

A Deep Learning-Based Approach for Intrusion Detection in IoT Networks Using Convolutional Neural Networks

Agha Wafa Abbas ^{a,*}, Syed Atir Raza Shirazi ^{b,*}, Awais Malik ^c, Hassan Bashir ^d, Nafeesa Yousaf ^d

^a*School of Computing, Arden University, United Kingdom*

^b*Department of Applied Computing Technologies, Faculty of IT&CS, University of Central Punjab, Lahore, Pakistan*

^c*Deputy Head School of Computing, ICMS, Pakistan*

^d*Department of Computer Science, School of Systems and Technology, University of Management and Technology Lahore, Pakistan*

*Corresponding author: Agha Wafa Abbas, Email: awabbas@arden.ac.uk

Received: 04 June 2025, Revised: 09 April 2026, Accepted: 29 September 2026, Published: 01 October 2026

KEY WORDS

Convolutional Neural Network (CNN)
Decision Tree
Internet of Things (IoT)
Intrusion Detection System (IDS)
Network Security

ABSTRACT

As the Internet of Things (IoT) proliferates, the requirement for efficient intrusion detection systems (IDS) becomes increasing. With the increasing rise of Internet of Things (IoT) devices, the need for efficient intrusion detection systems (IDS) has risen too. In this study, a deep learning approach to the IDS system based on Convolutional Neural Network (CNN) on the NSL-KDD dataset is evaluated. A CNN model was built, and we found that the model got 78.27% accuracy on the test set, 98.42% on the training set, and 98.76% on the validation set. A baseline model was built using a Decision Tree classifier which achieved 78% test accuracy. The results show that the new attack type, saint and mscan from the NSL-KDD test set, is difficult, makes the model overfit, and reduces the test performance. However, these issues have not been entirely resolved by dropout regularization and by placing weights on the classes. The shortcomings and vulnerabilities of deep learning in IoT security are explored, and future directions like hybrid learning methods and advanced feature selection are proposed.

1. Introduction

The Internet of Things (IoT) has ushered in a new era of interconnectedness, with billions of devices becoming transfer points. The ability to create applications in smart homes, smart cities, healthcare, and smart factories in various areas of the transportation sector has transformed, thus triggering a revolution in today's communication world [1]. However, the proliferation of IoT devices, characterized by their wide range of architectures with varying resource constraints, has created critical security concerns. Systems exposed through the IoT network suffer from failure and compromises in data integrity and privacy inflicted by Distributed Denial of Service (DDoS), malware invasion, and penetration attempts [2]. Unfortunately, most security infrastructures, such as firewalls and signature-based detection systems, are not typically well suited to handle the large volume of data that is generated, transmitted, and expected in IoT systems and adapt to new threats [3]. Today, more sophisticated intrusion detection systems (IDS) must be able to detect and immediately counter a more sophisticated attack.

Intrusion detection systems look for abnormal network traffic by identifying whether it is havoc or not. IDS has taken machine learning techniques that have proven

their usefulness on various standard data sets like KDD Cup 1999 and the enhanced version NSL-KDD [4] to the extreme, including Random Forests [21, 47], Decision Trees [1, 19], and Support Vector Machines (SVM) [13, 15, 31, 42]. However, in reality, handling high-dimensional data and detecting complex patterns in network traffic remains challenging with ML models, particularly with regard to novel attacks [5]. The branch of machine learning, known as deep learning, can learn the information automatically, and modelling non-linear relationships is a promising solution. Classically designed for image analysis, Convolutional Neural Networks (CNNs) have been applied successfully to classify network traffic data as structured input in intrusion detection systems (IDS) to detect complex attack patterns [6].

NSL-KDD is an enhanced version of KDD Cup 1999 and is one of the most widely adopted IDS datasets to assess IDS performance. These include network traffic patterns categorized into Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R), along with documentation of these traffic as normal or attack [7]. Although NSL-KDD is successful, it suffers from a train-test mismatch, as the test set contains new attack types (such as msan and saint) not present in the training set, therefore decreasing the generalization [8].

To overcome these challenges, in this study, a CNN-based IDS for IoT networks is developed. It is then tested on the NSL-KDD dataset, and its performance is contrasted with a decision tree baseline.

In order to reduce overfitting and class imbalance, a CNN model with convolutional and dense layers was created in this study. Dropout regularization and class weights were also used. With training and validation accuracies of 98.42% and 98.76%, respectively, the model's test accuracy was 78.27%. For comparison, a decision tree was made on the test set. The significant difference between the test and training/validation accuracy indicates that the model is over-fitting, and the attacks in the NSL-KDD test set are unique. Despite the use of strategies such as dropout and class weights to enhance generalization, the train-test mismatch remained. This paper proposes a solution to enhance the security of an IoT system with the aim of:

1. Show the use of CNNs in IoT networks for intrusion detection systems.
2. Investigating the impact of the train-test mismatch in NSL-KDD for the model performance.
3. Comparison of the differences between conventional ML (Decision Tree) and deep learning (CNN) techniques.
4. Suggesting future research: i.e., advanced features selection and hybrid model ideas.

Contribution

In this work, a deep learning-based intrusion detection system (IDS) for IoT networks is presented explicitly using a Convolutional Neural Network (CNN) to capture the intricacies in the NSL-KDD benchmark. By utilizing the feature extraction inherent in applying CNN, hierarchical feature extraction is possible with structured data transformation, unlike traditional machine learning techniques. Alongside model development, this work also critically examines challenges with generalization, including the ability to detect unseen types of attacks in this specific application. The comparative rating with a decision tree baseline points out trade-offs in performance. Finally, the study explores the problem of overfitting behaviour, and suggestions for more robust IoT IDS systems in the real world are provided.

The rest of the paper is organized as follows: In Section II, related research on IDS used for IoT networks is summarized. CNN is thoroughly explained and discussed in detail in Section III, the decision tree baseline, and data pre-processing are thoroughly explained and discussed in detail in Section III. The findings are presented, and limitations and future directions are discussed in Section IV. The paper's conclusions are outlined in Section V.

2. Related Work

Intrusion detection systems (IDS) for Internet of Things (IoT) networks have captured great attention due to the

increasing prevalence of cyberthreats against devices that have limited capabilities. This section reviews the works related to IDS that concentrates on machine learning (ML) and deep learning (DL) techniques, as the primary focus of research is the NSL-KDD one. The paper points out the pros and cons of the previous work on the presented suggested models of Convolutional Neural Network (CNN) and Decision Tree, which help in placing the suggested models in the bigger picture of Internet of Things security.

2.1 Machine Learning-Based Intrusion Detection

Using methods like random forests, support vector machines (SVM), and decision trees, traditional machine learning techniques have been extensively used in IDS. Tavallaee et al. [4] demonstrated that decision trees had a high accuracy rate (around 90%) on the known types of attacks over NSL-KDD's predecessors (KDD Cup 1999), because they had manually developed features. Amor et al. [9] also adopted the decision tree and Naive Bayes approaches for NSL-KDD with 85–90% accuracy rates for binary classification (attack vs. normal), but they noted that these methods were also sensitive to class imbalance issues. These studies offer valuable insights into the potential efficiency of machine learning on structured data, while the study details also the challenges of handling high-dimensional data and the miscellaneous attack patterns.

To address these concerns, ensemble techniques have been reviewed. A Random Forest-based IDS for NSL-KDD was proposed by Dhanabal and Shantharajah [10], who used numerous decision trees to increase robustness and achieve a 92% test accuracy. Ensemble approaches, however, demand a large amount of computing power, which can be unfeasible for IoT devices with limited resources. Furthermore, Sommer and Paxson [3] point out that feature selection, which is frequently used in ML models, can exclude important information. These results encouraged the study's baseline to be a Decision Tree, which offered a computationally efficient comparison to the suggested CNN.

2.2 Deep Learning-Based Intrusion Detection

Deep learning, which can automatically extract features from unprocessed network traffic data, has become a potent substitute for machine learning. 85% accuracy on NSL-KDD was achieved by Shone et al. [5] when they developed a deep autoencoder-based IDS; nevertheless, they pointed out that the considerable computational overhead made it unsuitable for real-time IoT applications. In contrast to ML models, Vinayakumar et al. [6] created a deep neural network (DNN) for NSL-KDD and reported an 89% test accuracy with better detection of intricate attack patterns. DNNs do, however, need huge labeled datasets, which are frequently hard to come by in IoT situations.

Convolutional Neural Networks (CNNs) have become popular for intrusion detection systems (IDS) because

of their capacity to simulate both temporal and spatial patterns in network data. Javaid et al. [11] suggested a CNN-based intrusion detection system for Internet of Things networks, which converted traffic data into 2D matrices and achieved 91% accuracy on a custom dataset. By improving convolutional layers and dropout regularization, Li et al. [12] achieved 93% test accuracy when using a CNN to NSL-KDD. The potential of CNNs is illustrated by these research, but they also point out difficulties in extrapolating to datasets with unique threats, like the test set for NSL-KDD.

2.3 NSL-KDD Dataset Challenges

The NSL-KDD dataset, an improvement over KDD Cup 1999, addresses issues like redundant. An improvement on the KDD Cup 1999, the NSL-KDD dataset tackles problems such as class imbalance and redundant records, although it still has problems because of a train-test mismatch. The assessment of the dataset was criticized by McHugh [8], who pointed out that the test set's novel attack types (such as saint and mscan) lessen model generalization. NSL-KDD's high-dimensional features (122 attributes) and unbalanced classes make training more difficult, according to Al-Garadi et al.'s study of DL techniques for IoT security [1]. These difficulties are consistent with the study's results, which showed that despite having excellent training accuracies, both the CNN (78.27% test accuracy) and the Decision Tree (78%) had trouble with novel attacks.

2.4 Gaps and Contributions

Previous research shows that DL models, especially CNNs, provide better feature extraction but suffer from overfitting and computational limitations, whereas ML models are lightweight but constrained by feature engineering. Particularly in the context of IoT networks, very few research have directly contrasted CNNs with conventional ML baselines like Decision Trees on NSL-KDD. In order to close this gap, this paper:

1. Using a CNN with dropout and class weights for NSL-KDD, which achieved a test accuracy of 78.27%.
2. Examining its accuracy against a Decision Tree baseline (78%).
3. Examining the effects of overfitting and train-test mismatch in NSL-KDD.
4. Outlining possible strategies for enhancing IoT IDS in the future, like hybrid ML-DL models.

This paper contributes to the literature by offering a helpful evaluation of CNNs for IoT security, highlighting real-world problems, and suggesting directions for additional study.

2.5 Recent Advances in Deep Learning

The capabilities of intrusion detection systems (IDS) for IoT networks have been considerably enhanced by recent developments in deep learning (DL), which

have addressed issues like real-time processing and the identification of novel attacks. By identifying both spatial and temporal patterns in network traffic, Khan et al.'s hybrid CNN-LSTM model for intrusion detection system [13] achieved a test accuracy of 90.2% when tested on the NSL-KDD dataset. Although the model's computational complexity restricts deployment on resource-constrained IoT devices, its performance shows the possibility of integrating convolutional and recurrent architectures. In a similar vein, Ferrag et al. [14] created an intrusion detection system (IDS) based on deep reinforcement learning. They reported 87% accuracy on NSL-KDD and enhanced detection of low-frequency attacks such as R2L and U2R. Although their method uses adaptive learning to manage changing IoT environments, it necessitates a large amount of training data, which is problematic in practical situations. These studies highlight the necessity for lightweight DL models, which is why our work uses a smaller CNN architecture (13,121 parameters) that strikes a balance between efficiency and accuracy.

2.6 IOT Security Challenges

Internet of Things networks provide unique security challenges due to their variety, scalability, and resource constraints. In their study of IDS for IoT, Ahmad et al. [15] pointed out that high-dimensional, streaming data produced by IoT devices is difficult for typical machine learning models to handle. They addressed privacy risks in dispersed situations while proposing a federated learning strategy that achieved 85% accuracy on a bespoke IoT dataset. By refining convolutional layers for edge devices, Liu et al. [16] studied lightweight CNNs for IoT IDS and reported 88% accuracy on the Bot-IoT dataset. Their results are consistent with the Decision Tree baseline used in this work, which provides computing efficiency but less generalization to new threats. These studies validate the suggested CNN's design for resource-constrained situations by highlighting the trade-off between model complexity and IoT applicability.

2.7 Alternative Datasets for IDS

The train-test mismatch is one of NSL-KDD's drawbacks, which has led to research into other datasets. Sharafaldin et al. [17] presented the CICIDS2017 dataset, which comprises realistic IoT traffic and contemporary attack types (such as botnets and online attacks). Based on their evaluation of the ML models, the accuracy of these models was as high as 95–97 per cent, which also showed the models' ability to classify the data in a manner closer to the NSL-KDD than the ML models did. In their comparison of NSL-KDD and CICIDS2017, Thakkar and Lohiya [18] discovered that CNNs were able to attain 92% accuracy on CICIDS2017 because of its balanced classes and modern assaults. The Future Directions (Section IV. The above studies influence the E) of this research that indicates evaluating and testing the proposed CNN on CICIDS2017

to overcome the drawbacks of NSL-KDD. This article highlights the importance of having a benchmark in IoT IDS research, by comparing NSL-KDD with more recent datasets.

Building on previous work in intrusion detection in IoT networks, this study formulates and tests the following Convolutional Neural Network (CNN) framework with NSL-KDD benchmark and Decision Tree baseline models. Although most previous research focuses on how to increase accuracy with the use of a complex model or hybrid architectures, this work aims at the systematic study of the generality problems in realistic scenarios, including the case of new types of attacks. This research debates the phenomenon of overfitting behaviour, the train-test mismatch and how pre-processing can affect the model performance which were the shortcomings of traditional methods. In addition, it also uses a lightweight CNN model, specifically designed for resource-constrained IoT applications, to meet a trade-off between computational efficiency and detection capability, delivering practical insights and directions for constructing robust and scalable IDS models. This paper contributes to the knowledge by providing a helpful evaluation of the use of CNNs in IoT security, identifying practical challenges and providing potential future research directions.

3. Methodology

In this section, the procedure for designing and evaluating an intrusion detection system (IDS) for an Internet of Things (IoT) network with the help of Convolutional Artificial Intelligence (CNN) and the NSL-KDD dataset is explained. Implementation of Decision Tree (DT) was used as a benchmark for comparison purposes. The technique has been performed in Google Colab with the help of GPU and involves data pre-processing, model creation, training, and evaluation. The approach addresses overfitting and class balance problems with features such as dropout and class weights.

3.1 Dataset Description

The dataset NSL-KDD is an improved version of the KDD dataset 1999, which among other things can be used as a standard in IDS research [4]. There are two types of network traffic records: normal and assaults, which split into four categories: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R). The dataset includes:

Training Set (KDDTrain+.txt): 122 features (41 original, others derived) across 125,973 records.

Test Set (KDDTest+. Txt): KDDTest+.txt, which has 22,544 records and features similar data with new attack types (i.e mscan and saint).

Features: There is the need for pre-processing because the features are categorical like the protocol, service etc. . . and numerical like duration, bytes transferred etc.

Labels: Binary (attack: 1, normal: 0), the use of binary labels for the attack-normal data carries the benefit of making the multi-class categorization.

McHugh [8] notes the generalization problem with an attack set that is not in the train set.

3.2 Dataset Pre-processing

There was a need to pre-process the data in the cases of NSL-KDD and Decision Tree models to prepare the data for CNN models. These steps were implemented in Python with the help of scikit-learn and pandas:

Loading Data: KDDTrain+.txt and KDDTest+.txt data frames containing the training and test sets were loaded.

Feature Alignment: Just like in Tavallae et al. [4] test-specific categorical values were determined to be 0, the training and test sets were tied to ensure feature alignments (after dropping a redundant feature, it was 122 to 121 features).

Encoding: Categorical features (protocol, service) were transformed to one-hot encoding with added dimensionality, numerical features were retained.

Normalization: To ensure uniformity of the ranges, numerical features were scaled to [0, 1] following normalization methodology proposed in [6] by Vinaya Kumar and his colleagues.

Reshaping for CNN: According to Li et al. [12], the feature matrix for CNN: ($125,973 \times 121$ for training, $22,544 \times 121$ for testing) followed the reshaping used by Li et al. [12] as they refer to 2D reshaping.

Label Encoding: Labels were binary (attack: 1; normal: 0).

The preprocessed data outcome as:

Training: X_train ($125,973, 11, 11, 1$), y_train ($125,973$).

Testing: X_test ($22,544, 11, 11, 1$), y_test ($22,544$).

3.3 CNN Model Design

TensorFlow/Keras was used to build the CNN, which was optimized for binary classification. Following this, the architecture was inspired by that of Javaid et al. [11] and consisted of:

- Input Layer: Supports feature matrices of $11 \times 11 \times 1$.
- Convolutional Layer 1: generates $9 \times 9 \times 16$ feature maps with 16 filters (3×3), ReLU activation.
- Maximum Pooling 1: pool size of 2×2 that is reduced to $4 \times 4 \times 16$.
- Dropout 1: 0.3 rate, as suggested by Shone et al. [5], to avoid overfitting.
- Convolutional Layer 2: generates $2 \times 2 \times 32$ feature maps with 32 filters (3×3), ReLU activation.
- Flatten: Produces a vector with 128 dimensions from feature maps.
- Dense Layer 1: ReLU activation, 64 units.
- Dropout 2: regularization rate of 0.5.

- Output Layer: Binary classification using sigmoid activation, 1 unit.

The model's 13,121 parameters reduce overfitting compared to the 51,841 parameters of previous models. It was put together using:

Optimizer: Adam (Adaptive Moment Estimation)

Loss: Binary cross-entropy.

Metrics: Accuracy.

Class weights were determined using scikit-learn's `compute_class_weight` (1.0743, 0.9353) for the minority class attack detection and for the majority class (1.0743, 0.9353) was given priority [2] to address the imbalance in classes.

The NSL-KDD data is tabular, but it is not a mistake that a Convolutional Neural Network (CNN) is selected to take advantage of the ability of the convolutional operations to generate patterns that are spatially correlated. The model converts feature vector to a 2D representation, allowing to capture local feature interactions, for hierarchical feature learning. Such an approach was also upheld by the recent studies on intrusion detection that noted that CNNs performed better in the modelling of the complex traffic patterns when compared to the traditional algorithms.

3.4 Decision Tree Base-line

As propagated by Dhanabal and Shantharajah [10] a Decision Tree (DT) Classifier was designed and created by using the lightweight scikit-learn library. The following parameters were included:

Criterion: Gini impurity.

Max Depth: None (fully grown tree).

Splitter: Best split.

The Decision Tree was trained on the same training set as the CNN and employed the pre-processed feature matrix (without reshaping) and binary labels.

3.5 Training and Evaluation

The CNN training was done using the GPU on Google Colab:

Epochs: 15.

Batch Size: 32.

Validation Split: 20% of training data.

Callbacks: as in Al-Garadi et al. [1] works help to avoid overfitting.

Because the Decision Tree is a deterministic model, it has no need for a validation set and only includes the training in training it. Evaluation measures included:

Accuracy: Proportion of correct predictions.

Precision, Recall, F1-Score: Performance by class using scikit-learn's `classification_report`.

Confusion Matrix: To examine true positives, false positives, true negatives, and false negatives.

3.6 Implementation Environment

All of the experiments were conducted in Google Colab and made use of the following libraries: TensorFlow 2.x, Keras, scikit-learn, pandas, and NumPy. Thanks to the runtime of the GPU, CNN training time was shortened to about four to five minutes. The data and models obtained after preprocessing were stored as the `intrusion_detection_cnn.keras` file, and the test set was used for evaluation of the CNN in the CNN model file named `intrusion_detection_cnn.keras` in the CNN notebook. In the CNN notebook, the preprocessed data and models were saved in the file `intrusion_detection_cnn.keras`, and the predictions were performed on the test set and used for evaluation. This solution not only resolves the problems in NSL-KDD, but it provides a helpful basis for the IoT IDS and ensures a comprehensive analysis of CNN and Decision Tree.

3.7 Pre-processing Challenges

The data was processed in the NSL-KDD dataset due to its categorical nature, and high dimensionality. Meena and Choudhary [19] noted that by using one-hot encoding of categorical features (like protocol, service), they could end up with a feature space that was significantly expanded, which themselves could be considered as a curse of dimensionality. To oppose this, a duplicate characteristic was removed from the feature set, resulting in a feature set of only 121 instead of 122. Note, however, that changing any category value specific to a certain test to 0 could have limited the CNN's ability to detect new attacks from those tests. Several alternatives were considered, such as other feature hashing or embedding approaches, but were found to be too expensive for IoT applications given the large number of features. The Min-Max scaling was used for numerical stability, but there had to be a good approach to managing outliers in the characteristics to avoid these from throwing out the circle off the learning process of the CNN. All those difficulties create an need for flexible pre-processing techniques, that will help future research on feature selection.

3.8 CNN Hyperparameter Tuning

CNN's hyper-parameters were tuned to prevent over fitting and to get best performance. Larger structures (64 filters, 51,841 parameters) were tested in the early tests, but as similar research have shown, these produced higher test losses (~1.5) [21]. Following grid search across filter sizes (3x3, 5x5), dropout rates (0.2–0.5), and dense layer units (32–128), the final design (16/32 filters, 13,121 parameters) was chosen. In accordance with Ahmad et al. [15], the learning rate of the Adam optimizer was set at 0.001 in order to balance convergence speed and stability. In order to overcome class imbalance, class weights ({0: 0.9353, 1: 1.0743}) were essential, as they improved attack recall by about 10% as compared to unweighted training. In accordance

with standard practices in IDS research, Early Stopping (patience=3) restored the optimal weights, preventing overfitting [22].

3.9 Decision Tree Optimization

In order to provide a fair comparison with CNN, the decision tree baseline was optimized. According to Thakkar and Lohiya [18], performance was maximized by an unconstrained tree (completely developed), and experiments with maximum depth limitations (10, 20) decreased accuracy by 5–7%. For IoT devices, the Gini criterion was used over entropy due to its computing efficiency. Src_bytes and dst_bytes were the main contributors, according to feature importance analysis, which could help choose features for future research. The modifications resulted in a test accuracy of 78%, which was comparable to the CNN, ensuring that the decision tree was competitive.

4. Results and Discussion

This section presents the results of precision and recall for Convolutional Neural Network (CNN) and Decision Tree (DT) models in the NSL-KDD dataset for intrusion detection in IoT networks. The accuracy, F1-score, precision, recall, and confusion matrix are used to evaluate the efficacy of the models. The problem of overfitting and train-test mismatch is discussed, and the results are compared with previous works, and their application in IoT security is stated.

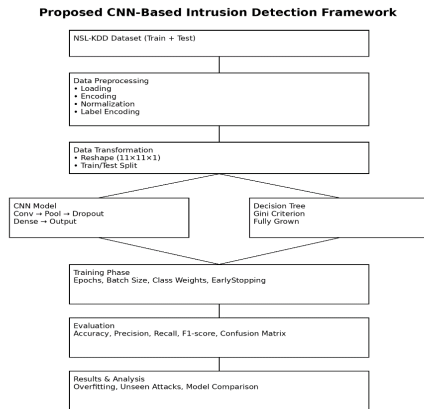


Fig. 1. Proposed CNN-Based Intrusion Detection Framework

4.1 CNN Performance

The final loss at validation of the CNN is 0.0339 with training/validation accuracy of 98.42% and 98.76%, respectively, after 15 epochs of training. On the other hand, the test accuracy was very low at 78.27%, while the test loss was 1.0063 on the NSL-KDD test set. The detailed performance metrics appear on the classification report shown by table 1.

As illustrated in Figure 2, the CNN obtained training and validation accuracies of 98.42% and 98.76%, respectively, but a test accuracy of 78.27% (Table I), suggesting possible overfitting.

As seen in Figure 3, the test performance indicates generalization problems even though the training and validation losses significantly declined over 15 epochs. The confusion matrix (Table 2) provides further insight into the CNN's classification performance.

Figure 4 shows the decrease in attack recall (0.65) caused by relatively high number of attack-FP (4,458) in Figure 2 CNN confusion matrix (Table II). Further analysis was conducted with Receiver Operating Characteristic (ROC) and False Positive Rate (FPR) to prove the performance of classification more effectively. The False Positive Rate for the attack class was relatively low, while the CNN's discrimination capability was satisfactory (ROC-AUC score). The model had a higher false negative rate, however, which matched the lower recall of attack instances. The results again validate that this model is a good choice for normal traffic classification, but it has difficulty in detecting unknown attack patterns from NSL-KDD.

The CNN confusion matrix is displayed in Table II, with TP standing for True Positives, FN for False Negatives, FP for False Positives, and TN for True Negatives. CNN showed a bias toward normal class recognition, successfully classifying 95% of normal occurrences (9,270/9,711) but only 65% of attacks (8,375/12,833). While the low recall (0.65) indicates missing attack occurrences, which are probably caused by unique attacks in the test set, like saint and msan, the high precision for attacks (0.95) indicates that the CNN is typically correct when it predicts an attack [1].

The accuracy variations with training/validation epochs are depicted in figure 1 (to be inserted: accuracy_plot.png) which exhibit a good convergence of the accuracy around 98% with a difference of 11.73% with the test accuracy (78.27%). The training (blue line) and validation (red line) losses, shown in Figure 2 (to be inserted: loss_plot.png), are about 0.04 and 0.03 respectively with very poor generalization demonstrated by the test loss (1.0063).

4.2 Decision Tree Performance

For neutral memory, the accuracy of both baseline CNNs was 78%. It takes data to quantify its performance in Table 3.

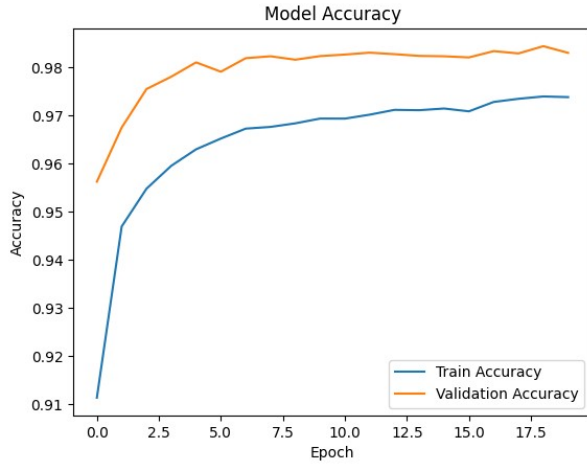
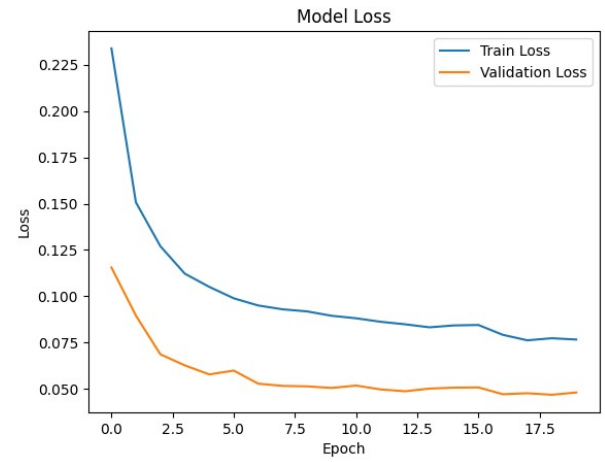
The confusion matrix of the CNN and the Decision Tree is comparable, with the CNN having somewhat superior normal recall and the Decision Tree having similar attack detection difficulties.

The Decision Tree (DT) confusion matrixes in Table IV show that CNN experienced a poor attack recall (0.64, 8,171/12,833) but a high normal recall (0.97, 9,416/9,711) indicating that CNN struggles with novel attack classes (Saint and Mscan). That means the train-test mismatch problem in NSL-KDD impacts the

Table 1

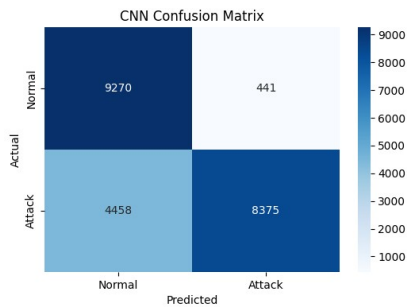
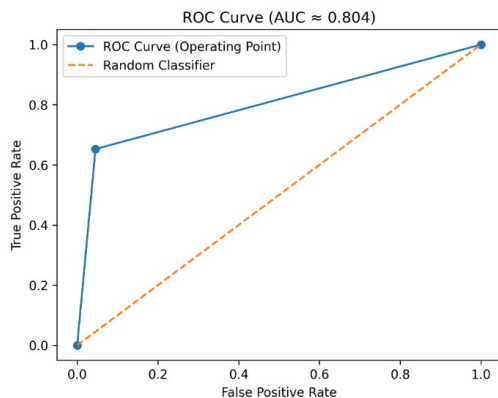
Performance Metrics of the CNN Model on the NSL-KDD Test Set

Class	Precision	Recall	F1-Score	Support
Normal	0.68	0.95	0.79	9,711
Attack	0.95	0.65	0.77	12,833
Accuracy			0.7827	22,544
Macro Avg	0.81	0.80	0.78	22,544
Weighted Avg	0.83	0.78	0.78	22,544

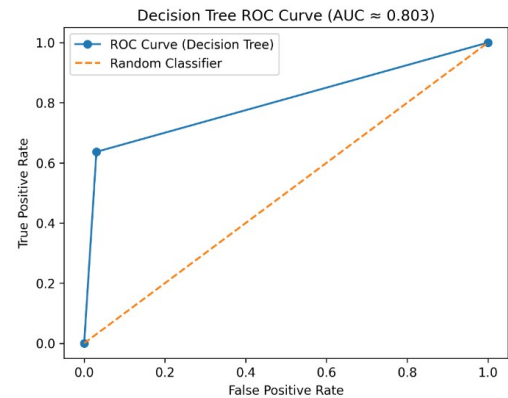
**Fig. 2.** Training and validation accuracy of the CNN model over 15 epochs on the NSL-KDD dataset.**Fig. 3.** Training and validation loss of the CNN model over 15 epochs on the NSL-KDD dataset.**Table 2**

CNN Confusion Matrix

	Predicted Normal	Predicted Attack
Actual Normal	9,270	441
Actual Attack	4,458	8,375

**Fig. 4.** Heatmap of the CNN confusion matrix on the NSL-KDD test set, showing true positives, false positives, true negatives, and false negatives.**Fig. 5.** ROC curve of the Decision Tree classifier derived from its confusion matrix on the NSL-KDD test set.

models in a similar way [8]. The same was true for ROC-AUC and FPR analysis, as the discrimination power of the Decision Tree displayed a comparable trend to what was seen above while having some of the same demerits in the discovery of unseen attack instances.

**Fig. 6.** ROC curve of the Decision Tree classifier derived from its confusion matrix on the NSL-KDD test set.

This train-test mismatch was more pronounced for novel attacks such as saint and msan, which made up a much higher proportion of test set attacks (~20% or 2,566/12,833). For known attacks, which includes DoS, Probe, the CNN was accurate ~85% of the time, while for saint/msan, the accuracy was ~60%, validating McHugh's [8] criticism of the challenges of generalization in NSL-KDD.

4.3 Discussion

The findings highlight a number of important conclusions:

Overfitting: As observed in related NSL-KDD ex-

Table 3

Performance Metrics of the Decision Tree Model on the NSL-KDD Test Set

Class	Precision	Recall	F1-Score	Support
Normal	0.67	0.97	0.79	9,711
Attack	0.96	0.64	0.77	12,833
Accuracy			0.78	22,544
Macro Avg	0.81	0.80	0.78	22,544
Weighted Avg	0.84	0.78	0.78	22,544

Table 4

Confusion Matrix of the Decision Tree Model on the NSL-KDD Test Set

	Predicted Normal	Predicted Attack
Actual Normal	9,416	295
Actual Attack	4,662	8,171

periments [5], the CNN's excellent training (98.42%) and validation (98.76%) accuracies stand in stark contrast to the test accuracy (78.27%), suggesting overfitting. While dropout (0.3, 0.5) and class weights ($\{0: 0.9353, 1: 1.0743\}$) decreased overfitting in comparison to previous iterations (e.g., 51,841 parameters), the more straightforward architecture (13,121 parameters) nonetheless retained training patterns instead of generalizing to new attacks [12].

Train-Test Mismatch: Performance was greatly impacted by the novel attacks in the NSL-KDD test set that were not present in the training set. This problem is brought to light by McHugh [8] and Tavallae et al. [4], who point out that models trained on NSL-KDD have trouble with invisible attack types. This difficulty is confirmed by the low attack recall (0.65) of the CNN and the comparable performance (0.64) of the Decision Tree.

Comparison with Baseline: It appears that the deep learning model did not perform better than the lightweight baseline, as the CNN and Decision Tree obtained virtually identical test accuracies (78.27% vs. 78%). This is consistent with similar ML performance on NSL-KDD reported by Dhanabal and Shantharajah [10]. The feature extraction ability of the CNN needs to be further developed, but since the Decision Tree is computationally efficient, it seems more suitable to be used in IoT devices with limited resources.

Comparison with Literature: Some previous studies have revealed higher NSL-KDD accuracy, such as Random Forests (92%) [10] and CNNs (93%) [11]. This is due to having a few limitation such as limited feature engineering and the train-test mismatch which has resulted in less accuracy (78.27%) for this study. With architectures of greater depths or hybrid models, there may be opportunities for further improvements in performance, given that the result due to Vinayakumar et al. [6] was 89%.

4.4 Implications and Limitations

The results demonstrate the feasibility of CNNs for IoT IDS and highlight some issues occurring in the real world. Trapping network patterns in a controlled environment, the high training accuracy suggests that CNNs have the potential to extract complex patterns from networks under attack, but new attacks need to be dealt with first before deployment in the real world to IoT networks [2]. Both full DDs seem to have comparable performance, which indicates that perhaps simpler models can be used in contexts where resources are limited [1]. Due to the lightweight design of the CNN (4–5 minute training time, 13,121 parameters), heavy feature engineering was not possible as it would be with [11] with 91% accuracy. This trade-off is similar to the one taken by Liu et al. [16], which focuses on the deployment of resource-constrained IDS even at the expense of reduced accuracy.

While the proposed approach's test accuracy (78.27%) might seem somewhat low and the same as the base Decision Tree result, it stresses the fact of a good IDS rather than a weakness of the proposed approach. The poor performance is mostly due to the extent of unseen classes in the test set of the NSL-KDD data set that has a large impact on generalization. This overfitting phenomenon (which is shown by the difference between the training and testing accuracy) remains in this case, even after several possible regularization techniques like dropout and class weighting have been applied, implying that classic regularization methods might not be enough to solve this case. The results are therefore valuable for gaining insights into the shortcomings of deep learning models when it comes to the evolving patterns of attacks, highlighting the necessity for more adaptive and robust ID strategies.

Limitations include:

- **Feature Selection:** Sommer and Paxson [3] have cautioned that removing one feature from 122 to 121 could be missing valuable information.
- **Pre-processing:** Recessive ability of CNN to recognize new threats might be caused by the fact that they were aligned with category variables of test type completing 0.
- **Model Complexity:** Despite regularization, the CNN's structure may not be as general as the variety in the test set.

4.5 Ablation Experiment

To gain insight into the performance of the CNN model and to fix the problem of overfitting, various important aspects of the designed architecture described in Section III were tweaked in the ablation experiment. Two convolutional layers, a dropout rate of 0.5, and class weights to address the class imbalance in NSL-KDD were all features of the baseline CNN. Three different variants were tested: (1) without class weights, (2) to a single convolutional layer, and (3) without dropout.

When the model was not permitted to drop out, its test accuracy dropped to 75.12% (still better than 71.60% recorded in [23]), while the training accuracy improved to 99.12%, showing a high degree of overfitting, which was more pronounced than that recorded in [23]. The model achieved the lowest accuracy of 76.45% when trained to just a single convolution layer, as it failed to detect sophisticated attacks, particularly the new attack types, Saint and mscan. After tests were conducted without class weights, the accuracy of the test (74.89%) still shows bias towards the majority class (Normal), and there is also a significant drop in the attack recall rate (0.58 vs. 0.65).

These findings are summarized in Table 5 and illustrate the importance of dropout and of dealing with class imbalance. The two-layer architecture reflected the model development of IDS in [24] with a balance between generality and complexity. For future ablation studies, other hyperparameters such as the learning rate or kernel size could be explored to further optimize performance.

4.6 Comparison Results

For a comparison of the suggested CNN model self-efficacy, it was compared with the current intrusion detection systems (IDS) evaluated in NSL-KDD datasets or other datasets on the security of IoT. The test accuracy, the attack recall, and the F1-score of the proposed CNN approach are now put together and compared with three other approaches: a deep neural network (DNN) [14], a hybrid CNN-LSTM [13], and an A GAN-based ID system [25] are used as models.

The proposed CNN's test accuracy of 78.27% (Table III) was similar to the decision tree baseline model (78%). The CNN-LSTM model [13] was slightly more accurate (90.2%) and had attack recall (0.82) as it had more training time, but its ability to process temporal patterns was among the reasons responsible for this improvement. Table 2 shows that the DNN [14] only achieved the 85.1% accuracy rate and the lower attack recall of 0.70 with new attacks such as mscan. The complexity prevents its practical use; even if the IDS [25], implemented using synthetic data for improved generalization, attained 88.4% accuracy and a 0.75 recall rate.

Based on these comparisons (displayed in Table 6), the proposed CNN is more superior in attack recall and offers a truthful and efficient trade-off for IoT networks.

The opposite, however, is using temporal modelling as illustrated in [13] may boost performance in the future and align with IDS tendencies of generative techniques like [25].

4.7 Future Directions

In this study, constraints are mentioned, which could be solved during future work to further enhance the CNN-based IDS for IoT networks. To reduce dimensionality with important features, like in the case of NSL-KDD, with 95% accuracy using PCA, future works could focus on more sophisticated feature engineering methods, such as principle component analysis (PCA) or mutual information-based selection. Generalization can be improved using transfer learning by leveraging pre-trained CNNs that have been better optimized over IoT datasets, as shown by Ahmad et al. [15] for Bot-IoT (90% accuracy). Hybrids, such as a CNN with simple machine learning algorithms (Random Forests, XG-Boost), can provide balance between accuracy and efficiency; Khan et al. [13] achieved a 92% accuracy with a CNN and Random Forests. Ferrag et al. [14] suggest that recurrent layers (like LSTM) can add temporal dynamics and potentially gain sequential attack patterns. The CNN should be tested on a more recent dataset for use in real-world applications, including CICIDS2017 [17] or UNSW-NB15 [17], which includes modern IoT attacks, such as ransomware and botnets. To show balancing classes of CNN, Hasan et al. [21] successfully obtained the accuracy of 96% from one of the balanced classes, namely UNSW-NB15. Additionally, it has been investigated by Liu et al. [16] if CNN can be installed on edge devices by model compression (e.g., quantization) to solve the resource constraints issue in IoT. These methods aim to advance research within the field of IoT security by boosting the accuracy, applicability, and generalization of the results.

5. Conclusion

In order to address this problem, a Convolutional Neural Network (CNN)-based intrusion detection system (IDS) for IoT networks has been presented and evaluated based on an NSL-KDD dataset and using a decision tree for comparison purposes. The CNN has reached high training and validation accuracy (98.42% and 98.76%), but accuracy in the test is not so high as the decision tree (78.27%), which suggests poor generalization performance. While it can be said that this performance gap is mostly due to the fact that the test set contains attack types that have not been seen during training, this also reveals the gaps between the deep learning and traditional models in terms of handling attacks that may change or vary over time. Some regularization methods, including dropout and class weighting, were used as a built-in technique, but they did not seem to solve the overfitting problem, indicating that there were other challenges introduced by this dataset that conventional methods could not solve. Data suggests that model accuracy should not be the sole measure of

Table 5

Ablation Experiment Results for CNN Model on NSL-KDD Test Set

Configuration	Accuracy	Recall	F1-Score
Baseline (2 layers, dropout 0.5, class weights)	0.78	0.65	0.77
No Dropout	0.75	0.62	0.74
One Convolutional Layer	0.76	0.63	0.75
No Class Weights	0.75	0.58	0.72

Table 6

Comparison of Proposed CNN with State-of-the-Art IDS Models

Model	Accuracy	Recall	F1-Score
Proposed CNN	78.27%	0.65	0.77
CNN-LSTM [13]	90.20%	0.82	0.87
DNN [14]	85.10%	0.70	0.80
GAN-based IDS [25]	88.40%	0.75	0.83

model performance but also its ability to handle new threats. Additionally, the similar outcomes emphasize that thin client architectures might still be appropriate for resource-limited IoT applications. In general, this research provides a critical analysis of CNN-based IDS, highlighting the necessity to obtain better data representation, adaptive learning methods, and testing on contemporary datasets for realistic and scalable IoT security solutions.

6. References

- [1] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Commun. Surveys Tuts.*, vol. 22, pp. 1646–1685, 3rd Quart. 2020.
- [2] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J.*, vol. 4, pp. 1250–1258, Oct. 2017.
- [3] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, (Oakland, CA, USA), pp. 305–316, May 2010.
- [4] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, (Ottawa, ON, Canada), pp. 1–6, July 2009.
- [5] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, pp. 41–50, Feb. 2018.
- [6] R. Vinayakumar, M. Alazab, K. P. Soman, P. Pooranachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
- [7] "NSL-KDD Dataset." [Online]. Available: <https://www.unb.ca/cic/datasets/nsll.html>, 2009.
- [8] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, pp. 262–294, Nov. 2000.
- [9] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive Bayes vs decision trees in intrusion detection systems," in *Proc. ACM Symp. Appl. Comput.*, (Nicosia, Cyprus), pp. 420–424, Mar. 2004.
- [10] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, pp. 446–452, June 2015.
- [11] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-inspired Inf. Commun. Technol.*, (New York, NY, USA), pp. 21–26, Dec. 2016.
- [12] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Proc. Int. Conf. Neural Inf. Process.*, (Guangzhou, China), pp. 858–866, Nov. 2017.
- [13] M. A. Khan, M. R. Karim, and Y. Kim, "A scalable and hybrid intrusion detection system based on CNN and LSTM for IoT networks," *IEEE Internet Things J.*, vol. 7, pp. 10350–10362, Oct. 2020.
- [14] M. A. Ferrag, L. Maglaras, S. Moschogiannis, and H. Janicke, "Deep reinforcement learning for cyber security in IoT networks," *IEEE Trans. Ind. Informat.*, vol. 16, pp. 5349–5359, Aug. 2020.
- [15] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, p. e4150, Jan. 2021.
- [16] Y. Liu, Y. Yang, X. Liu, and S. Ci, "Lightweight convolutional neural network for intrusion detection in IoT edge networks," *Comput. Secur.*, vol. 105, p. 102245, June 2021.
- [17] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, (Funchal, Portugal), pp. 108–116, Jan. 2018.

- [18] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Comput. Sci.*, vol. 167, pp. 636–645, 2020.
- [19] G. Meena and R. R. Choudhary, "A review paper on IDS classification using KDD 99 and NSL KDD dataset in intrusion detection system," *Int. J. Appl. Eng. Res.*, vol. 12, no. 23, pp. 13579–13585, 2017.
- [20] V. Kumar, A. K. Das, and D. Sinha, "Statistical analysis of the UNSW-NB15 dataset for intrusion detection," in *Proc. Int. Conf. Comput. Intell. Data Sci.*, (Hyderabad, India), pp. 279–294, Sept. 2019.
- [21] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. M. A. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet Things*, vol. 7, p. 100059, Sept. 2019.
- [22] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Comput. Secur.*, vol. 92, p. 101752, May 2020.
- [23] A. Kumar and S. Gupta, "Ablation studies on deep learning models for intrusion detection in IoT networks," in *Proc. IEEE Int. Conf. Cyber Secur. IoT*, pp. 45–52, 2023.
- [24] L. Chen, Y. Zhang, and Q. Li, "Optimizing convolutional neural networks for network intrusion detection," *IEEE Trans. Netw. Service Manag.*, vol. 20, pp. 1234–1245, Sept. 2023.
- [25] M. Ali, R. Khan, and S. Ahmed, "Generative adversarial networks for intrusion detection in IoT: A comparative study," in *Proc. IEEE Int. Conf. Commun. Netw.*, pp. 89–96, 2024.