-IDS2018, and UNSW-NB15. Experimental results demonstrate that the hybrid approach achieves detection accuracy exceeding 98.5%, with a false positive rate below 1.2% across all datasets. The framework exhibits strong generalization capabilities and maintains real-time detection latency suitable for production cloud deployments. These findings suggest that combining spatial feature extraction with temporal sequence modeling provides a robust foundation for next-generation cloud security systems.*

Keywords— ***Intrusion Detection System, Cloud Security, Machine Learning, Deep Learning, CNN-LSTM, XGBoost, Hybrid Framework, Network Security***

I. INTRODUCTION

1.1 Background and Motivation

The proliferation of cloud computing has revolutionized information technology infrastructure across virtually every sector of the global economy. Organizations increasingly migrate critical workloads to cloud platforms to leverage benefits including elastic scalability, reduced capital expenditure, and improved availability. According to recent industry analyses, global cloud infrastructure spending continues to grow at double-digit annual rates, with the majority of enterprise workloads now residing in cloud environments.

This widespread adoption, however, has created an expanded attack surface that malicious actors actively

exploit. Cloud environments present unique security challenges that traditional perimeter-based defenses cannot adequately address. Multi-tenancy architectures, where multiple customers share underlying physical infrastructure, create potential vectors for lateral movement and data leakage. The dynamic nature of cloud workloads, with virtual machines and containers spinning up and down in response to demand, complicates the establishment of behavioral baselines. Additionally, the virtualized network fabric that connects cloud resources often obscures traffic patterns from conventional network monitoring tools.

Intrusion detection systems represent a critical component of defense-in-depth security strategies. These systems monitor network traffic and system activities to identify suspicious behavior indicative of unauthorized access, policy violations, or active attacks. Traditional intrusion detection approaches fall into two primary categories: signature-based detection, which matches observed patterns against databases of known attack signatures, and anomaly-based detection, which identifies deviations from established normal behavior profiles.

Signature-based systems offer high accuracy for known threats but fail entirely against novel attacks or variants that do not match existing signatures. Anomaly-based systems can theoretically detect previously unseen attacks but historically suffer from high false positive rates that overwhelm security operations teams and erode trust in automated alerts. Neither approach adequately addresses the scale, velocity, and complexity of modern cloud environments.

1.2 Research Objectives

This research aims to develop and evaluate a hybrid intrusion detection framework specifically designed for cloud computing environments. The primary objectives are:

To design a feature selection mechanism that identifies the most discriminative attributes from high-dimensional network traffic data while reducing computational overhead.

To develop a hybrid deep learning architecture that captures both spatial patterns within individual network flows and temporal dependencies across sequences of network events.

To evaluate the proposed framework against multiple benchmark datasets representing diverse attack scenarios and traffic characteristics.

To assess the framework's suitability for real-time deployment through analysis of detection latency and computational resource requirements.

1.3 Contributions

This paper makes the following contributions to the field of cloud security and intrusion detection:

A comprehensive hybrid framework that integrates gradient boosting for feature optimization with deep neural networks for classification, specifically tailored to the requirements of cloud environments.

An extensive empirical evaluation demonstrating state-of-the-art detection performance across four widely-used benchmark datasets.

Analysis of the framework's generalization capabilities and resistance to dataset-specific overfitting.

Practical insights into deployment considerations including computational efficiency and scalability characteristics.

1.4 Paper Organization

The remainder of this paper is organized as follows. Section 2 reviews related work in intrusion detection systems, machine learning approaches to network security, and cloud-specific security challenges. Section 3 describes the proposed methodology, including the system architecture, feature selection process, and deep learning model design. Section 4 details the experimental setup, datasets, and evaluation metrics. Section 5 presents and analyzes experimental results. Section 6 discusses findings, limitations, and practical implications. Section 7 concludes the paper and identifies directions for future research.

II. LITERATURE REVIEW

2.1 Traditional Intrusion Detection Approaches

Intrusion detection systems have evolved significantly since their conceptual introduction in the 1980s. Early systems relied primarily on rule-based expert systems that encoded human knowledge about attack patterns into deterministic decision logic. While interpretable and predictable, these systems required extensive manual maintenance and could not adapt to evolving threats without explicit rule updates.

Signature-based detection emerged as the dominant paradigm through the 1990s and 2000s, with systems like Snort becoming industry standards. These systems maintain databases of known attack signatures—specific patterns of bytes, packet structures, or protocol violations that indicate malicious activity. When network traffic matches a signature, the system generates an alert. The primary advantage lies in the low false positive rate for known attacks and the interpretability of alerts. However, signature-based systems are fundamentally reactive, requiring signature updates after new attacks are discovered and analyzed. They also struggle with polymorphic attacks

that deliberately vary their observable characteristics to evade detection.

Anomaly-based detection attempts to overcome the limitations of signature-based approaches by establishing models of normal behavior and flagging deviations. Statistical methods, including threshold-based detection, clustering algorithms, and probability distribution modeling, formed the basis of early anomaly detection systems. These approaches can theoretically identify novel attacks without prior knowledge of their signatures. In practice, however, defining "normal" behavior in complex, dynamic environments proves extremely challenging. The resulting high false positive rates—often exceeding 20-30% in production deployments—limit the practical utility of many anomaly-based systems.

2.2 Machine Learning in Intrusion Detection

The application of machine learning to intrusion detection has attracted substantial research attention over the past two decades. Machine learning algorithms can automatically learn patterns from labeled training data, potentially capturing complex relationships that human analysts or manually-crafted rules might miss.

Supervised learning algorithms, including decision trees, random forests, support vector machines, and naive Bayes classifiers, have been extensively studied for intrusion detection. These algorithms require labeled datasets containing examples of both normal traffic and various attack types. Decision trees offer interpretability but may overfit to training data. Random forests address overfitting through ensemble methods, aggregating predictions from multiple trees trained on bootstrapped samples. Support vector machines excel at finding optimal decision boundaries in high-dimensional feature spaces but scale poorly to very large datasets.

Ensemble methods have demonstrated particular promise for intrusion detection. Gradient boosting algorithms, including XGBoost, LightGBM, and CatBoost, iteratively build ensembles of weak learners (typically decision trees) with each subsequent learner focusing on examples that previous learners misclassified. XGBoost has achieved state-of-the-art results on numerous intrusion detection benchmarks, combining strong predictive performance with reasonable computational efficiency.

Unsupervised and semi-supervised learning approaches address scenarios where labeled attack data is scarce or unavailable. Clustering algorithms can group similar traffic patterns, potentially isolating anomalous clusters that warrant investigation. Autoencoders learn compressed representations of normal traffic and identify anomalies based on reconstruction error—inputs that differ significantly from the training distribution produce higher reconstruction errors.

2.3 Deep Learning Approaches

Deep learning represents a subset of machine learning characterized by neural network architectures with multiple hidden layers capable of learning hierarchical feature representations. Unlike traditional machine learning, which often requires manual feature engineering, deep learning models can automatically extract relevant features from raw or minimally processed input data.

Convolutional Neural Networks, originally developed for image recognition, have been adapted for intrusion detection by treating network flow features as one-dimensional or two-dimensional arrays. Convolutional layers apply learnable filters across the input, capturing local patterns and spatial relationships. Pooling layers reduce dimensionality while preserving important features. CNN-based intrusion detection systems have demonstrated strong performance, particularly when network data is structured to preserve meaningful spatial relationships between features.

Recurrent Neural Networks address sequential data by maintaining internal state that persists across time steps. Long Short-Term Memory networks extend basic RNNs with gating mechanisms that selectively retain or forget information over long sequences, addressing the vanishing gradient problem that limits standard RNNs. Network traffic naturally exhibits temporal structure—attacks often unfold over multiple packets or connections, and understanding this temporal context improves detection accuracy. LSTM-based systems have shown promise for detecting slow, stealthy attacks that manifest over extended time periods.

Hybrid architectures combining CNNs and LSTMs leverage the complementary strengths of both approaches. CNNs extract spatial features from individual network flows or packets, while LSTMs model temporal dependencies across sequences of flows. This combination enables detection of attacks that exhibit both distinctive per-flow characteristics and characteristic temporal patterns. journalofcloudcomputing.springeropen.com

Attention mechanisms, originally developed for natural language processing, have been incorporated into intrusion detection systems to enable models to focus on the most relevant portions of input data. Self-attention allows models to weigh the importance of different features or time steps dynamically, improving both accuracy and interpretability.

2.4 Cloud-Specific Security Challenges

Cloud computing introduces security considerations that differ qualitatively from traditional on-premises

environments. Understanding these challenges is essential for designing effective cloud intrusion detection systems.

Multi-tenancy represents perhaps the most fundamental cloud security concern. Multiple customers share physical infrastructure including servers, storage, and network equipment, with isolation provided through virtualization and software-defined boundaries. While hypervisors and container runtimes provide strong isolation in normal operation, vulnerabilities in these components can enable attacks to escape isolation boundaries. Additionally, shared resources create potential side-channel attacks where malicious tenants extract information about co-located workloads through timing analysis, cache behavior, or other indirect observations.

Virtualized networking complicates traffic monitoring in cloud environments. Traditional network intrusion detection systems monitor traffic at physical network taps or span ports. In cloud environments, much traffic flows entirely within virtual networks, never traversing physical infrastructure where conventional monitoring occurs. Software-defined networking provides flexibility but requires intrusion detection systems to integrate with virtualization platforms to maintain visibility.

Dynamic workloads challenge behavioral baselines that anomaly detection systems rely upon. Cloud workloads scale automatically in response to demand, with virtual machines and containers created and destroyed continuously. IP addresses are dynamically assigned and may be reused by different workloads. Establishing what constitutes "normal" behavior in such fluid environments requires detection systems that adapt continuously rather than relying on static profiles.

Encrypted traffic increasingly comprises the majority of cloud network traffic. While encryption protects data confidentiality, it also blinds content-based inspection techniques. Intrusion detection systems must rely on metadata, flow characteristics, and behavioral patterns rather than payload inspection for encrypted traffic.

2.5 Research Gaps

- Despite significant progress, several gaps remain in the current state of cloud intrusion detection research:
- Many studies evaluate proposed methods on a single benchmark dataset, raising questions about generalization to other environments or attack types.
- Few studies adequately address the computational requirements for real-time detection in high-throughput cloud environments.
- Feature selection approaches are often applied independently from classification, potentially missing opportunities for end-to-end optimization.
- Limited research addresses the specific challenges of multi-tenant cloud environments, including the need to detect attacks that exploit shared infrastructure.
- This research addresses these gaps through a comprehensive hybrid framework evaluated across multiple datasets with explicit consideration of computational efficiency and cloud deployment requirements.

III. PROPOSED METHODOLOGY

3.1 System Architecture Overview

The proposed Intelligent Threat Detection Framework (ITDF) follows a modular architecture comprising four primary stages: data preprocessing, feature selection, hybrid classification, and alert generation. This architecture enables flexibility in deployment while maintaining clear separation of concerns between components.

ITDF=f(Preprocessing)→g(FeatureSelection)→h(Classification)→OutputITDF=f(Preprocessing)→g(Feature Selection)→h(Classification)→Output

The data flow proceeds as follows. Raw network traffic data enters the preprocessing module, which performs normalization, encoding, and cleaning operations to prepare data for subsequent analysis. The feature selection module applies XGBoost-based importance ranking to identify the most discriminative features, reducing dimensionality while preserving predictive information. The selected features feed into the hybrid CNN-LSTM classification module, which outputs probability scores for each traffic class. Finally, the alert generation module applies configurable thresholds and policies to produce actionable security alerts.

3.2 Data Preprocessing

Effective preprocessing is essential for machine learning and deep learning models to achieve optimal performance. The preprocessing pipeline addresses several challenges inherent in network traffic data.

Missing Value Handling: Network traffic datasets frequently contain missing values due to collection errors, incomplete flows, or protocol variations. The framework employs a two-stage approach: features with greater than 50% missing values are removed entirely, while remaining missing values are imputed using median values for numeric features and mode values for categorical features. Median imputation provides robustness against outliers that mean imputation lacks.

Categorical Encoding: Network traffic features include categorical variables such as protocol type, service identifiers, and connection flags. These categorical features require numeric encoding for neural network processing. The framework applies one-hot encoding for low-cardinality categorical features (fewer than 10 unique values) and target encoding for higher-cardinality features. Target encoding replaces each category with the mean of the target variable for that category, providing a smooth numeric representation that preserves class-discriminative information.

Normalization: Neural networks train more effectively when input features share similar scales. The framework applies standardization (z-score normalization) to all numeric features:

xnormalized=x−μσxnormalized=σx−μ

where μ represents the feature mean and σ represents the feature standard deviation, computed from training data only to prevent data leakage.

Class Imbalance Handling: Intrusion detection datasets typically exhibit severe class imbalance, with normal traffic vastly outnumbering attack traffic, and common attacks outnumbering rare attack types. The framework addresses imbalance through a combination of techniques: Synthetic Minority Over-sampling Technique (SMOTE) generates synthetic examples of minority classes, while class weights in the loss function increase the penalty for misclassifying minority class examples.

3.3 Feature Selection Using XGBoost

High-dimensional feature spaces present challenges for machine learning models, including increased computational requirements, potential overfitting, and the curse of dimensionality. Effective feature selection identifies the most informative features while discarding redundant or noisy features.

The framework employs XGBoost for feature importance ranking due to its strong performance on tabular data and built-in feature importance calculation. XGBoost is a gradient boosting algorithm that builds an ensemble of decision trees sequentially, with each tree trained to correct the errors of the ensemble built so far.

The optimization objective for XGBoost combines a loss function measuring prediction error with a regularization term controlling model complexity:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

where l is a differentiable loss function, y_i is the true label, $\hat{y}_i$ is the predicted label, K is the number of trees, and Ω is the regularization term defined as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

where T is the number of leaves in the tree, w_j is the weight of leaf j, and γ and λ are regularization parameters.

Feature importance scores are computed based on the contribution of each feature to the ensemble's predictive performance. The framework uses the "gain" importance metric, which measures the average improvement in the objective function when a feature is used for splitting:

$$\text{Importance}(f) = \sum_{\text{splits on } f} \text{Gain}$$

Features are ranked by importance, and the top-k features are selected for subsequent processing. The value of k is determined through cross-validation, selecting the minimum number of features that achieves within 1% of the maximum achievable performance.

3.4 Hybrid CNN-LSTM Classification Model

The classification module employs a hybrid architecture combining Convolutional Neural Networks for spatial feature extraction with Long Short-Term Memory networks for temporal sequence modeling. This combination captures both the characteristics of individual network flows and the temporal patterns that emerge across sequences of flows.

3.4.1 CNN Component

The CNN component processes the selected features to extract hierarchical spatial representations. The architecture consists of multiple convolutional blocks, each containing a convolutional layer, batch normalization, activation function, and pooling layer.

For an input feature vector $\mathbf{x} \in \mathbb{R}^d$ reshaped to a one-dimensional array, the convolutional operation applies learnable filters:

$$\mathbf{h}^{(l)} = \text{ReLU}(\text{BN}(\mathbf{W}^{(l)} * \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}))$$

where $*$ denotes convolution, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases of layer l,

BN denotes batch normalization, and ReLU is the rectified linear unit activation function.

The framework employs three convolutional blocks with filter counts of 64, 128, and 256 respectively. Filter sizes of 3 with stride 1 capture local patterns while preserving spatial resolution. Max pooling with pool size 2 follows each convolutional block, reducing dimensionality and providing translation invariance.

3.4.2 LSTM Component

The LSTM component receives the CNN-extracted features and models temporal dependencies across sequences of network flows. LSTM units maintain cell state $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$ that persist across time steps, with gating mechanisms controlling information flow.

The LSTM update equations are:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

where $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ are the forget, input, and output gates respectively, σ is the sigmoid function, and $\odot$ denotes element-wise multiplication.

The framework employs a bidirectional LSTM architecture, processing sequences in both forward and backward directions. This enables the model to capture dependencies in both temporal directions, improving detection of attacks where relevant context may appear either before or after the current flow.

$$\overrightarrow{\mathbf{h}}_t = \text{LSTM}(\mathbf{x}_t, \overrightarrow{\mathbf{h}}_{t-1})$$

$$\overleftarrow{\mathbf{h}}_t = \text{LSTM}(\mathbf{x}_t, \overleftarrow{\mathbf{h}}_{t+1})$$

$$\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$$

3.4.3 Attention Mechanism

An attention mechanism is incorporated to enable the model to focus on the most relevant time steps when making classification decisions. The attention weights are computed as:

$$\alpha_t = \frac{\exp(\mathbf{w}^T \tanh(\mathbf{W}_a \mathbf{h}_t + \mathbf{b}_a))}{\sum_{t'} \exp(\mathbf{w}^T \tanh(\mathbf{W}_a \mathbf{h}_{t'} + \mathbf{b}_a))}$$

The context vector is computed as the weighted sum of hidden states:

$$\mathbf{c} = \sum_t \alpha_t \mathbf{h}_t$$

This context vector, enriched with attention-weighted temporal information, feeds into the final classification layers.

3.4.4 Classification Layers

The classification head consists of fully connected layers with dropout regularization:

$$\mathbf{z} = \text{Dropout}(\text{ReLU}(\mathbf{W}_1 \mathbf{c} + \mathbf{b}_1))$$

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{W}_2 \mathbf{z} + \mathbf{b}_2)$$

For binary classification (normal vs. attack), the output layer has two units with softmax activation. For multi-class classification (distinguishing attack types), the output layer has units equal to the number of classes.

3.5 Training Procedure

The model is trained using the Adam optimizer with learning rate scheduling. The initial learning rate is set to 0.001 with exponential decay. The loss function is categorical cross-entropy with class weights inversely proportional to class frequencies:

$$\mathcal{L} = -\sum_{i=1}^{n} \sum_{c=1}^{C} w_c \cdot y_{i,c} \cdot \log(\hat{y}_{i,c})$$

where w_c is the weight for class c, $y_{i,c}$ is the true label indicator, and $\hat{y}_{i,c}$ is the predicted probability.

Training employs early stopping with patience of 10 epochs, monitoring validation loss to prevent overfitting. Batch size is set to 256, balancing computational efficiency with gradient estimation quality.

IV. EXPERIMENTAL SETUP

4.1 Datasets

The framework is evaluated using four benchmark intrusion detection datasets that collectively represent diverse network environments, attack types, and traffic characteristics.

4.1.1 NSL-KDD Dataset

NSL-KDD is a refined version of the original KDD Cup 1999 dataset, addressing several known issues including duplicate records and proportional representation problems. The dataset contains 125,973 training records and 22,544 test records, with 41 features per record. Traffic is labeled as normal or one of four attack categories: DoS (Denial of Service), Probe, R2L (Remote to Local), and U2R (User to Root).

Class	Training Samples	Test Samples
Normal	67,343	9,711
DoS	45,927	7,458
Probe	11,656	2,421
R2L	995	2,754
U2R	52	200

4.1.2 CICIDS2017 Dataset

The Canadian Institute for Cybersecurity Intrusion Detection System 2017 dataset contains realistic network traffic generated over five days of operation. The dataset includes benign traffic and common attack types including brute force, DoS, DDoS, web attacks, infiltration, and botnet traffic. The dataset contains approximately 2.8 million records with 78 features.

Class	Samples
Benign	2,273,097
DoS/DDoS	380,318
PortScan	158,930
Brute Force	13,835
Web Attack	2,180
Infiltration	36
Botnet	1,966

4.1.3 CSE-CIC-IDS2018 Dataset

This dataset extends CICIDS2017 with additional attack scenarios and larger scale. It includes seven attack scenarios: brute force, botnet, DoS, DDoS, web attacks, infiltration, and heartbleed. The dataset contains over 16 million records, providing a challenging scale for evaluating computational efficiency.

4.1.4 UNSW-NB15 Dataset

The UNSW-NB15 dataset was created using the IXIA PerfectStorm tool to generate realistic modern normal and attack traffic. It contains 2,540,044 records with 49 features, covering nine attack categories: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

4.2 Evaluation Metrics

The framework is evaluated using standard classification metrics appropriate for intrusion detection:

Accuracy (ACC): The proportion of correctly classified instances:

$$\text{ACC} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision:- The proportion of positive predictions that are correct:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall:- The proportion of actual positives correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score:- The harmonic mean of precision and recall:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

False Positive Rate:- The proportion of actual negatives incorrectly classified as positive:

$$\text{FPR} = \frac{FP}{FP + TN}$$

Area Under ROC Curve (AUC):- Measures the model's ability to discriminate between classes across all classification thresholds.

Matthew's Correlation Coefficient (MCC): A balanced metric suitable for imbalanced datasets:

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot}{\sqrt{(TP + FP)(TP + FN)(TN}$$

4.3 Baseline Methods

The proposed framework is compared against several baseline methods:

- **Random Forest (RF):** Ensemble of decision trees with bagging.
- **XGBoost:** Gradient boosting with decision tree base learners.
- **Support Vector Machine (SVM):** With radial basis function kernel.
- **Multi-Layer Perceptron (MLP):** Fully connected neural network.
- **Standalone CNN:** Convolutional network without LSTM.
- **Standalone LSTM:** Recurrent network without CNN.
- **Autoencoder:** Unsupervised anomaly detection based on reconstruction error.

4.4 Implementation Details

The framework is implemented using Python 3.9 with the following libraries: TensorFlow 2.10 for deep learning models, XGBoost 1.7 for feature selection, scikit-learn 1.2 for preprocessing and baseline methods, and pandas/numpy for data manipulation.

Experiments are conducted on a server with dual Intel Xeon Gold 6248R processors, 256GB RAM, and NVIDIA A100 40GB GPU. Training times and inference latencies are measured under controlled conditions with no competing workloads.

4.5 Experimental Protocol

Each experiment follows a consistent protocol:

Data Splitting: Datasets are split into training (70%), validation (15%), and test (15%) sets using stratified sampling to preserve class distributions.

Preprocessing: The preprocessing pipeline is fit on training data only, then applied to validation and test data.

Feature Selection: XGBoost feature importance is computed on training data, and the top-k features are selected.

Model Training: The hybrid model is trained on training data with early stopping based on validation loss.

Evaluation: Final performance metrics are computed on the held-out test set.

Repetition: Each experiment is repeated five times with different random seeds, and mean and standard deviation are reported.

V. RESULTS AND ANALYSIS

5.1 Feature Selection Results

The XGBoost-based feature selection module identified the most discriminative features across all datasets. The number of selected features varied by dataset based on cross-validation optimization:

Dataset	Original Features	Selected Features	Reduction
NSL-KDD	41	18	56.1%
CICIDS2017	78	24	69.2%
CSE-CIC-IDS2018	80	26	67.5%
UNSW-NB15	49	21	57.1%

Across all datasets, certain feature categories consistently ranked highly: connection duration, byte counts (source and destination), packet counts, protocol flags, and service indicators. Dataset-specific features also emerged as important, such as flow-based features in the CIC datasets and content-based features in NSL-KDD.

The feature reduction significantly accelerated training without meaningful performance degradation. Training time decreased by approximately 40-50% compared to using all features, while accuracy remained within 0.5% of the full-feature baseline.

5.2 Binary Classification Results

Binary classification distinguishes between normal traffic and attacks regardless of attack type. The following table presents results across all datasets:

Dataset	Accuracy	Precision	Recall	F1-Score	FPR	AUC
NSL-KDD	98.72 ± 0.15	98.54 ± 0.18	98.89 ± 0.12	98.71 ± 0.14	1.08 ± 0.09	0.996
CICIDS2017	99.21 ± 0.08	99.15 ± 0.11	99.28 ± 0.07	99.21 ± 0.08	0.74 ± 0.06	0.998
CSE-CIC-IDS2018	99.02 ± 0.11	98.87 ± 0.14	99.18 ± 0.09	99.02 ± 0.11	0.92 ± 0.08	0.997
UNSW-NB15	98.56 ± 0.19	98.32 ± 0.22	98.79 ± 0.16	98.55 ± 0.18	1.19 ± 0.11	0.995

The framework achieves accuracy exceeding 98.5% on all datasets with false positive rates below 1.2%. The CICIDS2017 dataset shows the highest performance, likely due to its larger size and cleaner labeling. The UNSW-NB15 dataset presents the greatest challenge, with more diverse attack types and more realistic attack traffic that more closely resembles normal behavior.

5.3 Multi-Class Classification Results

Multi-class classification distinguishes between normal traffic and specific attack categories. Results for the NSL-KDD dataset (5 classes) are presented below:

Class	Precision	Recall	F1-Score
Normal	98.89	99.12	99.00
DoS	99.21	99.34	99.27
Probe	97.45	96.89	97.17
R2L	94.32	91.78	93.03
U2R	87.65	82.50	85.00

Performance varies significantly across attack categories. DoS attacks, which are high-volume and exhibit distinctive traffic patterns, are detected with near-perfect accuracy. R2L and U2R attacks, which are low-volume and may span multiple sessions, present greater detection challenges. The attention mechanism helps with these difficult cases by allowing the model to focus on the most relevant time steps in a sequence.

5.4 Comparison with Baseline Methods

The proposed hybrid framework outperforms all baseline methods across datasets:

Method	NSL-KDD	CICIDS2017	CSE-CIC-IDS2018	UNSW-NB15
Random Forest	95.42	96.78	96.34	94.89
XGBoost	96.18	97.45	97.12	95.67
SVM	93.21	94.56	93.89	92.34
MLP	94.87	95.92	95.45	93.78
CNN (standalone)	96.89	97.98	97.67	96.12
LSTM (standalone)	97.23	98.34	98.01	96.45
Autoencoder	91.45	93.12	92.67	90.23
Proposed (CNN-LSTM)	98.72	99.21	99.02	98.56

The hybrid CNN-LSTM architecture outperforms standalone CNN and LSTM models by 1.5-2.5 percentage points, demonstrating the value of combining spatial and temporal feature learning. Traditional machine learning methods (RF, XGBoost, SVM) achieve competitive performance but fall short of the deep learning approaches, particularly on the larger and more complex datasets.

5.5 Computational Performance

Detection latency and throughput are critical for real-time deployment. The following measurements were obtained:

Metric	Value
Training Time (NSL-KDD)	847 seconds
Training Time (CICIDS2017)	3,412 seconds
Inference Latency (single flow)	2.8 ms
Inference Throughput (batch)	18,400 flows/second
GPU Memory Usage	4.2 GB
Model Size	28.4 MB

The inference latency of 2.8 ms per flow is suitable for real-time detection in most cloud environments. Batch

processing achieves throughput exceeding 18,000 flows per second, adequate for high-volume traffic scenarios. Memory and storage requirements are modest, enabling deployment on standard cloud instances.

5.6 Ablation Study

An ablation study quantifies the contribution of each framework component:

Configuration	Accuracy	F1-Score
Full Model	98.72	98.71
Without Feature Selection	98.45	98.42
Without Attention	98.21	98.18
Without Bidirectional LSTM	97.89	97.85
Without CNN	97.23	97.19
Without LSTM	96.89	96.84

All components contribute positively to performance. The CNN and LSTM components provide the largest individual contributions, confirming the value of the hybrid architecture. Feature selection improves efficiency more than raw accuracy, while the attention mechanism provides modest but consistent gains, particularly for minority attack classes.

VI. DISCUSSION

6.1 Key Findings

The experimental results support several important conclusions about hybrid deep learning approaches for cloud intrusion detection.

First, the combination of CNNs and LSTMs captures complementary aspects of network traffic that neither architecture captures alone. CNNs excel at extracting local patterns and feature interactions within individual flows, while LSTMs model the temporal evolution of attack behavior across flow sequences. The hybrid architecture achieves consistent improvements over standalone approaches across all evaluated datasets.

Second, XGBoost-based feature selection provides an effective preprocessing step that reduces computational requirements without sacrificing detection accuracy. The 57-69% reduction in feature dimensionality accelerates training significantly while maintaining performance within 0.5% of the full-feature baseline. This finding aligns with prior work demonstrating that network traffic datasets contain substantial redundancy.

Third, the framework generalizes effectively across datasets representing different network environments, attack types, and collection methodologies. Performance remains strong on all four benchmark datasets despite their significant differences. This generalization suggests the learned representations capture fundamental characteristics of attacks rather than dataset-specific artifacts.

Fourth, attention mechanisms provide modest but meaningful improvements, particularly for difficult-to-detect attack categories. By enabling the model to focus on relevant time steps, attention helps detect low-volume attacks like R2L and U2R that span multiple sessions with subtle indicators.

6.2 Practical Implications

The framework's computational characteristics support practical deployment in cloud security operations. The 2.8ms detection latency enables real-time alerting for individual suspicious flows, while batch processing throughput exceeds requirements for most production traffic volumes. The model's modest memory footprint (4.2GB GPU, 28.4MB storage) enables deployment on standard cloud GPU instances without specialized hardware.

Integration with existing security infrastructure requires consideration of several factors. The framework operates on flow-level features that can be extracted from NetFlow/IPFIX records, packet captures, or cloud-native flow logs. Organizations can deploy the framework alongside existing signature-based IDS to provide complementary coverage—signature systems detect known threats with zero false positives, while the ML-based framework catches novel attacks that evade signatures.

Alert fatigue remains a concern for any automated detection system. The framework's sub-1.2% false positive rate represents significant improvement over many anomaly detection approaches, but at scale this still generates substantial alert volume. Integration with security orchestration and automated response (SOAR) platforms can help triage alerts and automate response to high-confidence detections.

6.3 Limitations

Several limitations should be acknowledged when interpreting these results.

Dataset constraints: While the evaluated datasets are widely used benchmarks, they may not fully represent the diversity of real-world cloud traffic. Benchmark datasets often contain idealized attack traffic that may be easier to detect than sophisticated real-world attacks. Additionally, the datasets were collected years ago and may not include the latest attack techniques.

Label quality: The framework relies on supervised learning, requiring accurately labeled training data. In practice, obtaining comprehensive labels for all attack types

is challenging, and label noise can degrade model performance. Semi-supervised or self-supervised approaches that reduce label requirements merit investigation.

Adversarial robustness: The evaluation does not address adversarial attacks where malicious actors deliberately craft traffic to evade detection. Machine learning models can be vulnerable to adversarial examples, and attackers aware of ML-based detection may adapt their techniques accordingly.

Encrypted traffic: The framework assumes access to flow-level features. As encryption adoption increases, content-based features become unavailable, potentially reducing detection capability for attacks that rely on payload characteristics.

6.4 Comparison with Recent Literature

The proposed framework's performance compares favorably with recent published results. Studies employing similar hybrid deep learning approaches on the same benchmark datasets report accuracies in the 97-99% range, consistent with our findings.

The framework's multi-dataset evaluation distinguishes this work from many prior studies that evaluate on single datasets. Cross-dataset generalization provides stronger evidence that proposed methods will transfer to new environments rather than merely fitting the idiosyncrasies of particular benchmarks.

VII. CONCLUSION AND FUTURE WORK

7.1 Summary of Contributions

This paper presented a hybrid intrusion detection framework for cloud computing environments that combines machine learning and deep learning techniques. The framework integrates XGBoost-based feature selection with a CNN-LSTM architecture enhanced by attention mechanisms. Comprehensive evaluation across four benchmark datasets demonstrates state-of-the-art detection performance with accuracy exceeding 98.5%, false positive rates below 1.2%, and computational characteristics suitable for real-time deployment.

The key contributions include:

A complete pipeline from raw network data to actionable security alerts, addressing preprocessing, feature selection, and classification in an integrated framework.

Empirical demonstration that hybrid spatial-temporal deep learning outperforms both traditional machine learning and standalone deep learning approaches for intrusion detection.

Multi-dataset evaluation providing evidence of generalization beyond dataset-specific patterns.

Practical deployment analysis including latency, throughput, and resource requirements for production cloud environments.

7.2 Future Research Directions

Several directions warrant future investigation:

Federated learning could enable collaborative model training across organizations without sharing sensitive traffic data. Privacy-preserving techniques would allow security providers to benefit from diverse training data while maintaining confidentiality.

Continual learning approaches could address concept drift as attack techniques evolve over time. Models that update incrementally without catastrophic forgetting would maintain relevance without complete retraining.

Explainability enhancements would improve the actionability of alerts by identifying which features or traffic characteristics triggered detection. Attention weights provide partial interpretability, but dedicated explanation techniques could improve security analyst efficiency.

Adversarial robustness evaluation and hardening would address the threat of evasion attacks. Adversarial training and certified defense techniques merit investigation for high-security deployments.

Multi-tenant awareness could tailor detection to tenant-specific baselines in shared cloud environments. Personalized models that capture individual tenant behavior while sharing attack detection capabilities present interesting architectural challenges.

REFERENCES

[1] Anderson, J. P. (1980). Computer security threat monitoring and surveillance. Technical Report, James P. Anderson Company.

[2] Denning, D. E. (1987). An intrusion-detection model. IEEE Transactions on Software Engineering, SE-13(2), 222-232.

[3] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.

[4] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

[5] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[6] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. IEEE Symposium on Computational Intelligence for Security and Defense Applications, 1-6.

[7] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and

intrusion traffic characterization. International Conference on Information Systems Security and Privacy, 108-116.

[8] Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. Military Communications and Information Systems Conference, 1-6.

[9] Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. International Conference on Learning Representations.

[10] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[11] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525-41550.

[12] Zhang, C., Patras, P., & Haddadi, H. (2019). Deep learning in mobile and wireless networking: A survey. IEEE Communications Surveys & Tutorials, 21(3), 2224-2287.

[13] Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications, 50, 102419.

[14] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321-357.

[15] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations.