Ingénierie des Systèmes d'Information
Vol. 30, No. 3, March, 2025, pp. 753-762
Journal homepage: http://iieta.org/journals/isi

Elevating Intrusion Detection: A CNN Approach with Pre-Processing Enhancements

Anil Patidar, Kailash Chandra Bandhu*, Ratnesh Litoriya

Department of Computer Science and Engineering, Medicaps University, Indore 453331, India

Corresponding Author Email: kailashchandra.bandhu@gmail.com

https://doi.org/10.18280/isi.300319

Received: 15 December 2024
Revised: 21 January 2025
Accepted: 14 March 2025
Available online: 31 March 2025

Keywords:
CNN, normalization method, pre-processing, intrusion detection, IoT

ABSTRACT

The Internet of Things has significantly improved in different industries, although the enormous scale and complexity of IoT systems have faced security risks. To solve this problem, intrusion detection systems (IDS) are utilized, which guarantee the security state of IoT and protect it from cyber threats efficiently. In the current era, machine learning (ML) methodologies have been widely employed for IDS in IoT systems. Still, more work needs to be done, particularly when handling the functional and physical diversity of IoT devices. This work proposed a model that integrates with the normalization steps with Three dense layers CNN. Furthermore, this study employs the two dataset which are UNSW-NB15, CIC-IDS- 2017. The final selection of model is test on 24 different setting. These steps are essential to improve the quality of input data and to highlight the effectiveness of IDS. The proposed model is evaluated using accuracy, F1 score, precision, recall, RMSE, training accuracy, AUC score, TPR, FPR, specificity, sensitivity, training time and memory usage.

1. INTRODUCTION

One of the main problems with the communication in recent days is the IoT, because of many benefits and applications of IoT, various researchers have currently sought to overcome the obstacles in this area in an effort to attribute the required basis for the technology's rapid growth in real world [1]. Security is one of the primary problems with IoT applications, because new attack types are emerging intrusion techniques as threats are grown [2]. Since, the IDS frequently employs pattern analysis to identify attacks as it is thought to be one of the most effective security solutions for networking system [3-5].

Some researchers have employed ML approaches to minimize false alerts in IDS. Based on the characteristics of each attack, an IDS identify attacks more precisely by adopting the ML topology [6]. The traditional approaches of ML, including Decision Trees (DT) [7], Support Vector Machines (SVMs) [8], and k-Nearest Neighbors (k-NN) [9], for improving the detection accuracy of intrusion but these each approaches have its advantages and limitations. The foremost objective of utilizing DT is to generate a training model that employed to forecast the target variable class by applying decision rules obtained from the training data and classify the IDS efficiently [10].

However, it overfit to training data that leads to worsen the performance of Intrusion detection [11]. Because of strong generalization abilities and capacity to determine patterns, SVM have gained popularity for IDS as it overcomes the dimensionality issue. Nevertheless, this approach face challenges to apply large training sets owing to its higher computational time and space consumption and the primary difficulties with SVM falsehood in determining pertinent features for anomaly identification while handling high computational overhead.

K-NN is recognized for its simplicity and adaptability. The algorithm's non-parametric nature allows it to classify data based on proximity to labelled instances, which advantageous for dynamic environments where patterns change frequently. Despite this, k-NN's reliance on distance calculations and leads to inefficiencies and increased processing time as the volume of data scales up [12, 13].

To address the limitations of these traditional methods and meet the growing demands of IoT security, it is crucial to explore advanced DL approaches [14-22] that enhance detection accuracy and efficiency [23].

This paper tackles three questions. The first question is, is the minimal number of layers of CNN sufficient to achieve good accuracy? The second question is, does a different pre-processing phase result in different accuracy? Third question is, does minimum number of layer achieved generalization?

For the first query, the minimal number of dense values with different hyperparameters setting to check the accuracy with two datasets [24, 25].

In response to the second question, it compares the accuracy of the different models by comparing the three pre-processing phases of each CNN which is most commonly used. For third query it will compare the result of different accuracy measure to check for generality.

The rest of the paper is organized into different sections. The background work is covered in Section 2. Section 3 covers motivations. A more detailed analysis of the model is covered in the methodology Section 4. Result and discussion are covered in Section 5. The conclusion and future work are given in Section 6.

2. BACKGROUND WORK

This section explores the what other author have done in crucial pre-processing step and the Convolutional Neural Network (CNN) architecture. This will provide insights into how the IDS is enhanced and how it can effectively identify various threats by implementing machine learning and deep learning models. Table 1 shows the various author implementation details including advantages and disadvantages.

Table 1. Background works

S. No.	Author	Advantages	Disadvantages	Accuracy
1.	Zhao et al. [1]	This paper introduces with Lightweight model that reduces complexity using PCA. Author uses classifier for expansion and compression, inverse residual structure to extract features efficiently without needing much computation.	The PCA may lead to loss of critical feature as it belongs dimensionality reduction and which limit to robustness evaluation for different intrusion types.	
2.	Saba et al. [2]	Introduces a CNN-based method for anomaly detection in IDS, which uses IoT's capabilities to efficiently monitor all network traffic. The model is designed to detect potential intrusions and unusual traffic behaviour.	Optimize hyper parameters in deep learning that connect to model performance.	99.51% on NID, BOT-IOT on 92.85% dataset
3.	Liu et al. [3]	Author combines PSO and Light GBM for effective feature selection and for classification. Additionally, this model detects Backdoor, Shellcode, Worms attacks.	High computational cost and timing.	Backdoor rate is 51.28, where Shellcode rates is 64.47% and 77.78% frowarm detection
4.	Abbas et al. [4]	Author combines logistic regression, naive Bayes, and decision tree classifiers using a voting mechanism for intrusion detection.	None.	99.67%
5.	Musleh et al. [11]	Systematic evaluation of feature extraction techniques using VGG-16 and stacking model.	High computational cost.	98.3%
6.	El-Sayed et al. [15]	Author proposes a new approach called PCAP which compare seven Algorithm that split into two categories: CNN-based models (Two-Layer CNN, Four-Layer CNN, VGG16) and standard classifiers (Logistic Regression, Support Vector Machine, K-Nearest Neighbours Creative image-based malware detection.	Overhead from PCAP-to-RGB conversion.	94%
7.	Liu et al. [26]	The paper proposed an IDS Algorithm located on network layer of IOT, that use the BPSO Algorithm to extract feature from the NSL KDD dataset.	The disadvantage of anomaly detection of the whole network layer is missing.	82.9%

3. MOTIVATION

After thoroughly reviewing the paper, it was noted that the author employed various machine learning algorithms. However, the most approach behind with respect the optimization using minimum number of layers that can lead to better result and take less memory. Furthermore, high computational cost of some model makes less practical to implement for others due to resources constrained, necessitating the design of an efficient CNN architecture. Lastly, many works are limited on the basis of evolution matrix, which may not provide the in-depth analysis. To address these gaps the propose method solve the computational time, memory, and choose the better preprocessing phase to get more generalize result. In this work, the proposed model is evaluated using confusion matrix score, RMSE, training score, training time, memory usage, computation time, FPR and TPR.

4. METHODOLOGY

The primary objective of this research is to explore the impact of preprocessing techniques on the accuracy of different Internet of Things (IoT) datasets, mainly when using a minimal number of layers in a Convolution Neural Network (CNN). The research model comprises the following detailed subsections.

4.1 Dataset description

To show the proposed method's importance, this research uses two of the most common datasets, CIC-IDS- 2017 [24] and UNSW-NB15 [25]. The first data set contains 82333 data and 45 features and second having 191033 rows × 79 columns. which are used to evaluate the proposed model and show how this affects its performance. This research uses the two datasets to evaluate the model's performance and compare it further.

4.2 Proposed model component

The proposed model's primary components involve preprocessing techniques such as one-hot encoding and data normalization. One-hot encoding represents categorical data in a binary format, while data normalization is employed to scale the numerical features. After these preprocessing steps, the output of data normalization is further processed using the train test split function which break into training dataset and testing dataset, facilitating the application of convolutional neural networks (CNN). Finally, deep CNN is utilized to analyse the dataset, generating both an accuracy matrix and a confusion matrix different accuracy to evaluate the model performance for both datasets.

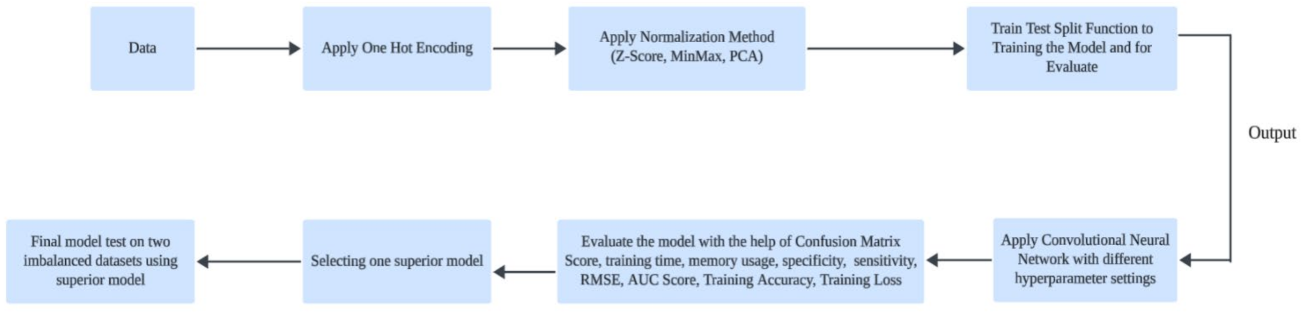

Data
Apply One Hot Encoding
Apply Normalization Method (Z-Score, MinMax, PCA)
Train Test Split Function to Training the Model and for Evaluate
Output
Apply Convolutional Neural Network with different hyperparameter settings
Evaluate the model with the help of Confusion Matrix Score, training time, memory usage, specificity, sensitivity, RMSE, AUC Score, Training Accuracy, Training Loss
Selecting one superior model
Final model test on two imbalanced datasets using superior model

Figure 1. Flow diagram of proposed method

4.3 One hot encoding

One hot encoding converts a specific feature's nominal data into integer data. Owing to one hot encoding, only one of the N states has a value of 1, while other states have a value of 0, guaranteeing that only one state of every sample is in the activated state. One hot encoding method is applied to the nominal data in this dataset.

Normalization is a data preprocessing step, where ML algorithm which is used by using sklearn, to transform categorial data into numerical values. The main aim is to normalization and make different features of a dataset comparable to each other, especially when they have different scales. This method helps to improve the neural network and gradient descent. This model uses three commonly used methods: Z score normalization, min-max normalization, and PCA. In the first setting, use min-max normalization and then apply CNN with different hyperparameter settings. Similarly, use the same hyperparameter settings as above for Z-score normalization and PCA.

The output of the preprocessing phase is now further fed into the train test split function to evaluate it into CNN as shown in Figure 1. This will also be used to evaluate the model's performance. The total number of layers used in the paper is 11, with 3 dense layers, which combine the pooling layer, flatten layer, and dense layer.

4.4 Investigation tool

The proposed model was conducted using google Colab with powerful T4 GPU with 12 GB RAM. The proposed model used the most frequent library called tensorflow and skit-learn. The proposed model was designed to use minimal computational resources and it is tested on advanced tool and use advanced library which give optimum result.

4.5 Model flow

The first step is to read data from the panda's library. After reading the data, it is passed to one-hot encoding if it has categorial features. Following this, the encoded data undergoes normalization using three different techniques: Z-score normalization, which standardizes the data, and Minmax normalization, which scales the data between a specified range. Lastly, PCA is applied. After the normalization, the dataset is split into training and testing sets to facilitate model evaluation, and the setting of train test data is 70%-30% split for both datasets.

After the data is prepared, it is passed through a convolutional neural network architecture. The first layer of the CNN consists of a Conv 1D layer with 64 filters and a kernel size of 3, using the ReLU activation function to introduce nonlinearity, followed by a max pooling-1D layer with a pool size (Different Combination) to down sample the data and reduce dimensionality. The second convolutional layer consists of a Conv1D with 128 filters and a kernel size of 3, again followed by a MaxPooling1D layer with a pool size of (Different Combination) for further down sampling. Finally, the output of this layer is then flattened into a vector format, which is fed into two dense layers. The first dense layer contains 128 units, while the second dense layer contains 64 units, and different settings have different activation functions, which are ReLU and tanh. The output layer consists of a sigmoid activation function to get the CNN output. Finally, the output of CNN goes for model evaluation phase. Lastly, after comparison all the model with parameters, we pick one those which are best of all the apply imbalanced dataset for both datasets.

5. RESULT AND DISCUSSION

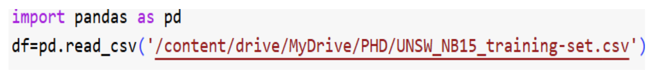
This section provides an analysis of proposed model which aim to find the best normalization method and hyperparameters which is more suitable for generalization in terms of the IDS system. Therefore, use different models with different hyperparameters to assess each model's accuracy.

5.1 Dataset 1: UNSW-NB15

This work considers first dataset, to check different result with different hyperparameters setting.

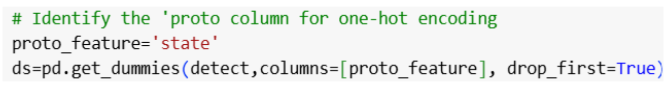
5.1.1 Min-max normalization with CNN

The following procedure is applied to implement the proposed model. First, data is read from the panda's library, as shown in Figure 2. After reading the data, one hot encoding is applied, as shown in Figure 3. The output of one hot encoding is fed into Min-Max normalization, as shown in Figure 4. Afterward, split the data into train and test parts using the sklearn function, as shown in Figure 5.



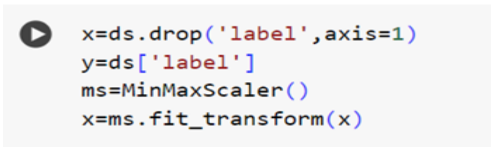
```
import pandas as pd
df=pd.read_csv('/content/drive/MyDrive/PHD/UNSW_NB15_training-set.csv')
```

Figure 2. Read the dataset 1



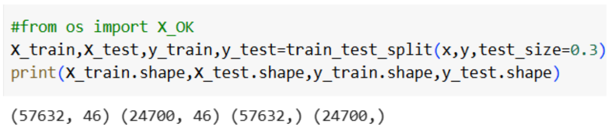
```
# Identify the 'proto column for one-hot encoding
proto_feature='state'
ds=pd.get_dummies(detect,columns=[proto_feature], drop_first=True)
```

Figure 3. One hot encoding function



```
x=ds.drop('label',axis=1)
y=ds['label']
ms=MinMaxScaler()
x=ms.fit_transform(x)
```

Figure 4. Minmax function



```
#from os import X_OK
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)

(57632, 46) (24700, 46) (57632,) (24700,)
```

Figure 5. Train test split function

Setting 1: Three dense layer CNN with ReLU activation function with pool size 2, 3

The output of the train-test split function is fed into the CNN model to train the CNN model. This model combines Conv1D, max pooling (Different size), Conv1D, max pooling (Different size), and flatten layers, followed by three dense layers sequence (128, 64, 1) of setting 1. After training the model, confusion matrix in terms of recall, F1 score, accuracy, precision, and AUC score, RMSE, TPR, FPR, Specificity, Sensitivity as shown in the Table 2. Additional Accuracy 0.9179 indicate weak generalization and poor generalization due to RMSE.

Table 2. Setting 1 of mix-max

Parameters	Values
Accuracy	0.9179
Precision	0.9167
Recall	0.9358
F1 Score	0.9262
Specificity	0.8961
Sensitivity	93.58%
RMSE	0.2865
AUC Score	0.9159
False Positive Rate (FPR)	0.1039
True Positive Rate (TPR)	0.9358

Setting 2: Three dense layer CNN with tanh activation function with pool size 2, 4

Apply same process and after evaluation it is found that the improved generalization over Setting 1 and low error rate followed by strong sensitivity indicates balanced performance and good generalization as shown in the Table 3.

Table 3. Setting 2 of mix-max

Parameters	Values
Accuracy	0.9372
Precision	0.9318
Recall	0.9559
F1 Score	0.9437
Specificity	0.9144
Sensitivity	95.59%
RMSE	0.2505
AUC Score	0.9352
False Positive Rate (FPR)	0.0856
True Positive Rate (TPR)	0.9559

Setting 3: Three dense layer CNN with tanh activation function with pool size 3, 3

Feeding the output to the train-test split function into CNN model, the accuracy is lower than setting 2 as shown in Table 4.

Setting 4: Three dense layer CNN with ReLU activation function with pool size 3, 3

The same process is used as above, and the result showed that setting 4 gives poor precision, generalization and AUC score as shown in Table 5.

Table 4. Setting 3 of mix-max

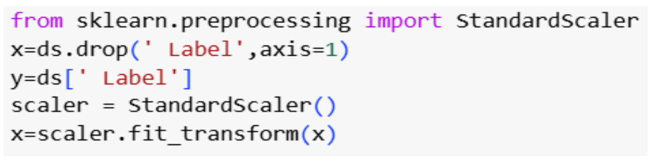
Parameters	Values
Accuracy	0.9309
Precision	0.9407
Recall	0.9332
F1 Score	0.9369
Specificity	0.9280
Sensitivity	93.32%
RMSE	0.2629
AUC Score	0.9306
False Positive Rate (FPR)	0.0720
True Positive Rate (TPR)	0.9332

Table 5. Setting 4 of mix-max

Parameters	Values
Accuracy	0.9164
Precision	0.8888
Recall	0.9692
F1 Score	0.9273
Specificity	0.8518
Sensitivity	96.92%
RMSE	0.2892
AUC Score	0.9105
False Positive Rate (FPR)	0.1482
True Positive Rate (TPR)	0.9692

5.1.2 Z-Score normalization with CNN

The output of one-hot encoding is then fed into Z normalization. After that, split the data into train and test parts using the sklearn function. Apply Z-score as shown in Figure 6.



```
from sklearn.preprocessing import StandardScaler
x=ds.drop(' Label',axis=1)
y=ds[' Label']
scaler = StandardScaler()
x=scaler.fit_transform(x)
```

Figure 6. Apply Z-score on dataset

Setting 1: Three dense layer CNN with ReLU activation function with pool size 2, 3

The same process is used which is given above and setting 4 is applied, and evaluate on Z-Score. The result showed good generalization, moderate recall, very few false positive and more generalization as shown in Table 6.

Setting 2: Three dense layer CNN with tanh activation function with pool size 2, 4

In this setting the performance is better than setting 1, as shown in Table 7.

Setting 3: Three dense layer CNN with tanh activation function with pool size 3, 3

In this setting 95.39% accuracy is achieved, which indicates strong generalization, lowest error among Z-score, Highest recall and high generalization as shown in Table 8.

Setting 4: Three dense layer CNN with ReLU activation function with pool size 3, 3

The output of this setting gives good performance matrix but it is lower than setting 3 of Z-Score. Furthermore, RMSE gives higher error than Setting 3, precision is lower than Setting 2 and lastly good sensitivity but not highest. It is shown in Table 9.

Table 6. Setting 1 of Z-Score

Parameters	Values
Accuracy	0.9489
Precision	0.9763
Recall	0.9297
F1 Score	0.9524
Specificity	0.9724
Sensitivity	92.97%
RMSE	0.2260
AUC Score	0.9510
False Positive Rate (FPR)	0.0276
True Positive Rate (TPR)	0.9297

Table 7. Setting 2 of Z-Score

Parameters	Values
Accuracy	0.9533
Precision	0.9739
Recall	0.9402
F1 Score	0.9567
Specificity	0.9693
Sensitivity	94.02%
RMSE	0.2161
AUC Score	0.9547
False Positive Rate (FPR)	0.0307
True Positive Rate (TPR)	0.9402

Table 8. Setting 3 of Z-Score

Parameters	Values
Accuracy	0.9539
Precision	0.9633
Recall	0.9524
F1 Score	0.9578
Specificity	0.9558
Sensitivity	95.24%
RMSE	0.2146
AUC Score	0.9541
False Positive Rate (FPR)	0.0442
True Positive Rate (TPR)	0.9524

Table 9. Setting 4 of Z-Score

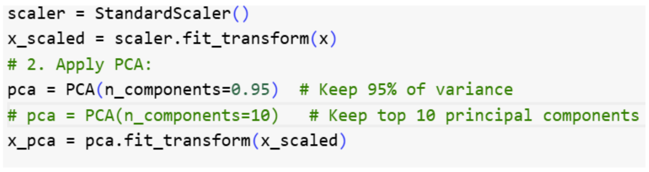
Parameters	Values
Accuracy	0.9509
Precision	0.9606
Recall	0.9497
F1 Score	0.9551
Specificity	0.9524
Sensitivity	94.97%
RMSE	0.2215
AUC Score	0.9511
False Positive Rate (FPR)	0.0476
True Positive Rate (TPR)	0.9497

5.1.3 PCA normalization with CNN

In the case of PCA, we follow the same procedure as above. The output of one-hot encoding is fed into PCA, as shown in the Figure 7.

Setting 1: Three dense layer CNN with ReLU Activation function with pool size 2, 3

This setting of PCA gives poor generalization, RMSE error is high and less precise in prediction and low sensitivity, and moderate generalization, which is shown in Table 10.



```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
# 2. Apply PCA:
pca = PCA(n_components=0.95)  # Keep 95% of variance
# pca = PCA(n_components=10)   # Keep top 10 principal components
x_pca = pca.fit_transform(x_scaled)
```

Figure 7. Apply PCA on dataset

Table 10. Setting 1 of PCA

Parameters	Values
Accuracy	0.9018
Precision	0.9565
Recall	0.8597
F1 Score	0.9055
Specificity	0.9527
Sensitivity	85.97%
RMSE	0.3133
AUC Score	0.9062
False Positive Rate (FPR)	0.0473
True Positive Rate (TPR)	0.8597

Setting 2: Three dense layer CNN with tanh activation function with pool size 2, 4

The same process is used but setting has changed. In setting 2, it underperforms as compared to other PCA settings due to poor accuracy, RMSE, F1 score, AUC score which is shown in Table 11.

Setting 3: Three dense layer CNN with tanh activation function with pool size 3, 3

This setting gives poor generalization due to accuracy (0.8965), and poor precision (0.8969), high sensitivity, and poor balance, which indicate that this setting is not sufficient to carry forward as shown in Table 12.

Table 11. Setting 2 of PCA

Parameters	Values
Accuracy	0.8965
Precision	0.8969
Recall	0.9162
F1 Score	0.9064
Specificity	0.8727
Sensitivity	91.62%
RMSE	0.3217
AUC Score	0.8945
False Positive Rate (FPR)	0.1273
True Positive Rate (TPR)	0.9162

Table 12. Setting 3 of PCA

Parameters	Values
Accuracy	0.8867
Precision	0.8711
Recall	0.9307
F1 Score	0.8999
Specificity	0.8335
Sensitivity	93.07%
RMSE	0.3366
AUC Score	0.8821
False Positive Rate (FPR)	0.1665
True Positive Rate (TPR)	0.9307

Setting 4: Three dense layer CNN with sigmoid activation function with pool size 3, 3

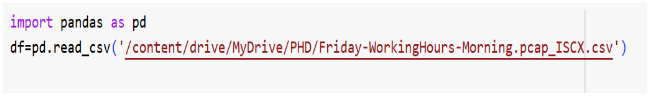
The result showed that moderate generalization, medium recall, and high RMSE is achieved and in conclusion it performs better among PCA settings, but still poor than Min-Max or Z-Score normalization method as shown in Table 13.

Table 13. Setting 4 of PCA

Parameters	Values
Accuracy	0.8977
Precision	0.9449
Recall	0.8633
F1 Score	0.9023
Specificity	0.9392
Sensitivity (Recall)	86.33%
RMSE	0.3199
AUC Score	0.9013
False Positive Rate (FPR)	0.0608
True Positive Rate (TPR)	0.8633

5.2 Dataset 2: CIC-IDS-2017

This work considers second dataset CIC-IDS-2017 to result with different hyperparameters settings of min-max, PCA, Z-score. Firstly, dataset is read from pandas' library as shown in Figure 8.



```
import pandas as pd
df=pd.read_csv('/content/drive/MyDrive/PHD/Friday-WorkingHours-Morning.pcap_ISCX.csv')
```

Figure 8. Second dataset read from pandas' library

5.2.1 Min-max normalization with CNN

Setting 1: Three dense layer CNN with ReLU activation function with pool size 2, 3

Setting 1 of min-max indicates the poor recall limits and good generalization results as shown in Table 14.

Table 14. Setting 1 of min-max

Parameters	Values
Accuracy	0.9943
Precision	0.8068
Recall	0.5946
F1 Score	0.6847
Specificity	0.9985
Sensitivity (Recall)	59.46%
RMSE	0.0755
AUC Score	0.7966
False Positive Rate (FPR)	0.0015
True Positive Rate (TPR)	0.5946

Setting 2: Three dense layer CNN with min-max activation function with pool size 2, 4

The accuracy 99.47% indicates the model with this setting indicate good generalization in terms of high precision, weak recall, and low error. The RMSE 0.073 indicate that the model of the setting 2 slightly better balance as shown in Table 15.

Setting 3: Three dense layer CNN with tanh activation function with pool size 3, 3

This work applied setting 3, which revealed the lower generalization in terms of AUC score and poor performance in terms of F1 Score, which is shown in Table 16.

Setting 4: Three dense layer CNN with ReLU activation function with pool size 3, 3

The setting 4 is the best among min-max normalization, due to best generalization, lowest error, slightly better recall, and achieve best F1 Score as shown in Table 17.

Table 15. Setting 2 of min-max

Parameters	Values
Accuracy	0.9947
Precision	0.8724
Recall	0.5729
F1 Score	0.6916
Specificity	0.9991
Sensitivity (Recall)	57.29%
RMSE	0.073
AUC Score	0.786
False Positive Rate (FPR)	0.0009
True Positive Rate (TPR)	0.5729

Table 16. Setting 3 of min-max

Parameters	Values
Accuracy	0.9941
Precision	0.8028
Recall	0.5729
F1 Score	0.6686
Specificity	0.9985
Sensitivity (Recall)	57.29%
RMSE	0.0769
AUC Score	0.7857
False Positive Rate (FPR)	0.0015
True Positive Rate (TPR)	0.5729

Table 17. Setting 4 of min-max

Parameters	Values
Accuracy	0.9949
Precision	0.8787
Recall	0.5946
F1 Score	0.7093
Specificity	0.9991
Sensitivity (Recall)	59.46%
RMSE	0.0713
AUC Score	0.7969
False Positive Rate (FPR)	0.0009
True Positive Rate (TPR)	0.5946

5.2.2 Z-Score normalization with CNN

The output of one-hot encoding is then fed into Z normalization, as shown in the Figure 6.

Setting 1: Three dense layer CNN with ReLU activation function with pool size 2, 3

In this setting the accuracy was 99.56%, which achieved good generalizations and lower RMSE, and AUC 0.801, but the recall value is 0.6025, which is lower as shown in Table 18.

Setting 2: Three dense layer CNN with tanh activation function with pool size 2, 4

From the above process, the different settings are used where RMSE is higher than setting 1, but the precision 0.9511 is effective which indicates that it may be considered for final result which is shown in Table 19.

Setting 3: Three dense layer CNN with tanh activation function with pool size 3, 3

Based on setting 3 of Z-score, the AUC Score of setting 3 indicates the lower generalization than Setting 2. But high precision which is 0.9511, but recall limits its effectiveness in this setting as shown in Table 20.

Setting 4: Three dense layer CNN with ReLU activation function with pool size 3, 3

This setting is the best among of all z-score of datasets 2. Which means good AUC score, high precision, and F1 score is best balance among the above setting as shown in Table 21.

Table 18. Setting 1 of Z-score

Parameters	Values
Accuracy	0.9956
Precision	0.9348
Recall	0.6025
F1 Score	0.7327
Specificity	0.9996
Sensitivity (Recall)	60.25%
RMSE	0.0662
AUC Score	0.801
False Positive Rate (FPR)	0.0004
True Positive Rate (TPR)	0.6025

Table 19. Setting 2 of Z-score

Parameters	Values
Accuracy	0.9955
Precision	0.9511
Recall	0.5797
F1 Score	0.7203
Specificity	0.9997
Sensitivity (Recall)	57.97%
RMSE	0.067
AUC Score	0.7897
False Positive Rate (FPR)	0.0003
True Positive Rate (TPR)	0.5797

Table 20. Setting 3 of Z-score

Parameters	Values
Accuracy	0.9954
Precision	0.935
Recall	0.5797
F1 Score	0.7157
Specificity	0.9996
Sensitivity (Recall)	57.97%
RMSE	0.0677
AUC Score	0.7896
False Positive Rate (FPR)	0.0004
True Positive Rate (TPR)	0.5797

Table 21. Setting 4 of Z-score

Parameters	Values
Accuracy	0.9955
Precision	0.9198
Recall	0.6025
F1 Score	0.728
Specificity	0.9995
Sensitivity (Recall)	60.25%
RMSE	0.067
AUC Score	0.801
False Positive Rate (FPR)	0.0005
True Positive Rate (TPR)	0.6025

5.2.3 PCA normalization with CNN

The output of one-hot encoding is then fed into PCA, as shown in the Figure 7.

Setting 1: Three dense layer CNN with ReLU activation function with pool size 2, 3

Finally, the last phase of dataset 2 is used. In first setting 1, this setting is best among the above PCA setting 2, 3, 4 due to high precision, best recall, and excellent balance of F1 Score as shown in Table 22.

Setting 2: Three dense layer CNN with tanh activation function with pool size 2, 4

In this setting 2, the parameters indicate this an excellent generalization, second best in PCA. Additionally, F1 Score indicate good balance and RMSE is slightly higher error than setting 1, which is shown in Table 23.

Table 22. Setting 1 of PCA

Parameters	Values
Accuracy	0.9956
Precision	0.932
Recall	0.6208
F1 Score	0.7452
Specificity	0.9995
Sensitivity (Recall)	62.08%
RMSE	0.0664
AUC Score	0.8102
False Positive Rate (FPR)	0.0005
True Positive Rate (TPR)	0.6208

Table 23. Setting 2 of PCA

Parameters	Values
Accuracy	0.9955
Precision	0.9375
Recall	0.604
F1 Score	0.7347
Specificity	0.9996
Sensitivity (Recall)	60.40%
RMSE	0.0674
AUC Score	0.8018
False Positive Rate (FPR)	0.0004
True Positive Rate (TPR)	0.6040

Table 24. Setting 3 of PCA

Parameters	Values
Accuracy	0.9954
Precision	0.9326
Recall	0.604
F1 Score	0.7332
Specificity	0.9995
Sensitivity (Recall)	60.40%
RMSE	0.0676
AUC Score	0.8018
False Positive Rate (FPR)	0.0005
True Positive Rate (TPR)	0.6040

Table 25. Setting 4 of PCA

Parameters	Values
Accuracy	0.9949
Precision	0.842
Recall	0.6258
F1 Score	0.718
Specificity	0.9988
Sensitivity (Recall)	62.58%
RMSE	0.0715
AUC Score	0.8123
False Positive Rate (FPR)	0.0012
True Positive Rate (TPR)	0.6258

Setting 3: Three dense layer CNN with tanh activation

function with pool size 3, 3

For the setting 3 in PCA phase of dataset 2, the overall value indicate that it is weaker than Setting 1 and 2 but still good. Furthermore, recall (60.40%) is same as Setting 2 and F1 score 73.32% is slightly lower than setting 2 as shown in Table 24.

Setting 4: Three dense layer CNN with sigmoid activation function with pool size 3, 3

Finally, the last one which is setting 4, this setting gives weakest generalization among PCA setting but recall is balanced as shown in Table 25.

From the above setting of both dataset 1 and dataset 2. The Z-score setting 3 have best accuracy (95.39%), low RMSE (0.2146), high precision (96.33%), and high recall (95.24%). And For Dataset 2 the best setting is (PCA, setting 1) which shows strong accuracy (99.56%), low RMSE (0.0664), high precision (93.20%), and best recall (62.08%). And in conclusion, which one should pick among those? In that case, Z-Score setting 3 is used because due to balanced performance across all metrics which makes suitable for all other application like intrusion detection even on the basis of both precision and recall are critical factor.

5.3 Memory and time usage

This section discussed the different normalization method of PCA, Z score, Min-Max with respect to two datasets are as follows.

Table 26. Dataset 1 training time and memory usage

Model Setting	Training Time (s)	Memory Usage (MiB)
Z-Score Setting 1	157.33	1271.28
Z-Score Setting 2	187.39	1281.83
Z-Score Setting 3	104.62	1249.21
Z-Score Setting 4	102.34	1266.54
Min-Max Setting 1	159.11	1317.90
Min-Max Setting 2	144.47	1353.68
Min-Max Setting 3	101.85	1329.94
Min-Max Setting 4	123.12	1396.12
PCA Setting 1	94.71	1190.85
PCA Setting 2	67.05	1208.55
PCA Setting 3	42.55	1231.015
PCA Setting 4	50.71	1256.05

Table 27. Dataset 2 training time and memory usage

Model Setting	Training Time (s)	Memory Usage (MiB)
Min-Max Setting 1	474.63	1586.74
Min-Max Setting 2	446.63	1726.32
Min-Max Setting 3	368.08	1639.22
Min-Max Setting 4	397.86	1680.63
Z-Score Setting 1	503.53	1608.5
Z-Score Setting 2	506.7	1782.68
Z-Score Setting 3	387.33	1665.37
Z-Score Setting 4	385.23	1638.32
PCA Setting 1	117.01	1666.37
PCA Setting 2	126.79	1724.15
PCA Setting 3	94.98	1734.71
PCA Setting 4	99.57	1777.37

Dataset 1: This section includes total 12 setting of dataset 1, to check the training time and memory usage which is shown in Table 26.

Dataset 2: This section includes total 12 setting of dataset 2, to check the training time and memory usage which is shown in Table 27.

From all above settings, Z-score setting 1 has less memory as compared to PCA but the disadvantage is that time of PCA is faster than Z-score.

5.4 Evaluation of model performance on imbalanced datasets

After running the Z score setting 3 on dataset 1. It applied on two dataset and check on imbalanced ratio and how to tackle the attacks in IDS.

Figures 9 and 10 show the imbalanced ratio, which is used in further evaluation. After the evaluation the dataset 1 and dataset 2 parameters value are shown in Tables 28 and 29. In first dataset as the attack ratio increase the model recall value is increased as shown in Table 28 which indicate better detection ratio. For second dataset the accuracy maintains even though apply first setting of dataset 2. Also, AUC score is high which indicate model has potential to identify threat easily. Additionally, a visualization graph is depicted for dataset 1 and dataset 2 in Figures 9 and 10.

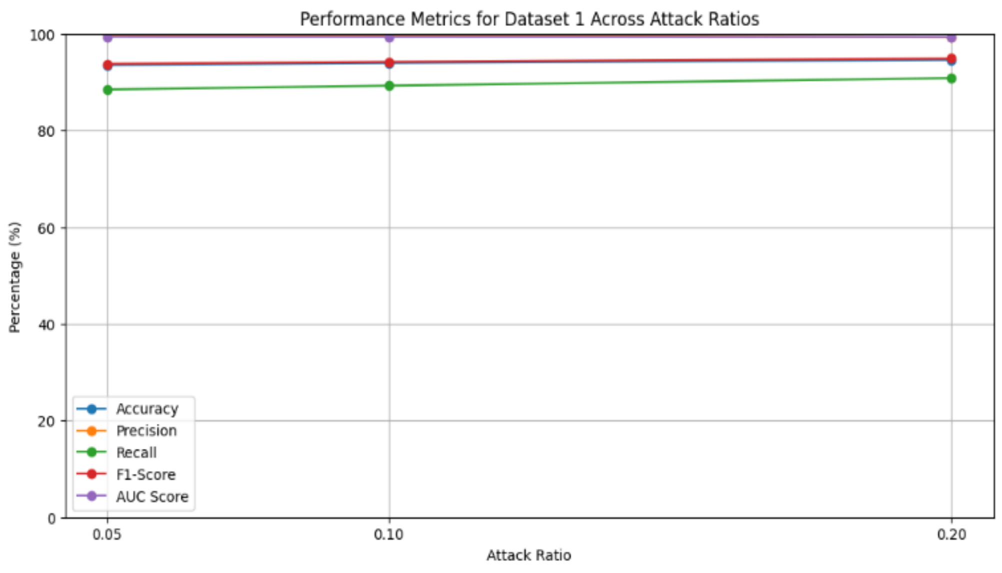


Figure 9. First dataset performance metrics

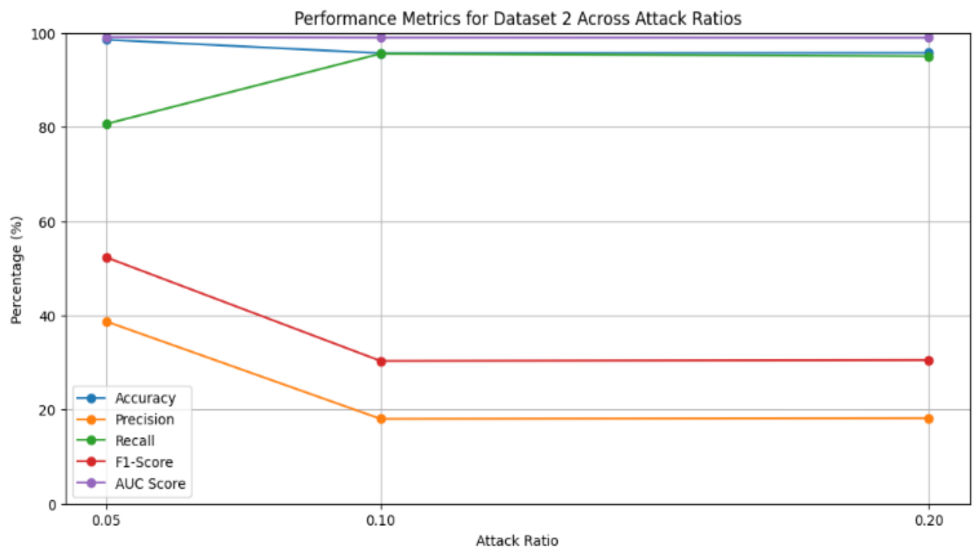


Figure 10. Second dataset performance metrics

Table 28. Dataset 1 performance metrics

Metric	Attack Ratio: 0.05	Attack Ratio: 0.1	Attack Ratio: 0.2
Accuracy	93.54%	93.95%	94.59%
Precision	99.73%	99.68%	99.27%
Recall (Sensitivity)	88.47%	89.26%	90.82%
F1 Score	93.76%	94.18%	94.86%
AUC Score	99.37%	99.35%	99.33%
RMSE	0.2295	0.2157	0.2004

Table 29. Dataset 2 performance metrics

Metric	Attack Ratio: 0.05	Attack Ratio: 0.1	Attack Ratio: 0.2
Accuracy	98.55%	95.67%	95.74%
Precision	38.69%	18.02%	18.16%
Recall (Sensitivity)	80.67%	95.57%	95.04%
F1 Score	52.30%	30.32%	30.50%
AUC Score	99.11%	98.99%	98.97%
RMSE	0.0985	0.1534	0.1592

6. CONCLUSIONS AND FUTURE SCOPE

IDS enhances network security by identifying threats and malicious activity in computer systems. We propose a model that improves the IDS system by integrating normalization and only three dense layers of CNN. Aim of work to demonstrate how a 3-layer CNN would work effectively with the combination of normalization methods to enhance network security. After applying 24 parameters settings it is observed that the Z-score setting 3 has the best option to choose for model evaluation on the basis of F1 score, precision, recall, accuracy, RMSE, AUC Score and learning curve accuracy. The future scope of the proposed model with 3 dense layer include testing on large dataset and decrease the computational cost in terms of time and memory with these settings.

REFERENCES

[1] Zhao, R., Gui, G., Xue, Z., Yin, J., Ohtsuki, T., Adebisi, B., Gacanin, H. (2021). A novel intrusion detection method based on lightweight neural network for Internet of Things. IEEE Internet of Things Journal, 9(12): 9960-9972. https://doi.org/10.1109/JIOT.2021.3069234

[2] Saba, T., Rehman, A., Sadad, T., Kolivand, H., Bahaj, S.A. (2022). Anomaly-based intrusion detection system for IoT networks through deep learning model. Computers and Electrical Engineering, 99: 107810. https://doi.org/10.1016/j.compeleceng.2022.107810

[3] Liu, J., Yang, D., Lian, M., Li, M. (2021). Research on intrusion detection based on particle swarm optimization in IoT. IEEE Access, 9: 38254-38268. https://doi.org/10.1109/ACCESS.2021.3062376

[4] Abbas, A., Khan, M.A., Latif, S., Ajaz, M., Shah, A.A., Ahmad, J. (2022). A new ensemble-based intrusion detection system for Internet of Things. Arabian Journal for Science and Engineering, 47: 1805-1819. https://doi.org/10.1007/s13369-022-06399-4

[5] Mehedi, S.T., Anwar, A., Rahman, Z., Ahmed, K., Islam, R. (2022). Dependable intrusion detection system for IoT: A deep transfer learning-based approach. IEEE Transactions on Industrial Informatics, 19(1): 1006-1017. https://doi.org/10.1109/TII.2022.3185875

[6] Sarhan, M., Layeghy, S., Portmann, M. (2021). Feature analysis for machine learning-based IoT intrusion detection. arXiv preprint, arXiv:2108.12732. https://arxiv.org/abs/2108.12732

[7] Bouke, M.A., Abdullah, A., ALshatebi, S.H., Abdullah, M.T. (2022). E2IDS: An enhanced intelligent intrusion detection system based on decision tree algorithm. Journal of Applied Artificial Intelligence, 3(1): 1-16. https://doi.org/10.48185/jaai.v3i1.450

[8] Tally, M.T., Amintoosi, H. (2021). A hybrid method of genetic algorithm and support vector machine for intrusion detection. International Journal of Electrical & Computer Engineering, 11(1): 1-12. https://doi.org/10.11591/ijece.v11i1.12345

[9] Pathak, A., Pathak, S. (2020). Study on decision tree and KNN algorithm for intrusion detection system. International Journal of Engineering Research & Technology, 9(5): 376-381.

[10] Saheed, Y.K., Abiodun, A.I., Misra, S., Holone, M.K., Colomo-Palacios, R. (2022). A machine learning-based

intrusion detection for detecting Internet of Things network attacks. Alexandria Engineering Journal, 61(12): 9395-9409. https://doi.org/10.1016/j.aej.2022.09.123

[11] Musleh, D., Alotaibi, M., Alhaidari, F., Rahman, A., Mohammad, R.M. (2023). Intrusion detection system using feature extraction with machine learning algorithms in IoT. Journal of Sensor and Actuator Networks, 12(2): 29. https://doi.org/10.3390/jsan12020029

[12] Syamsuddin, I., Barukab, O.M. (2022). SUKRY: Suricata IDS with enhanced kNN algorithm on Raspberry Pi for classifying IoT botnet attacks. Electronics, 11(5): 737. https://doi.org/10.3390/electronics11050737

[13] Aslam, M., Ye, D., Tariq, A., Asad, M., Hanif, M., Ndzi, D., Chelloug, S.A., Elaziz, M.A., Al-Qaness, M.A.A., Jilani, S.F. (2022). Adaptive machine learning based distributed denial-of-services attacks detection and mitigation system for SDN-enabled IoT. Sensors, 22(7): 2697. https://doi.org/10.3390/s22072697

[14] Verma, A., Ranga, V. (2020). Machine learning based intrusion detection systems for IoT applications. Wireless Personal Communications, 111(4): 2287-2310. https://doi.org/10.1007/s11277-019-06986-8

[15] El-Sayed, R., El-Ghamry, A., Gaber, T., Hassanien, A.E. (2021). Zero-day malware classification using deep features with support vector machines. In Proceedings of the 10th International Conference on Intelligent Computing and Information Systems (ICICIS), pp. 1-8. https://doi.org/10.1109/ICICIS.2021.9634967

[16] Garg, S., Kaur, K., Kumar, N., Kaddoum, G., Zomaya, A.Y., Ranjan, R. (2019). A hybrid deep learning-based model for anomaly detection in cloud datacenter networks. IEEE Transactions on Network and Service Management, 16(3): 924-935. https://doi.org/10.1109/TNSM.2019.2927886

[17] Savic, M., Lukic, M., Danilovic, D., Bodroski, Z., Bajovic, D., Mezei, I., Vukobratovic, D., Skrbic, S., Jakovetic, D. (2021). Deep learning anomaly detection for cellular IoT with applications in smart logistics. IEEE Access: Practical Innovations, Open Solutions, 9: 59406-59419. https://doi.org/10.1109/ACCESS.2021.3072916

[18] Gopali, S., Siami Namin, A. (2022). Deep learning-based time-series analysis for detecting anomalies in Internet of Things. Electronics, 11(19): 3205. https://doi.org/10.3390/electronics11193205

[19] Otoum, Y., Liu, D., Nayak, A. (2022). DL-IDS: A deep learning-based intrusion detection framework for securing IoT. Transactions on Emerging Telecommunications Technologies, 33(3): e3803. https://doi.org/10.1002/ett.3803

[20] Apostol, I., Preda, M., Nila, C., Bica, I. (2021). IoT botnet anomaly detection using unsupervised deep learning. Electronics, 10(16): 1876. https://doi.org/10.3390/electronics10161876

[21] Ullah, I., Ullah, A., Sajjad, M. (2021). Towards a hybrid deep learning model for anomalous activities detection in Internet of Things networks. IoT, 2(3): 428-448. https://doi.org/10.3390/iot2030022

[22] Awajan, A. (2023). A novel deep learning-based intrusion detection system for IoT networks. Computers, 12(2): 34. https://doi.org/10.3390/computers12020034

[23] 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. Security and Communication Networks, 2022(1): 4016073. https://doi.org/10.1155/2022/4016073

[24] Kaggle. (2025). CICIDS 2019 Dataset. https://www.kaggle.com/datasets/tarundhamor/cicids-2019-dataset?select=UDPLag_data_2_0_per.csv.

[25] Moustafa, N., Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 military communications and information systems conference (MilCIS), Canberra, ACT, Australia, pp. 1-6. https://doi.org/10.1109/MilCIS.2015.7348942

[26] Liu, J., Yang, D., Lian, M., Li, M. (2021). Research on classification of intrusion detection in internet of things network layer based on machine learning. In 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR), Tokoname, Japan, pp. 106-110. https://doi.org/10.1109/ISR50024.2021.9419529