2026 Volume: 02 Issue: 11 www.ijrpa.com ISSN 2456-9995 Research Article

International Journal Research Publication Analysis

Page: 01-33

A BLOCKCHAIN-SUPPORTED DUAL-SAMPLING DILATED CONVOLUTIONAL NEURAL NETWORK MODEL FOR INTRUSION DETECTION IN IOT NETWORKS

***Aliyu Yakubu, A. A. Sadiq, H. B. Mamman, Ashiru Musa**

Department of Electrical and Electronics Engineering, Faculty of Engineering and Engineering Technology, Abubakar Tafawa Balewa University (ATBU), P.M.B. 0248, Bauchi, Nigeria.

Article Received: 28 September 2026
Article Revised: 18 October 2026
Published on: 07 November 2026

***Corresponding Author: Aliyu Yakubu**
Department of Electrical and Electronics Engineering, Faculty of Engineering and Engineering Technology, Abubakar Tafawa Balewa University (ATBU), P.M.B. 0248, Bauchi, Nigeria.

DOI: https://doi-doi.org/101555/ijrpa.6096

ABSTRACT

The increasing proliferation of Internet of Things (IoT) devices has expanded network attack surfaces and created significant challenges for intrusion detection systems (IDS), particularly in detecting attacks within imbalanced traffic data and maintaining the integrity and transparency of intrusion-event records. This study developed and evaluated a Dual-Sampling Dilated Convolutional Neural Network (ConNet) for network intrusion detection and formulated a conceptual blockchain-based mechanism for fair and tamper-resistant validation and logging of detected intrusion events. The proposed ConNet architecture combines coarse and fine feature-sampling pathways with dilated convolution to capture complementary multi-scale features from network traffic. The model was implemented in Python using TensorFlow/Keras and evaluated using the NSL-KDD benchmark dataset, with KDDTrain+ used for model development and KDDTest+ used as the independent test set. Its performance was compared with five architectures: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Three-Layer Neural Network (TNN), CNN-LSTM and CNN-TNN. Evaluation was based on accuracy, precision, recall, F1-score, specificity, Matthews Correlation Coefficient (MCC), false positive rate (FPR), false negative rate (FNR), RMSE, MAE and Receiver Operating Characteristic (ROC) analysis with Area Under the Curve (AUC). The proposed ConNet achieved the highest accuracy (82.06%), recall (74.40%), F1-score (82.52%) and MCC (66.19%) among the six models, while recording the lowest FNR

(25.60%), RMSE (0.388) and MAE (0.179). Its AUC of 0.962 was also the highest among the evaluated models. However, the model produced a relatively higher FPR of 7.82%, indicating a trade-off between increased attack detection sensitivity and false alarms. In addition, a conceptual blockchain mechanism incorporating SHA-256 hashing, fairness scoring and weighted voting was formulated to support transparent and tamper-resistant intrusion-event validation. The findings demonstrate that the proposed ConNet achieved the best overall classification performance among the evaluated architectures, particularly in detecting a greater proportion of attack instances and reducing missed intrusions. The blockchain component provides a conceptual foundation for enhancing the integrity and transparency of intrusion-event records, however, its performance under real distributed blockchain conditions remains to be experimentally validated.

KEYWORD: Internet of Things (IoT); Intrusion Detection System (IDS); Dual-Sampling Dilated CNN; Deep Learning; Blockchain; NSL-KDD; Class Imbalance.

1. INTRODUCTION

The rapid proliferation of the Internet of Things (IoT) has transformed modern digital ecosystems by enabling billions of interconnected devices to collect, communicate, and process data autonomously. Recent studies estimate that IoT devices surpassed 30 billion globally, creating unprecedented opportunities across healthcare, industry, smart environments, and critical infrastructures (Awajan, 2023). However, this hyper-connectivity also exposes IoT infrastructures to extensive security vulnerabilities, ranging from unauthorized access to large-scale cyberattacks such as Distributed Denial-of-Service (DDoS), spoofing, and malware injection (Susilo & Sari, 2020).

Traditional security methods such as fixed-rule authentication, encryption, and access control are insufficient because IoT nodes are resource-constrained and operate in dynamic network environments (Islam et al., 2021).

Intrusion Detection Systems (IDS) have therefore become essential in monitoring IoT traffic and identifying anomalies that signal malicious activity. Conventional signature-based IDS can only recognise attacks that match stored patterns, while classical machine-learning approaches depend heavily on hand-crafted features and often struggle with the heterogeneous, high-volume traffic of IoT networks. Deep learning, particularly using neural networks such as CNNs, DNNs, LSTMs, and MLPs, has been reported to enhance intrusion detection accuracy by learning complex traffic patterns and identifying sophisticated attacks

(Awajan, 2023; Rybacki et al., 2024). Benchmark datasets such as NSL-KDD, BoT-IoT, and IoTID20 are widely used in the development and testing of these IDS models (Islam et al., 2021; Susilo & Sari, 2020). Despite these advancements, centralized deep-learning IDS frameworks remain vulnerable to model poisoning, adversarial manipulation, and single-point failures, especially in distributed IoT environments. Moreover, because such models are mostly developed and tested on benchmark data, their reported performance reflects the conditions of those datasets and does not, on its own, establish how they will behave against previously unseen attacks or in live IoT traffic.

A further limitation concerns how traffic features are represented and how imbalanced class distributions are handled. Network traffic records contain patterns at different scales, and attack categories are often under-represented relative to normal traffic or to other attack categories, which can bias a classifier towards the majority class and increase the number of missed intrusions. Models that learn richer, multi-scale feature representations are therefore of interest. In this regard, dual-branch and dual-sampling architectures, together with advanced convolutional networks, have improved feature extraction from complex medical image datasets such as CT scans and chest X-rays, as reported by Vineth Ligi et al., (2022). These imaging studies are not evidence that the same approach will improve intrusion detection in IoT networks. They are used in this study only as architectural inspiration for combining coarse and fine sampling pathways with dilated convolution to capture complementary multi-scale features from network traffic. Whether this design benefits intrusion detection is a question to be tested empirically, which this study does using the NSL-KDD benchmark.

A final concern is that even an accurate detector produces intrusion alerts whose records must themselves be trustworthy. When detection logs are held in centralized storage, they can be altered, deleted, or disputed after an incident, which creates a need for secure and tamper-resistant records of intrusion events. This need has motivated researchers to integrate blockchain technology into intrusion detection frameworks because of its decentralization, immutability, and tamper-resistant logging capabilities. Blockchain has shown promise in securing Software-Defined Networks (SDNs) by ensuring transparent communication between controllers, detecting control plane attacks, and improving resilience against false-data injection (Alkhamisi et al., 2024).

In healthcare IoT, blockchain is being combined with federated learning to secure sensitive medical data, provide reliable intrusion detection, and enhance patient safety (Alzakari et al., 2024). Moreover, blockchain enabled federated machine learning allows distributed IDS

models to be trained collaboratively without sharing raw data, thereby reducing privacy risks and enabling scalable detection of attacks such as DDoS (Saveetha et al., 2024).

However, achieving consensus in blockchain-enhanced IDS frameworks poses performance trade-offs in speed, energy use, and fairness among participating nodes. Recent literature therefore emphasizes the need for optimized, fair, and lightweight blockchain-based consensus mechanisms, coupled with high-performance deep-learning models, to enable secure, scalable, and accurate intrusion detection in IoT environments (Saveetha et al., 2024). In this study, blockchain is treated as a conceptual component: a mechanism based on SHA-256 hashing, fairness scoring, and weighted voting is formulated and illustrated by simulation to support tamper-resistant validation and logging of detected intrusion events. It is not implemented as a distributed network and is not experimentally compared with an IDS that has no blockchain component.

Against this background, this research develops and evaluates a Dual-Sampling Dilated Convolutional Neural Network (ConNet) for network intrusion detection on the NSL-KDD benchmark, and formulates a conceptual blockchain-based mechanism for fair and tamper-resistant logging of detected intrusion events. The experimental evaluation is limited to NSL-KDD, and the study does not claim to demonstrate the detection of zero-day or other genuinely unseen attacks.

The rapid growth of the Internet of Things (IoT) has led to the deployment of interconnected devices in various domains, including healthcare, transportation, smart homes, and industrial automation. However, the increasing connectivity of these devices has also exposed IoT environments to a wide range of cyber-attacks, posing significant threats to data confidentiality, integrity, and availability. Existing intrusion detection systems (IDSs) often struggle to effectively detect attacks in IoT networks due to the heterogeneous nature of IoT traffic, resource constraints, and the continuous evolution of attack patterns. Limited intrusion-detection sensitivity. Deep-learning IDS models trained on traffic in which normal records and some attack categories dominate can be biased towards the majority class, and single-scale feature extraction may fail to capture the complementary local and wider patterns in network traffic. The consequence is that a considerable share of attack instances may go undetected, and missed intrusions are particularly costly in IoT environments.

2. MATERIALS AND METHOD

This study employs an experimental simulation methodology comprising an empirical deep-learning evaluation and a conceptual blockchain simulation. The two components are treated

differently throughout the thesis. The deep-learning component (ConNet and the five comparison models) was implemented, trained on KDDTrain+, and empirically evaluated on the independent KDDTest+ set, so its results are experimental findings. The blockchain component was not evaluated experimentally: it is a conceptual mechanism illustrated through a Python simulation of a hash-chained ledger and fairness-weighted voting among simulated nodes, and no claims are made about its performance, scalability, or security under attack. The study does not evaluate the two components as an integrated pipeline and does not compare blockchain-integrated with non-blockchain-integrated detection.

Materials and Tools

The experimental setup consisted of a laptop computer running the Windows operating system. The implementation was carried out using Python as the primary programming language, together with several specialized libraries. TensorFlow with its Keras API served as the deep learning framework for constructing the Dual-Sampling Dilated CNN. Scikit-learn was used for machine learning operations and performance metric computations, including accuracy, precision, recall, and F1-score. Pandas and NumPy were used for data manipulation and numerical computations, while Matplotlib and Seaborn were used for visualizations such as confusion matrices and ROC curves. The conceptual blockchain model was implemented using Python's hashlib library to compute SHA-256 hashes for the tamper-evident ledger. Key implementations include the construction of the dual-sampling CNN layers, the implementation of dilated convolution, model training and testing, confusion matrix and ROC curve generation, statistical performance analysis, and the simulation of the fairness-weighted voting mechanism. Software and hardware details: Python, TensorFlow/Keras, scikit-learn, Pandas, NumPy, processor, memory, GPU, random seed.

Methods

The study uses the NSL-KDD benchmark dataset (Tavallaee et al., 2009). Two official files were used: KDDTrain+ (125,973 records) for model development, and KDDTest+ (22,543 records) as the independent, held-out test set. The files were used as distributed, so no additional random train–test split (such as a 70/30 split) was applied, and all six models were evaluated on the same 22,543 test records. KDDTest+ contains attack types that do not occur in KDDTrain+, so it is a more demanding test of generalization than a random split of the training file would be.

Preprocessing was performed using Python's Pandas library for data handling and cleaning, LabelEncoder for categorical features, and scikit-learn's MinMaxScaler for normalization. Because the training and test sets are separate files, the encoder mappings and the scaler's minimum and maximum values were learned from KDDTrain+ only and then applied to KDDTest+, so that no information from the test set influenced training. Model development was carried out in TensorFlow/Keras and comprised parallel coarse-sampling, fine-sampling, and dilated-convolution branches, as specified in Section 3.3.4. Training used the Adam optimizer, with early stopping to limit overfitting and ReduceLROnPlateau to adjust the learning rate. Both callbacks monitored on a validation subset held out from KDDTrain+; KDDTest+ was not used for early stopping, learning-rate scheduling, or any other model-selection decision. Model validation used accuracy, precision, recall, F1-score, specificity, MCC, FPR, FNR, RMSE, MAE, the confusion matrix, and the ROC curve with AUC.

Methodological Flowchart

Figure 1 shows the system architecture of the proposed blockchain-supported dual-sampling dilated CNN for IoT intrusion detection.

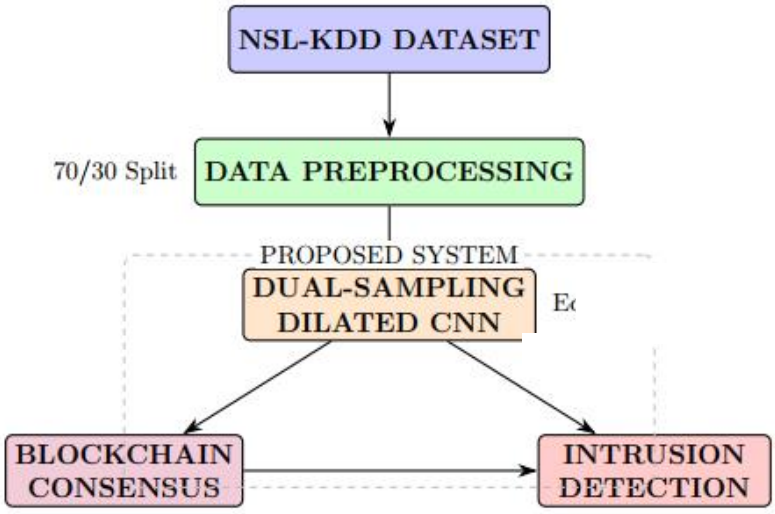


Figure 1: System Architecture.

The proposed blockchain-supported dual-sampling dilated CNN intrusion detection system is organized into seven sequential phases. Phase 1 acquires the NSL-KDD dataset: KDDTrain+ (125,973 samples) is used for training and KDDTest+ (22,543 samples) is kept as the independent test set. Phase 2 preprocesses the data through cleaning, label encoding, and MinMaxScaler normalization (fitted on the training file only), with no further train-test split. Phase 3 forms the core CNN architecture with an input layer of 41 features, three parallel

branches (coarse sampling, fine sampling, and dilated convolution), followed by upsampling, a supporting attention step, fusion via depth-wise concatenation, global average pooling, fully connected layers with dropout, and a softmax output for binary classification. Phase 4 trains the model using the Adam optimizer, categorical crossentropy loss, batch size of 128, early stopping, and ReduceLROnPlateau. Phase 5 evaluates performance using accuracy, precision, recall, F1-score, specificity, MCC, FPR, FNR, RMSE, MAE, and AUC, generating the confusion matrix and ROC curve. Phase 6 runs the conceptual blockchain simulation with 10 nodes over 100 rounds, using participation tracking, fairness scoring, and weighted voting for validation, and a SHA-256 hash-chained, tamper-evident ledger. Phase 7 produces final outputs including visualizations, the trained H5 model, and a performance report. The system flows vertically with the three CNN branches converging at the fusion layer. A detail explanation of the flowchart is described in Figure 2.

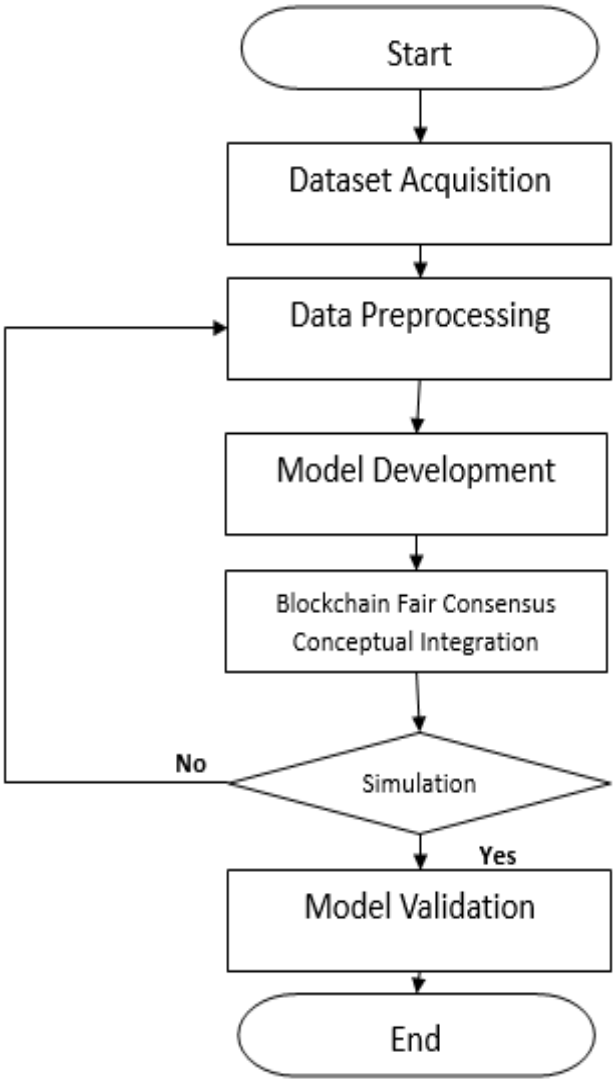


Figure 2: system flowchart.

The process begins with the Start node, after which the system proceeds to Dataset Acquisition**,** where the NSL-KDD dataset (KDDTrain+.csv and KDDTest+.csv) is obtained. This is followed by Data Preprocessing**,** which involves cleaning, feature encoding using LabelEncoder, and normalization with MinMaxScaler (no further train-test split is applied, since KDDTest+ is held out as the test set). The next stage is Model Development, where the dual-sampling dilated CNN architecture is constructed, including the three parallel branches (coarse sampling, fine sampling, and dilated convolution), fusion layer, and softmax output layer. Subsequently, the system proceeds to Blockchain Fair Consensus Conceptual Integration, which simulates fairness-weighted voting, node participation tracking, and SHA-256 hash-chained logging for tamper-evident records. At this point, a decision branch is encountered: if the simulation results are not satisfactory, the system loops back to Model Development for further refinement; if satisfactory, the system proceeds to Model Validation, where performance metrics (accuracy, precision, recall, F1-score, specificity, MCC, FPR, FNR, RMSE, MAE, and AUC) are computed. Finally, the process reaches the End node.

Data Collection and Data Preprocessing

This study uses the publicly available NSL-KDD dataset (Tavallaee et al., 2009), a refined version of the KDD Cup 1999 data from which redundant records were removed. Each record describes one network connection using 41 features (continuous, discrete, and categorical), together with a class label and a difficulty-level score. The three categorical features are protocol_type, service, and flag. The attack records in NSL-KDD belong to four categories: Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R); all other records are labelled Normal. NSL-KDD does not contain separate DDoS, botnet, or brute-force categories, and none are attributed to it in this study. The class label was converted to a binary target (Normal versus Attack), and the difficulty-level score was not used as an input feature. Because NSL-KDD is a general network-intrusion benchmark and not IoT-specific traffic, the results are benchmark results.

Data Preprocessing Steps

i. Import the KDDTrain+ and KDDTest+ files into Python using Pandas
ii. Handle missing or corrupted values
iii. Encode the categorical features (protocol_type, service, flag) with LabelEncoder and convert the class label to a binary target (Normal versus Attack)

iv. Normalize numerical features to the range [0, 1] with MinMaxScaler, fitted on KDDTrain+ only

v. Use KDDTrain+ for training and KDDTest+ as the held-out test set (no further split)

Design of Dual-Sampling Dilated CNN (ConNet)

The proposed Dual-Sampling Dilated Convolutional Neural Network (ConNet) architecture is specifically designed to address the challenges of imbalanced intrusion datasets and the need for multi-scale feature extraction in IoT network traffic analysis. The architecture employs a parallel multi-branch design that simultaneously captures both global and local patterns from network flow data, enabling robust detection of both common and rare attack types. The design is inspired by successful dual-branch networks in medical image segmentation and adapted for the unique characteristics of network intrusion detection, where attacks exhibit varying temporal and spatial signatures. The architecture consists of four main components that work synergistically to extract hierarchical features and produce accurate classification decisions.

Network Traffic Representation

Network traffic data is the input to the proposed intrusion detection system. In NSL-KDD, each traffic instance is a connection record represented as a feature vector of 41 attributes, such as duration, protocol type, service, connection flag, and the numbers of source and destination bytes. After preprocessing, each record is arranged as a one-dimensional sequence of 41 values with a single channel (input shape 41 × 1), so that it can be processed by one-dimensional convolutional layers. The network traffic representation is defined as:

$$X = \{x_1, x_2, x_3, \dots, x_n\} \quad (1)$$

Where X represents the input feature vector of one connection record, x_i denotes an individual feature, and n = 41 is the number of input features.

Dilated Convolution Layer

The dilated convolution layer forms a critical component of the proposed architecture, employing dilation rates to expand the receptive field of convolutional filters without increasing the number of parameters or sacrificing spatial resolution. The dilated convolution operation is mathematically represented as:

$$y(i) = \sum_{k=1}^{K} x[i + r \cdot k] \cdot w[k] \quad (2)$$

Where, r is the dilation rate, and K is the filter size.

Unlike standard convolutions that process adjacent inputs, dilated convolutions introduce gaps between sampled input positions, effectively increasing the receptive field exponentially with minimal computational overhead. This capability is particularly advantageous in intrusion detection because attack patterns often span multiple time steps and involve correlated features distributed across various network layers. The dilation rate can be adjusted to capture different scales of temporal dependencies, from short-term burst patterns to long-term persistent attack behaviors. By incorporating dilated convolutions, the architecture can detect both immediate anomalies and gradual changes in network behavior without requiring deeper networks or larger filter sizes that would increase computational requirements. The dilation mechanism also improves the model's ability to generalize across different attack types by learning multi-scale representations that are invariant to the specific temporal distribution of attack signatures. This component significantly enhances the model's capability to detect sophisticated attacks that employ evasion techniques or attempt to mask their presence within normal traffic patterns.

CNN Feature Extraction

The CNN feature extraction mechanism transforms raw network traffic data into meaningful high-level representations through hierarchical convolutional operations. The feature extraction process begins with the input layer receiving the normalized traffic matrix, followed by sequential convolution and pooling operations. The convolution operation extracts spatial features from the input using learnable kernels, mathematically represented as:

$$F_{\text{conv}} = \sigma(W * X + b) \qquad (3)$$

*Where, W represents the kernel weights, * denotes the convolution operation, b is the bias term, σ is the activation function and Fconv is the extracted feature map.*

Fully Connected and Softmax Layers

The fully connected and softmax layers constitute the final classification stage of the proposed ConNet architecture, responsible for transforming the multi-scale features extracted by the parallel branches into definitive classification decisions. After the features from all branches are fused through depth-wise concatenation and processed through global average pooling to reduce dimensionality, the resulting feature vector is passed through a series of fully connected layers that perform high-level reasoning and pattern association. These layers

learn complex combinations of the extracted features, capturing nonlinear relationships that are essential for distinguishing between normal traffic and various attack categories. The fully connected layers incorporate dropout regularization to mitigate overfitting, ensuring that the model generalizes well to unseen data and maintains robust performance across diverse IoT network environments.

The final output layer employs a softmax activation function that produces probability distributions over the possible classes, enabling the system to make probabilistic decisions about the nature of the analyzed network traffic. For binary classification (normal vs. attack), the softmax layer outputs two probabilities that sum to unity, with the higher probability determining the final classification. In multi-class scenarios, the softmax layer can accommodate multiple attack categories, providing detailed threat classification that aids security analysts in appropriate response selection. The softmax function is mathematically expressed as:

$$P(y = j \mid x) = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \quad (4)$$

Where, C is the number of classes, and zj represents the logit values from the preceding fully connected layer.

The combination of fully connected and softmax layers ensures that the network produces not only accurate classifications but also confidence scores that can be used for decision threshold adjustment and false alarm management. This classification stage completes the dual-sampling dilated CNN pipeline, delivering actionable intrusion detection results that balance high detection accuracy with acceptable false positive rates, making the architecture suitable for deployment in resource-constrained IoT environments.

Loss Function

The loss function quantifies the discrepancy between the model's predicted outputs and the actual labels during training, guiding the optimization process through backpropagation. The categorical cross-entropy loss was used for the two-class (Normal versus Attack) classification task, with one-hot encoded labels, and is defined as:

$$L_{CE} = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}) \quad (5)$$

Where N is the number of samples, C is the number of classes (C = 2: Normal and Attack), y(i,c) is the one-hot encoded true label, and ŷ(i,c) is the predicted probability that sample i belongs to class c.

Architecture Specification and Training Configuration

To allow the model to be reproduced, Table 1 lists the layers of ConNet in order, and Table 2 lists the training configuration.

Table 1: lists the layers of ConNet in order.

Stage	Layer	Configuration
Input	Input	Shape: (41, 1) 41 features, 1 channel
Fine sampling branch	Conv1D	Filters: 32, kernel size: 3, stride: 1, padding: same, activation: ReLU
Fine sampling branch	BatchNormalization	Applied after the first convolution
Fine sampling branch	Conv1D	Filters: 64, kernel size: 3, stride: 1, padding: same, activation: ReLU
Coarse sampling branch	AveragePooling1D	Pool size: 2, padding: same
Coarse sampling branch	Conv1D	Filters: 32, kernel size: 3, stride: 1, padding: same, activation: ReLU
Coarse sampling branch	BatchNormalization	Applied after the first convolution
Coarse sampling branch	Conv1D	Filters: 64, kernel size: 3, stride: 1, padding: same, activation: ReLU
Up-sampling	UpSampling1D	Size: 2; applied to the coarse branch, followed by cropping or zero-padding when necessary to match the input sequence length
Dilated branch	Conv1D (dilated)	Filters: 64, kernel size: 3, dilation rate: 2, padding: same, activation: ReLU
Dilated branch	BatchNormalization	Applied after the first dilated convolution
Dilated branch	Conv1D (dilated)	Filters: 64, kernel size: 3, dilation rate: 4, padding: same, activation: ReLU
Fusion	Concatenate	Depth-wise concatenation along the channel axis; 192 channels
Attention	GlobalAveragePooling1D	Squeeze-and-excitation channel attention
Attention	Dense	Units: 24, activation: ReLU
Attention	Dense	Units: 192, activation: sigmoid
Attention	Reshape and Multiply	Reshape to (1, 192) and multiply by the fused feature map
Pooling	GlobalAveragePooling1D	Converts the attended feature map into a feature vector
Fully connected	Dense	Units: 128, activation: ReLU
Regularization	Dropout	Rate: 0.4
Fully connected	Dense	Units: 64, activation: ReLU
Regularization	Dropout	Rate: 0.3

Output	Dense	2 units, softmax activation (Normal versus Attack)
Model size	Total parameters	68,314 parameters
Model size	Trainable parameters	68,058 parameters

Table 2: Training Configuration.

Item	Setting
Optimizer	Adam
Initial learning rate	0.001 (1×10^{-3})
Loss function	Categorical cross-entropy
Batch size	256
Maximum epochs	30 for the proposed CNN-TNN model; 10 for the baseline models
Early stopping	Monitor: val_loss; patience: 7 for the proposed model and 3 for the baselines; restore best weights: Yes
ReduceLROnPlateau	Monitor: val_loss; factor: 0.5; patience: 3 for the proposed model; minimum learning rate: 0.000001 (1×10^{-6})
Validation data	20% of KDDTrain+ for the proposed model, using stratified splitting; 20% internal validation split for the baseline models
Random seed	42

Hash-Chained Ledger (Tamper-Evidence)

Each block in the simulated ledger stores an intrusion-event record together with the hash of the previous block, so that the blocks form a hash chain. The hash of a block is computed with the SHA-256 function, using Python's hashlib library, as:

$$H = \mathrm{SHA} - 256\,(M) \qquad (6)$$

Where M is the message formed by the block contents (the event record and the previous block's hash), and H is the resulting 256-bit hash value.

Because each block's hash depends on the previous hash, altering any recorded event changes that block's hash and breaks the links of all later blocks, so tampering can be detected when the chain is re-verified. This provides tamper-evidence (detection of modification), not tamper-prevention, and the hash chain does not by itself decide which records are valid or which nodes may add them.

Fairness-Weighted Voting (Consensus Rule)

Whether an event record is accepted is decided by a weighted voting rule among the participating nodes. Each node n casts a vote V_n (1 if it judges the record valid, 0

otherwise), and the vote is weighted by the node's fairness score F_n, so that the total weighted vote is:

$$V_{\text{total}} = \sum_{n=1}^{N} F_n \cdot V_n \quad (7)$$

Where N is the number of participating nodes, F_n is the fairness score of node n, and V_n is the vote (0 or 1) cast by node n.

A record is accepted when V_total ≥ θ · ΣF_n, where θ is the acceptance threshold. Weighting by a bounded fairness score is intended to limit the influence of any single node and to reduce the effect of nodes with inconsistent participation. This property was illustrated in the simulation but was not proven, and it was not tested against adversarial behavior such as vote manipulation, collusion, or data poisoning.

Fairness Score

The fairness score F_n of node n is computed from its participation consistency using a sigmoid function:

$$F_n = \frac{1}{1+e^{-x_n}} \quad (8)$$

Where x_n represents the participation consistency of node n. The sigmoid bounds F_n between 0 and 1, so the score rises with consistent participation but cannot grow without limit. The simulation used 10 nodes over 100 rounds.

Scope and Limitations of the Blockchain Component

The simulation is a software model in Python in which nodes are represented programmatically. It does not implement network communication, leader selection, Byzantine fault tolerance, digital signatures, or incentive mechanisms, and it was not compared with established consensus protocols such as Proof of Work, Proof of Stake, or PBFT. Fairness here means participation-weighted voting influence as defined by Equations (9) and (10); no formal fairness guarantee is claimed. The component therefore demonstrates a design concept and does not provide experimental evidence of performance, scalability, or security.

Validation of Model Performance

Model performance was evaluated on the held-out KDDTest+ set using standard metrics commonly applied in intrusion detection, computed with Python's scikit-learn library. The

Attack class is treated as the positive class. The confusion-matrix terms are defined as follows:

TP (True Positive): an attack correctly classified as an attack. TN (True Negative): normal traffic correctly classified as normal. FP (False Positive): normal traffic incorrectly flagged as an attack. FN (False Negative): an attack incorrectly classified as normal

Accuracy

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (9)$$

Precision

$$\text{Precision} = \frac{TP}{TP+FP} \quad (10)$$

Recall (Detection Rate)

$$\text{Recall} = \frac{TP}{TP+FN} \quad (11)$$

F1-Score

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (12)$$

False Positive Rate (FPR)

$$\text{FPR} = \frac{FP}{FP+TN} \quad (13)$$

False Negative Rate (FNR)

$$\text{FNR} = \frac{FN}{FN+TP} \quad (14)$$

Receiver Operating Characteristic (ROC)

$$\text{TPR} = \frac{TP}{TP+FN} \quad (15)$$

The ROC curve is plotted using the True Positive Rate (TPR) against the False Positive Rate as the decision threshold is varied.

Area Under the Curve (AUC)

AUC is the area under the ROC curve and evaluates the discriminative capability of the classifier across all decision thresholds; a value of 0.5 corresponds to random ranking and 1.0 to perfect separation. It is computed as:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}) \quad (16)$$

Specificity

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (17)$$

Matthews Correlation Coefficient (MCC)

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (18)$$

Root Mean Square Error (RMSE) and Mean Absolute Error (MAE)

RMSE and MAE are error measures usually associated with regression. In this study the task is binary classification, so they are not used as regression metrics. They are computed from the model's predicted probability of the Attack class, p_i ∈ [0, 1], and the true binary label y_i ∈ {0, 1} (Attack = 1, Normal = 0) over the N test records

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2} \quad (19)$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - p_i| \quad (20)$$

Both measures are on the same 0–1 scale as the probabilities, and lower values are better. MAE is the average absolute gap between the predicted probability and the true label; RMSE squares the gaps before averaging and therefore penalizes confident wrong predictions more heavily. They are supplementary measures of how far the predicted probabilities lie from the true labels, and they depend on how well calibrated the probabilities are as well as on the decision threshold. If hard 0/1 class predictions were used instead of probabilities, MAE would equal the misclassification rate (1 − accuracy) and RMSE would be its square root, so neither would add information beyond accuracy. RMSE and MAE are therefore interpreted alongside, and not in place of, the classification metrics above.

Visualizations (confusion matrix, ROC curve, training history) are generated using Python's matplotlib and seaborn libraries.

3. RESULTS AND DISCUSSION

A confusion matrix is a two-dimensional table that cross-tabulates the actual class of each test sample against the class predicted by the model, and it forms the basis from which all threshold-dependent performance metrics used in this study, including accuracy, precision, recall, specificity, FPR and FNR, are derived. In each of the confusion matrices presented in this section, the rows represent the actual (true) labels, Normal and Attack, while the columns represent the labels predicted by the respective model. For consistency in interpretation across the six models, the Attack class is treated as the positive class throughout this chapter; consequently, a True Positive (TP) is an attack correctly identified as an attack, a True Negative (TN) is normal traffic correctly identified as normal, a False Positive (FP) is normal traffic incorrectly flagged as an attack, and a False Negative (FN) is an actual attack that the model fails to detect and instead classifies as normal. Because a False Negative represents an undetected intrusion, it is the most operationally critical error type in an intrusion detection

system, while a False Positive represents a false alarm that, though less severe, still consumes analyst time and network resources. Each confusion matrix is presented and explained individually below, after which a consolidated summary table is provided for ease of comparison.

Confusion Matrix of the Convolutional Neural Network (CNN) Model

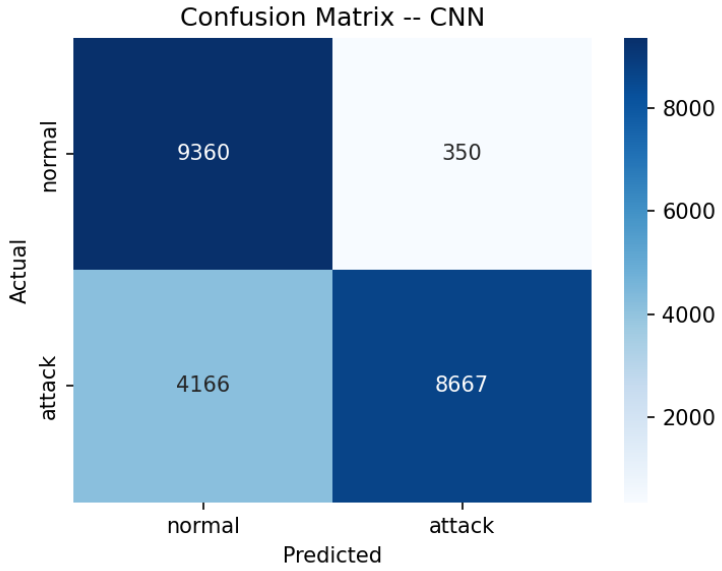


Figure 3a: Confusion Matrix – CNN.

Figure 3a presents the confusion matrix obtained from the standalone CNN model. Out of the 22,543 test samples, the model correctly classified 9,360 normal instances as normal and 8,667 attack instances as attack, giving True Negative and True Positive counts of 9,360 and 8,667 respectively. However, the model misclassified 350 normal instances as attacks (False Positives) and, more significantly, misclassified 4,166 actual attacks as normal traffic (False Negatives). These figures translate to an overall accuracy of 79.97%, a precision of 96.12%, and a recall (detection rate) of 67.54%. The relatively low False Positive count indicates that the CNN model is conservative in raising alarms and rarely misclassifies benign traffic, which explains its high precision and specificity of 96.40%. However, the comparatively large number of False Negatives shows that the standalone CNN, which relies solely on standard convolutional filters with a fixed receptive field, struggles to detect a substantial proportion of attacks, particularly those with subtle or minority-class signatures such as R2L and U2R attacks that are underrepresented in the training data.

Confusion Matrix of the CNN-LSTM Hybrid Model

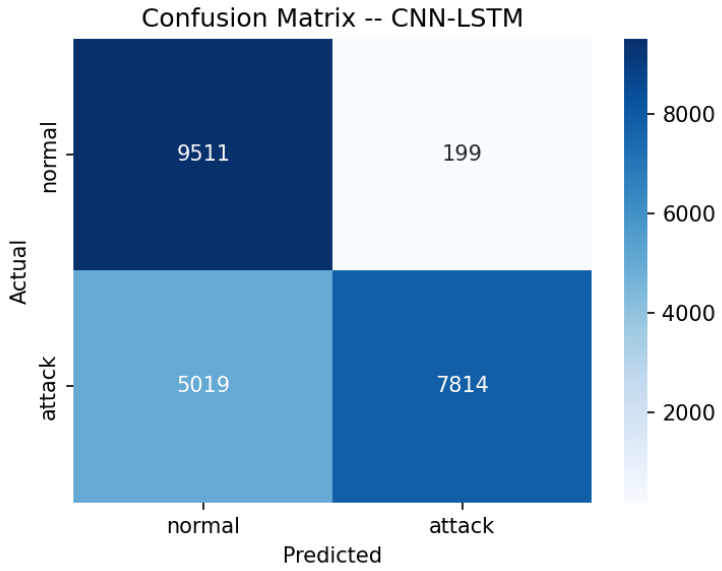


Figure 3b: Confusion Matrix -- CNN-LSTM.

Figure 3b shows the confusion matrix for the hybrid CNN-LSTM model, which combines convolutional feature extraction with sequential learning through an LSTM layer. The model correctly classified 9,511 normal samples (True Negatives) and 7,814 attack samples (True Positives), while misclassifying 199 normal samples as attacks (False Positives) and 5,019 actual attacks as normal (False Negatives). This corresponds to an accuracy of 76.85%, a precision of 97.52%, and a recall of 60.89%, the lowest recall recorded among the six models evaluated in this study. Although the CNN-LSTM model achieved the second-lowest False Positive count of all the models, indicating very high specificity (97.95%), its False Negative count of 5,019 is the highest recorded, meaning that a large share of genuine intrusions was left undetected. This outcome suggests that, for this dataset, sequential (temporal) modelling of already convolution-extracted features did not translate into improved detection sensitivity, and instead increased the tendency of the model to default to the majority Normal class for borderline traffic patterns.

Confusion Matrix of the CNN-TNN Hybrid Model

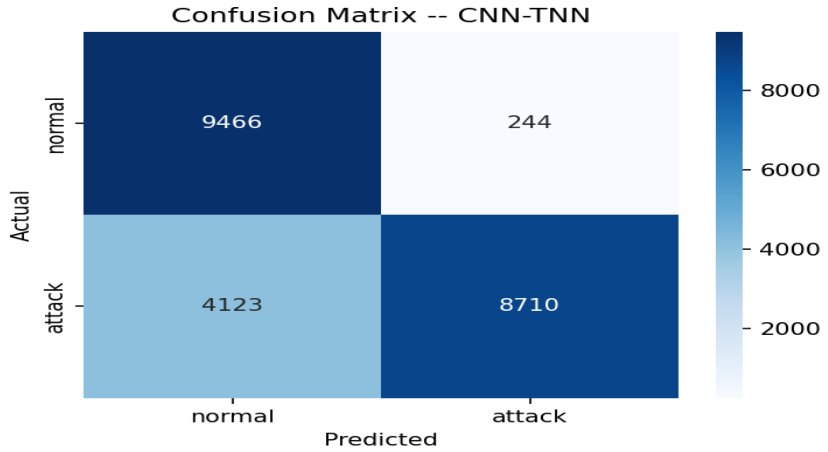


Figure 3c: Confusion Matrix -- CNN-TNN.

Figure 3c illustrates the confusion matrix generated by the CNN-TNN hybrid model. This model correctly classified 9,466 normal instances and 8,710 attack instances, while producing 244 False Positives and 4,123 False Negatives. The resulting accuracy is 80.63%, with a precision of 97.27% and a recall of 67.87%. Compared with the CNN-LSTM model, the CNN-TNN architecture achieved a higher True Positive count (8,710 against 7,814) and a lower False Negative count (4,123 against 5,019), indicating that the trilayered neural network component integrated with convolutional layers was more effective than the LSTM component at retaining discriminative attack features. The CNN-TNN model also recorded one of the highest specificity values (97.49%), showing that its ability to correctly recognise normal traffic remained strong even as its sensitivity to attacks improved relative to the CNN-LSTM model.

Confusion Matrix of the Proposed Dual-Sampling Dilated CNN (ConNet) Model

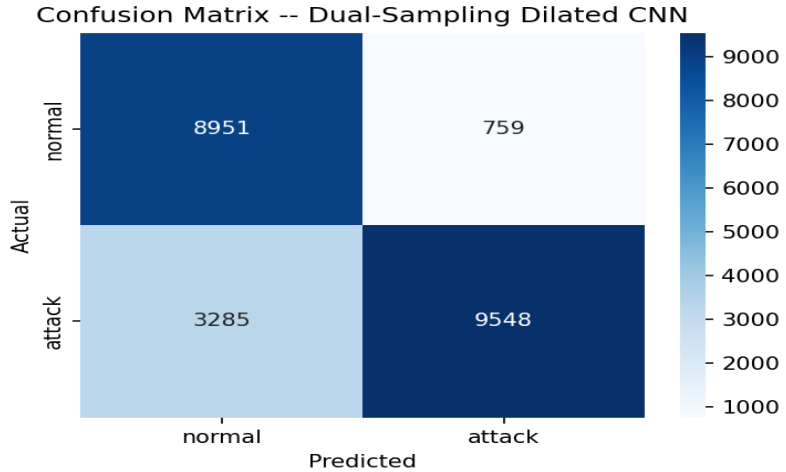


Figure 3d: Confusion Matrix -- Dual-Sampling Dilated CNN.

Figure 3d presents the confusion matrix of the proposed Dual-Sampling Dilated CNN (ConNet) model. The model correctly classified 8,951 normal instances and 9,548 attack instances, the highest True Positive count achieved by any of the six models evaluated. It produced 759 False Positives and 3,285 False Negatives, the lowest False Negative count recorded in this study. These values correspond to an accuracy of 82.06%, a precision of 92.64%, and a recall of 74.40%, all consistent with the highest overall accuracy, the highest recall, and the highest F1-score (82.52%) among the six models. Although the proposed model recorded a higher False Positive count than the other five architectures, reflecting a comparatively lower specificity of 92.18%. This result indicates a trade-off in the present experiment: the proposed model detected more attack instances but also generated more false alarms than the comparison models. In an intrusion detection context, where a missed attack (False Negative) is generally more costly than a false alarm (False Positive), this gain in sensitivity is operationally relevant, and is consistent with the second objective of this study, which is to develop a deep-learning intrusion detection model capable of improved detection accuracy and handling of imbalanced IoT attack data.

Confusion Matrix of the LSTM Model

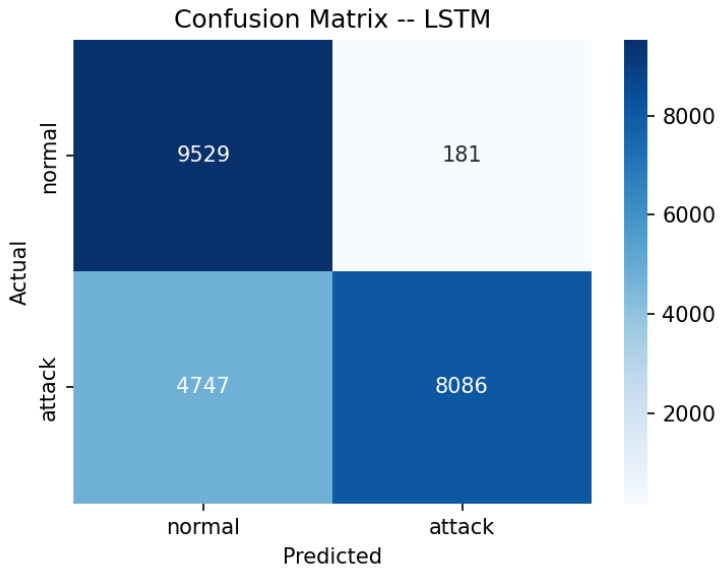


Figure 3e: Confusion Matrix – LSTM.

Figure 3e shows the confusion matrix for the standalone LSTM model. The model correctly classified 9,529 normal samples and 8,086 attack samples, while misclassifying 181 normal samples as attacks and 4,747 actual attacks as normal. This produced an accuracy of 78.14%, a precision of 97.81%, and a recall of 63.01%. The LSTM model recorded the lowest False

Positive count (181) and the highest specificity (98.14%) among all six models, indicating that it is highly reliable at correctly recognising normal traffic and rarely raises false alarms. However, its comparatively high False Negative count of 4,747 shows that a purely sequence-based architecture, without convolutional feature extraction, is less able to capture the localised, feature-level irregularities that distinguish many attack types in the NSL-KDD dataset, resulting in a moderate detection rate.

Confusion Matrix of the TNN Model

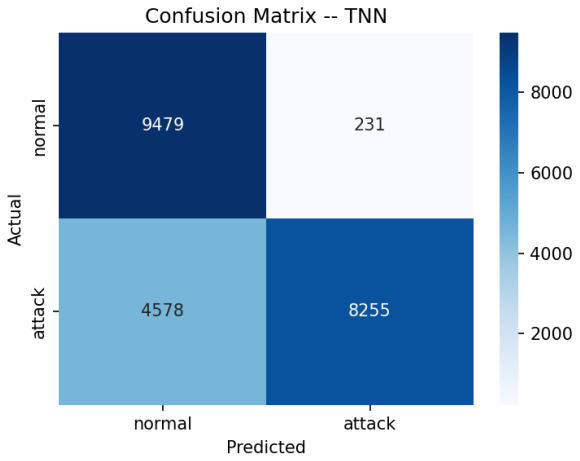


Figure 3f: Confusion Matrix – TNN.

Figure 3f presents the confusion matrix obtained from the standalone TNN model. The model correctly classified 9,479 normal samples and 8,255 attack samples, while recording 231 False Positives and 4,578 False Negatives. This corresponds to an accuracy of 78.67%, a precision of 97.28%, and a recall of 64.33%. The TNN model performed marginally better than the LSTM model in terms of True Positives (8,255 against 8,086) and overall accuracy, while maintaining a similarly low False Positive count and high specificity (97.62%). Taken together, the results in Figures 8 to 13 show a consistent pattern: the proposed Dual-Sampling Dilated CNN achieved higher sensitivity to attacks (higher recall and higher True Positive counts) than the CNN-only, LSTM-only, TNN-only and hybrid models. This result indicates a trade-off in the present experiment: the proposed model detected more attack instances but also generated more false alarms than the comparison models.

Table 3: Summary of Confusion Matrix Values for the Six Intrusion Detection Models.

Model	TN	FP	FN	TP	Total
Dual-Sampling Dilated CNN	8,951	759	3,285	9,548	22,543
CNN-TNN	9,466	244	4,123	8,710	22,543
CNN	9,360	350	4,166	8,667	22,543
TNN	9,479	231	4,578	8,255	22,543
LSTM	9,529	181	4,747	8,086	22,543
CNN-LSTM	9,511	199	5,019	7,814	22,543

Table 3 consolidates the raw classification outcomes of all six models. It is evident that the proposed Dual-Sampling Dilated CNN recorded both the highest number of correctly detected attacks (TP = 9,548) and the lowest number of missed attacks (FN = 3,285), while the CNN-LSTM model recorded the lowest number of correctly detected attacks (TP = 7,814) and the highest number of missed attacks (FN = 5,019). Conversely, the LSTM model recorded the fewest false alarms (FP = 181), followed closely by the CNN-LSTM model (FP = 199), showing that models incorporating a recurrent or temporal component tend to be more conservative in flagging attacks. These raw counts form the basis of the derived performance metrics discussed in the following section.

Comparative Performance Metrics

To enable a holistic comparison of the six models beyond raw classification counts, seven derived performance metrics, namely accuracy, precision, recall, F1-score, AUC, specificity and MCC, were computed for each model using the equations presented in Section 3.4. Table 2 summarises these metrics, while Figure 14 presents them graphically.

Table 4: Comparative Performance Metrics of the Six Intrusion Detection Models. (%)

Model	Accuracy	Precision	Recall	F1-Score	AUC	Specificity	MCC
Dual-Sampling Dilated CNN	82.06	92.64	74.40	82.52	96.20	92.18	66.19
CNN-TNN	80.63	97.27	67.87	79.96	95.35	97.49	66.14
CNN	79.97	96.12	67.54	79.33	95.51	96.40	64.62
TNN	78.67	97.28	64.33	77.44	95.61	97.62	63.31
LSTM	78.14	97.81	63.01	76.64	87.69	98.14	62.83
CNN-LSTM	76.85	97.52	60.89	74.97	94.60	97.95	60.87

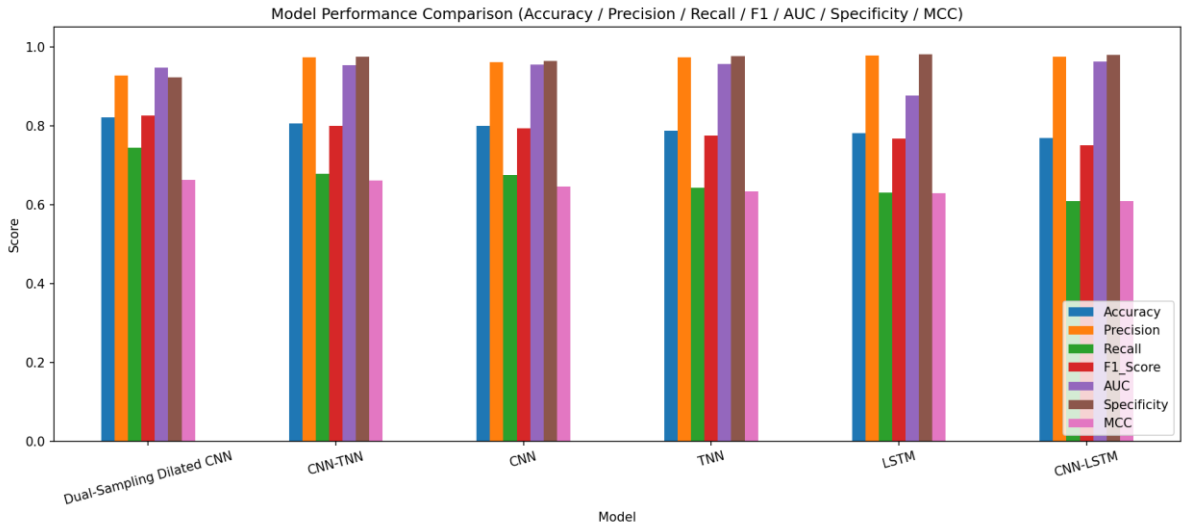


Figure 4: Model Performance Comparison (Accuracy, Precision, Recall, F1-Score, AUC, Specificity, MCC.)

Figure 4 presents a grouped bar chart comparing the seven-performance metrics across the six models. Two clear patterns emerge from the chart. First, the blue Accuracy bars are consistently tallest for the proposed Dual-Sampling Dilated CNN (82.06%), and decline progressively across CNN-TNN, CNN, TNN, LSTM, and CNN-LSTM, which recorded the lowest accuracy (76.85%). Second, the orange Precision bars follow an almost inverse pattern: the sequential and hybrid models (LSTM, TNN, CNN-TNN, CNN-LSTM) achieve the highest precision, all above 97%, while the Dual-Sampling Dilated CNN records the lowest precision (92.64%) of the group, although this remains a high value in absolute terms. This inverse relationship between accuracy/recall on one hand and precision/specificity on the other is a direct consequence of the confusion matrix values discussed, models that are more conservative about labelling traffic as an attack naturally make fewer False Positives (raising precision and specificity) but also miss more genuine attacks (lowering recall and, ultimately, overall accuracy and F1-score). The green Recall bars in Figure 4 show the same ordering as the True Positive counts in Table 1, with the Dual-Sampling Dilated CNN clearly ahead of all other models at 74.40%, roughly ten percentage points above the CNN-LSTM model's 60.89%. The red F1-Score bars, which balance precision and recall, again place the Dual-Sampling Dilated CNN first (82.52%), confirming that its improvement in recall is not achieved at a disproportionate cost to precision. The purple AUC bars are comparatively close for five of the six models (all between approximately 94.6% and 96.2%), with the conspicuous exception of the LSTM model, whose AUC of 87.69% is visibly shorter than the rest, indicating a weaker overall ranking ability between the normal and attack classes despite

its high precision. The brown Specificity bars mirror the precision pattern, while the pink MCC bars, which provide a single balanced measure that accounts for all four confusion matrix quadrants, again rank the Dual-Sampling Dilated CNN (66.19%) and CNN-TNN (66.14%) as the two strongest models, closely followed by CNN (64.62%), with CNN-LSTM recording the lowest MCC (60.87%).

ROC Curve Comparison

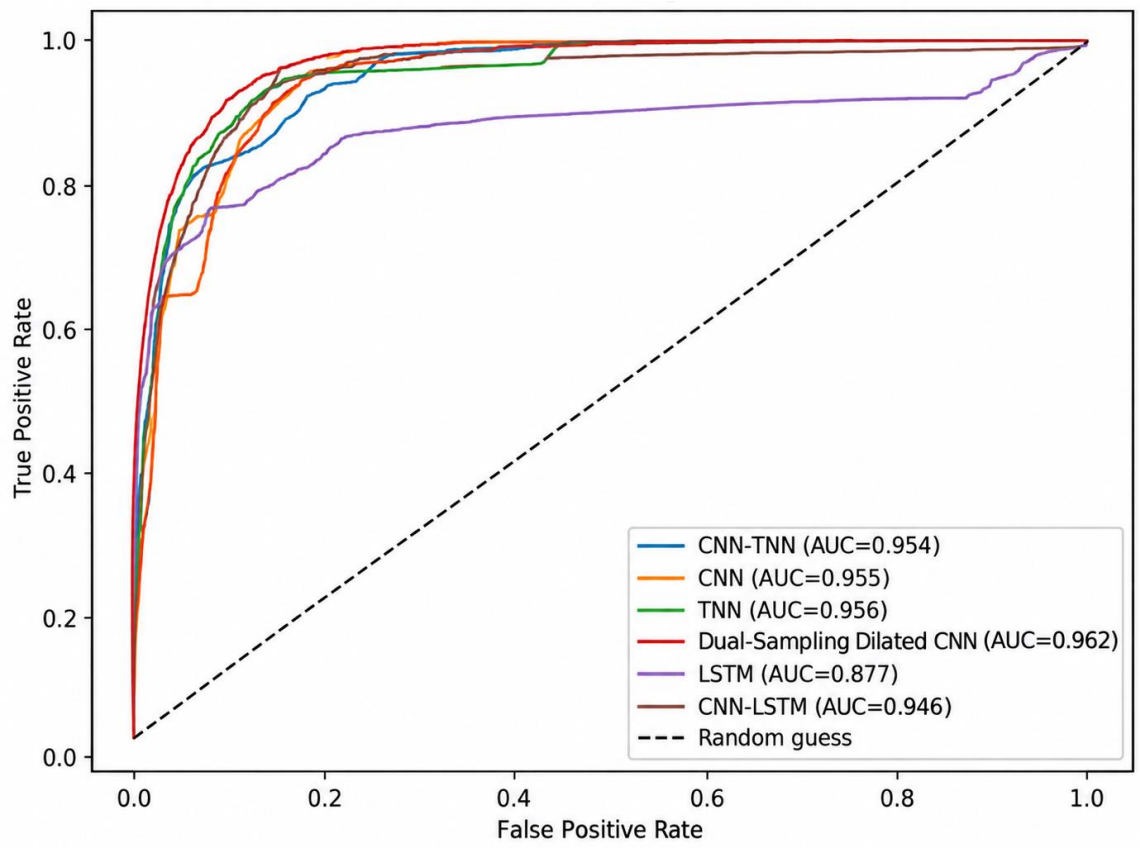


Figure 5: ROC Curve Comparison.

Figure 5 presents the Receiver Operating Characteristic (ROC) curves for all six models, illustrating the relationship between the True Positive Rate (TPR) and False Positive Rate (FPR) across different classification thresholds. The diagonal dashed line represents the performance of a random classifier, corresponding to an AUC of 0.5. Among all six models, the Dual-Sampling Dilated CNN achieves the highest AUC of 0.962, demonstrating the strongest overall discriminative capability and outperforming the CNN-TNN (AUC = 0.954), CNN (AUC = 0.955), TNN (AUC = 0.956), CNN-LSTM (AUC = 0.946), and LSTM (AUC = 0.877) models. Its ROC curve rises rapidly toward the top-left region of the plot, indicating that the proposed Dual-Sampling Dilated CNN is highly effective at distinguishing between normal and attack traffic across a wide range of decision thresholds. The CNN-TNN, CNN,

TNN, and CNN-LSTM curves also demonstrate strong classification performance, with AUC values above 0.94, but none surpasses the Dual-Sampling Dilated CNN. In particular, the relatively small AUC differences among the convolution-based models indicate that they all possess strong discriminative ability; however, the 0.962 AUC recorded by the Dual-Sampling Dilated CNN confirms its superior overall performance when threshold-independent classification capability is considered.

In contrast, the LSTM model produces a visibly lower ROC curve across much of the plot and records the lowest AUC of 0.877. This substantial difference confirms that the LSTM has weaker overall ability to rank normal and attack instances correctly compared with the other models. The ROC analysis therefore reinforces the findings from the preceding evaluation and provides additional evidence that the Dual-Sampling Dilated CNN is the best-performing model among the six evaluated approaches. Its highest AUC indicates that the model maintains strong sensitivity while controlling false-positive rates across different operating thresholds, making it the most robust model for distinguishing between normal and attack traffic in this study.

Model Error Comparison

In addition to the classification metrics discussed above, four error-oriented metrics, RMSE, MAE, FPR and FNR, were computed for each model to further characterise their predictive error behaviour. Lower values are preferred for all four metrics. Table 5 presents the numeric values, while Figure 6 presents them graphically.

Table 5: Comparative Error Metrics of the Six Intrusion Detection Models.

Model	RMSE	MAE	FPR (%)	FNR (%)
Dual-Sampling Dilated CNN	0.3877	0.1793	7.82	25.60
CNN-TNN	0.4332	0.2047	2.51	32.13
CNN	0.4154	0.1983	3.60	32.46
TNN	0.4352	0.2081	2.38	35.67
LSTM	0.4339	0.2133	1.86	36.99
CNN-LSTM	0.4720	0.2302	2.05	39.11

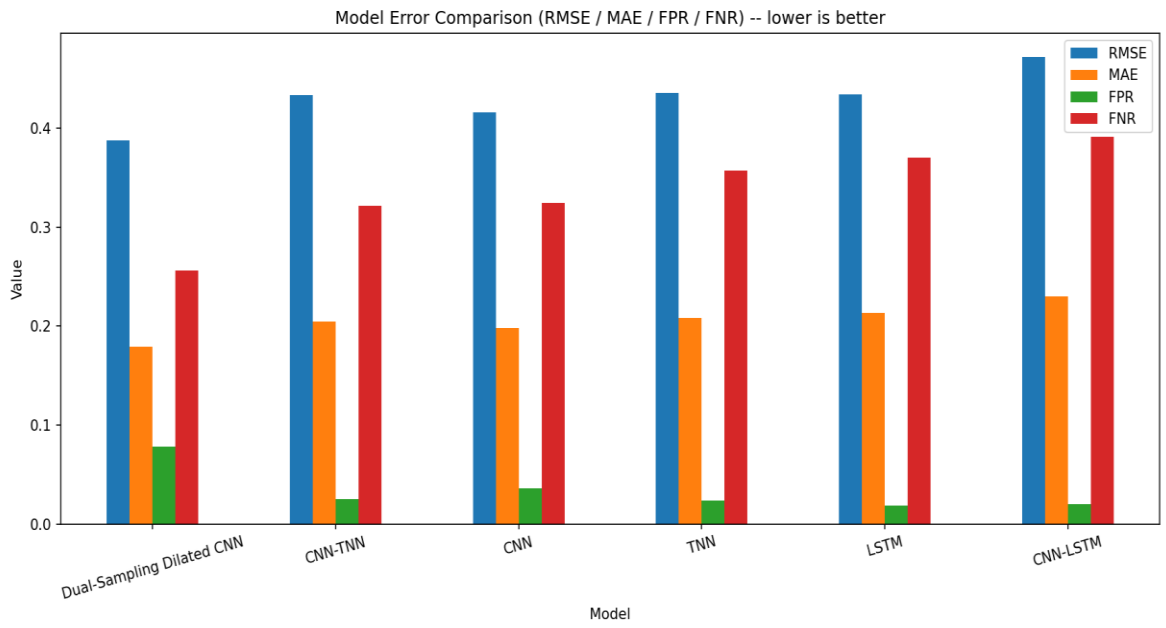


Figure 6: Model Error Comparison. (RMSE, MAE, FPR, FNR)

Figure 6 compares the RMSE, MAE, FPR and FNR values of the six models. The Dual-Sampling Dilated CNN recorded the lowest RMSE (0.388) and lowest MAE (0.179) of all six models, indicating that its predicted class probabilities deviate least, on average, from the true labels across the entire test set. The CNN-LSTM model, by contrast, recorded the highest RMSE (0.472) and highest MAE (0.230), consistent with its comparatively weaker classification performance observed in Table 5. When the green FPR bars and red FNR bars are compared, an important trade-off becomes visible: the Dual-Sampling Dilated CNN has the tallest FPR bar (7.82%) of the six models, reflecting its higher tendency to flag normal traffic as an attack, but it has by far the shortest FNR bar (25.60%), whereas every other model records an FNR above 32%, with the CNN-LSTM model reaching the highest FNR of 39.11%, meaning that almost two out of every five actual attacks in the test set went undetected by that model. For an intrusion detection system, where the consequence of a missed attack (False Negative) is typically more severe than the consequence of an unnecessary alert (False Positive), the substantially lower FNR of the proposed model is the difference of greatest operational relevance between the proposed model and the other five architectures in this experiment. RMSE and MAE are treated here as supplementary measures only; lower values do not by themselves establish superior classification performance, which is assessed primarily through the confusion-matrix-based metrics.

DISCUSSION OF FINDINGS

The results presented consistently show that the proposed Dual-Sampling Dilated CNN outperformed the five comparison models on the metrics most relevant to intrusion detection, namely accuracy (82.06%), recall (74.40%), F1-score (82.52%), MCC (66.19%), RMSE (0.388) and, most notably, FNR (25.60%). This outcome supports the theoretical motivation for the architecture discussed, where the coarse sampling branch was designed to capture global, network-wide traffic patterns while the fine sampling branch was designed to preserve detailed, local traffic characteristics, and the dilated convolution layer was introduced to expand the receptive field of the network without increasing its parameter count. The combination of these three mechanisms enabled the model to detect a substantially larger proportion of the minority attack instances than the other five architectures, due to regarding the effectiveness of dual-branch and multi-scale architectures for extracting features from complex, imbalanced datasets.

Precision-Recall Trade-off Across Models

A recurring theme across Tables 4 and 5 and Figures 5 to 6 is the trade-off between precision (and, correspondingly, specificity) and recall (and, correspondingly, sensitivity to attacks). The five comparison models, particularly LSTM, TNN and CNN-LSTM, all achieved precision and specificity values above 96%, meaning that they are very unlikely to misclassify normal traffic as an attack. However, this conservative behaviour comes at the cost of recall values below 68% in all five cases, meaning that these models fail to detect between roughly one-third and two-fifths of genuine attacks. The proposed Dual-Sampling Dilated CNN accepted a moderate reduction in precision (92.64%) and specificity (92.18%) in exchange for a substantial improvement in recall (74.40%), and its F1-score, which balances both quantities, was nonetheless the highest of the six models. This finding illustrates that, for intrusion detection tasks in particular, evaluating a model solely on precision or accuracy can be misleading, and that recall, F1-score and FNR should be weighted heavily in model selection, since the operational cost of a missed intrusion generally exceeds the cost of investigating a false alarm. This is best understood as a security operating trade-off: a detection model can be tuned toward fewer false alarms or toward fewer missed attacks, and in the present experiment the proposed model sits at the more attack-sensitive end of that trade-off.

False Positive and False Negative Rate Implications

The FPR and FNR results illustrate an important operational trade-off in intrusion detection. The proposed Dual-Sampling Dilated CNN produced the highest FPR of 7.82% but the lowest FNR of 25.60% among the six evaluated models. This means that the model identified a larger proportion of attack instances but also generated more false alarms.

From an intrusion-detection perspective, a high FNR can be particularly undesirable because malicious traffic may remain undetected. Conversely, a high FPR can increase the number of alerts requiring investigation and may contribute to analyst workload. The appropriate balance between these two error types therefore depends on the operational requirements of the intended security environment.

For the present experiment, the proposed model provides greater attack sensitivity than the comparison models. However, this result was obtained using the NSL-KDD benchmark dataset and should not be interpreted as direct evidence of performance in a live IoT environment. Evaluation using contemporary IoT-specific datasets and live traffic would be required to establish practical deployment performance.

Convolutional versus Sequential and Hybrid Architectures

C Comparing the four baseline and hybrid models (CNN, LSTM, TNN, CNN-LSTM, CNN-TNN) provides further insight into which architectural components contribute most to detection performance. The standalone LSTM model recorded the lowest AUC (87.69%) of the six models, showing that a purely recurrent architecture, without any convolutional feature extraction stage, is comparatively less effective at learning the discriminative feature representations needed for this classification task. Combining CNN with LSTM (CNN-LSTM) did not improve recall or accuracy over the standalone CNN model; in fact, the CNN-LSTM model recorded the lowest accuracy, recall, F1-score and MCC of all six models, despite achieving the highest AUC (96.19%). This suggests that, for the NSL-KDD feature set used in this study, which does not have a strong inherent temporal ordering once the 41 features are represented as a single flattened vector per record, adding a recurrent layer after convolutional feature extraction increased model complexity without a corresponding improvement in default-threshold detection performance. The CNN-TNN model, by comparison, performed better than CNN-LSTM across almost every metric, including accuracy (80.63% against 76.85%) and recall (67.87% against 60.89%), indicating that the Three-Layered Neural Network component used in this study was more compatible with the convolutional feature maps than the LSTM component. Nonetheless, neither hybrid model

matched the performance of the proposed Dual-Sampling Dilated CNN, reinforcing the conclusion that the dual-sampling and dilated convolution mechanisms, rather than the addition of a sequential learning component, are the primary drivers of the performance improvement observed in this study.

Comparison with Related Empirical Studies

The findings of this study are broadly consistent with the empirical literature reviewed. Prior studies have shown that deep-learning architectures, particularly convolutional neural networks, tend to outperform purely statistical or shallow machine-learning approaches to intrusion detection on benchmark datasets such as NSL-KDD. The comparatively weaker performance of the standalone LSTM model observed in this study, especially its lower AUC, is also consistent with reports in the literature that recurrent architectures alone are less suited to feature sets, such as those derived from NSL-KDD, that lack an explicit sequential or time-series structure once transformed into fixed-length feature vectors. Furthermore, the superior recall and F1-score achieved by the proposed Dual-Sampling Dilated CNN corroborate findings from dual-branch and multi-scale architectures applied in other domains, where the combination of coarse and fine feature extraction pathways has been shown to improve sensitivity to minority patterns in imbalanced datasets. These parallels strengthen confidence that the performance gains observed in this study arise from genuine architectural advantages rather than from characteristics specific only to the NSL-KDD dataset.

4. CONCLUSION

Based on the results obtained in this study, the following conclusions are drawn:

i. The study successfully developed and evaluated a Dual-Sampling Dilated Convolutional Neural Network for network intrusion detection. The architecture combined coarse and fine sampling pathways with dilated convolution to provide complementary feature representations.

ii. The proposed model achieved the best overall performance among the six evaluated architectures. It recorded an accuracy of 82.06%, precision of 92.64%, recall of 74.40%, F1-score of 82.52%, MCC of 66.19% and AUC of 0.962. It also recorded the lowest FNR of 25.60%. The proposed model's higher recall and lower FNR were accompanied by a higher FPR of 7.82% and lower specificity of 92.18%. The results therefore demonstrate a trade-off between increased sensitivity to attack traffic and increased false alarms.

iii. The comparative evaluation provides empirical evidence that the complete proposed architecture performed better overall than the five comparison architectures under the experimental conditions used in this study. However, the absence of an ablation study means that the individual contributions of dual sampling and dilated convolution cannot be isolated. A blockchain-based intrusion-event validation mechanism incorporating SHA-256 hashing, fairness scoring and weighted voting was formulated and simulated. The mechanism illustrates how cryptographic hashing and participation-based voting could be incorporated into intrusion-event management.

Recommendations

Based on the findings and limitations of the study, the following recommendations are made:

i. Evaluation using contemporary IoT datasets: The proposed ConNet should be evaluated using contemporary IoT-specific intrusion datasets such as BoT-IoT, IoTID20 and other appropriate datasets to determine whether the observed performance generalises beyond NSL-KDD.

ii. Live IoT traffic evaluation: Future studies should evaluate the model using real or controlled IoT network traffic to assess its performance under realistic traffic volumes, device heterogeneity and operational conditions.

iii. Ablation studies: Controlled ablation experiments should be conducted to quantify the individual contributions of coarse sampling, fine sampling, dilated convolution, attention and feature fusion.

iv. Repeated experimental runs: Future experiments should be repeated across multiple independent runs and reported using mean ± standard deviation to determine the stability of the observed performance and reduce the influence of random initialization.

v. Threshold optimisation: Since the proposed model demonstrates a trade-off between FPR and FNR, future research should investigate decision-threshold optimisation and cost-sensitive evaluation to identify operating points suitable for different IoT security requirements.

vi. Real blockchain implementation: The conceptual blockchain mechanism should be implemented on a multi-node testbed to evaluate transaction latency, throughput, storage requirements, computational overhead, energy consumption and consensus performance.

vii. Security evaluation of the consensus mechanism: Future research should evaluate the proposed fairness-oriented validation mechanism under malicious-node, collusion, Sybil and Byzantine attack scenarios.

viii. Blockchain scalability: The effect of increasing the number of validators, transaction rates and network nodes should be experimentally investigated.

ix. Comparative consensus evaluation: The proposed weighted-voting mechanism should be compared with established consensus approaches using clearly defined performance and security criteria.

x. Model deployment optimisation: Since IoT devices may have limited computational and energy resources, future research should investigate model compression, pruning, quantisation and edge-based deployment of the proposed IDS architecture.

REFERENCES

1. Alabsi, B. A., Anbar, M., & Rihan, S. D. A. (2023). CNN-CNN: Dual Convolutional Neural Network Approach for Feature Selection and Attack Detection on Internet of Things Networks. *Sensors*, *23*(14). https://doi.org/10.3390/s23146507
2. Alkhamisi, A., Katib, I., & Buhari, S. M. (2024). Blockchain-Based Control Plane Attack Detection Mechanisms for Multi-Controller Software-Defined Networks. *Electronics (Switzerland)*, *13*(12). https://doi.org/10.3390/electronics13122279
3. Alsharif, N. A., Mishra, S., & Alshehri, M. (2023). IDS in IoT using Machine Learning and Blockchain. *Engineering, Technology and Applied Science Research*, *13*(4), 11197–11203. https://doi.org/10.48084/etasr.5992
4. Alzakari, S. A., Sarkar, A., Khan, M. Z., & Alhussan, A. A. (2024). Converging Technologies for Health Prediction and Intrusion Detection in Internet of Healthcare Things With Matrix-Valued Neural Coordinated Federated Intelligence. *IEEE Access*, *12*, 99469–99498. https://doi.org/10.1109/ACCESS.2024.3420078
5. Arul Anitha, A., & Arockiam, L. (2022). A Review on Intrusion Detection Systems to Secure IoT Networks. In *International Journal of Computer Networks and Applications* (Vol. 9, Issue 1, pp. 38–50). EverScience Publications. https://doi.org/10.22247/ijcna/2022/211599
6. Awajan, A. (2023). A Novel Deep Learning-Based Intrusion Detection System for IoT Networks. *Computers*, *12*(2). https://doi.org/10.3390/computers12020034
7. Chauhan, D., & Jain, J. K. (2025). Hybrid Deep Learning and Blockchain-Enabled Intrusion Detection System for IoT Networks using Enhanced Dataset Fusion. *Journal of Engineering Science and Technology Review*, *18*(4), 205–212. https://doi.org/10.25103/jestr.184.26
8. Dey, B., Ferdous, J., Ahmed, R., & Hossain, J. (2024). Assessing deep convolutional

neural network models and their comparative performance for automated medicinal plant identification from leaf images. *Heliyon*, *10*(1). https://doi.org/10.1016/j.heliyon.2023.e23655

9. Hu, W. W., Alzubi, J. A., Shreyas, J., Al-Razgan, M., Ali, Y. A., & Karthikayan, A. (2025). Enhancing IoT Network Security by Anomaly Detection and Intrusion Prevention Using Gannet Optimization-Based Adaptive Deep Capsule Network. *International Journal of Computational Intelligence Systems*, *18*(1). https://doi.org/10.1007/s44196-025-00940-2
10. Islam, N., Farhin, F., Sultana, I., Kaiser, S., Rahman, S., Mahmud, M., Hosen, S., & Cho, G. H. (2021). Towards Machine Learning Based Intrusion Detection in IoT Networks. *Computers, Materials and Continua*, *69*(2), 1801–1821. https://doi.org/10.32604/cmc.2021.018466
11. Kably, S., Benbarrad, T., Alaoui, N., & Arioua, M. (2023). Multi-Zone-Wise Blockchain Based Intrusion Detection and Prevention System for IoT Environment. *Computers, Materials and Continua*, *74*(1), 253–278. https://doi.org/10.32604/cmc.2023.032220
12. Kashif, M., & Kalkan, K. (2025). Differential privacy preserving based framework using blockchain for internet-of-things. *Peer-to-Peer Networking and Applications*, *18*(1), 1–23. https://doi.org/10.1007/s12083-024-01858-w
13. Khan, A. R., Kashif, M., Jhaveri, R. H., Raut, R., Saba, T., & Bahaj, S. A. (2022). Deep Learning for Intrusion Detection and Security of Internet of Things (IoT): Current Analysis, Challenges, and Possible Solutions. In *Security and Communication Networks* (Vol. 2022). Hindawi Limited. https://doi.org/10.1155/2022/4016073
14. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (n.d.). *ImageNet Classification with Deep Convolutional Neural Networks*. http://code.google.com/p/cuda-convnet/
15. Lei, C., Jiang, Y., Xu, K., Liu, S., Cao, H., & Wang, C. (2025). Convolutional Neural Network Models for Visual Classification of Pressure Ulcer Stages: Cross-Sectional Study. *JMIR Medical Informatics*, *13*. https://doi.org/10.2196/62774
16. Liu, Q., Liu, H., & Liang, Y. (2021). *Dual-Branch Network with Dual-Sampling Modulated Dice Loss for Hard Exudate Segmentation from Colour Fundus Images*. http://arxiv.org/abs/2012.01665
17. Madhu, B., Venu Gopala Chari, M., Vankdothu, R., Silivery, A. K., & Aerranagula, V. (2023). Intrusion detection models for IOT networks via deep learning approaches. *Measurement: Sensors*, *25*. https://doi.org/10.1016/j.measen.2022.100641
18. Nagarjun, A. V., & Rajkumar, S. (2025). Quantum deep learning-enhanced ethereum

blockchain for cloud security: intrusion detection, fraud prevention, and secure data migration. *Scientific Reports*, *15*(1). https://doi.org/10.1038/s41598-025-22408-1

19. Rybacki, P., Niemann, J., Derouiche, S., Chetehouna, S., Boulaares, I., Seghir, N. M., Diatta, J., & Osuch, A. (2024). Convolutional Neural Network (CNN) Model for the Classification of Varieties of Date Palm Fruits (Phoenix dactylifera L.). *Sensors*, *24*(2). https://doi.org/10.3390/s24020558
20. Saba, T., Rehman, A., Sadad, T., Kolivand, H., & Bahaj, S. A. (2022). *Anomaly-based Intrusion Detection System for IoT Networks through Deep Learning Model*.
21. Saveetha, D., Maragatham, G., Ponnusamy, V., & Zdravkovic, N. (2024). An Integrated Federated Machine Learning and Blockchain Framework With Optimal Miner Selection for Reliable DDOS Attack Detection. *IEEE Access*, *12*, 127903–127915. https://doi.org/10.1109/ACCESS.2024.3413076
22. Susilo, B., & Sari, R. F. (2020). Intrusion detection in IoT networks using deep learning algorithm. *Information (Switzerland)*, *11*(5). https://doi.org/10.3390/INFO11050279
23. Thandu, A. L., & Gera, P. (2024). Privacy-Centric Multi-Class Detection of COVID 19 Through Breathing Sounds and Chest X-Ray Images: Blockchain and Optimized Neural Networks. *IEEE Access*, *12*, 89968–89985. https://doi.org/10.1109/ACCESS.2024.3418202
24. Udurume, M., Shakhov, V., & Koo, I. (2024). Comparative Analysis of Deep Convolutional Neural Network—Bidirectional Long Short-Term Memory and Machine Learning Methods in Intrusion Detection Systems. *Applied Sciences (Switzerland)*, *14*(16). https://doi.org/10.3390/app14166967
25. Vineth Ligi, S., Kundu, S. S., Kumar, R., Narayanamoorthi, R., Lai, K. W., & Dhanalakshmi, S. (2022). Radiological Analysis of COVID-19 Using Computational Intelligence: A Broad Gauge Study. In *Journal of Healthcare Engineering* (Vol. 2022). Hindawi Limited. https://doi.org/10.1155/2022/5998042
26. Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, *57*(4). https://doi.org/10.1007/s10462-024-10721-6