International Journal Of Multi Research

Online ISSN: 3107 - 7676

IJMR 2026; 2(3): 27-33

2026 May - June

www.allmultiresearchjournal.com

Received: 18-03-2026

Accepted: 20-04-2026

Published: 15-05-2026

DOI: https://doi.org/10.54660/IJMR.2026.2.3.27-33

Enhanced Anomaly Detection in Cloud Environments Using Deep Learning

Ankit Sharma [1], Aniket Soni [2], Prashant Gupta [3], Kunal Saxena [4]

[1, 3, 4] Parul University, Vadodara, Gujarat, India

[2] Poornima University, Jaipur, Rajasthan, India

Corresponding Author; **Ankit Sharma**

Abstract

End-to-end system communication has increased dramatically as a result of advancements in computer networking. Nevertheless, concerns about security have also been voiced. Consequently, it is still difficult to spot abnormalities in a complicated cloud environment. As a result, a deep Convolutional Neural Network (CNN) detection and classification model for near-real-time cloud network intrusions is proposed in this study. To choose the most appropriate characteristics to feed into the CNN model, the random forest model is also available and used. The experiments utilized CSE-CIC-IDS2018 datasets. With an error rate of just 2.93 percent and a testing accuracy of 97.07 percent, the proposed CNN model performed better than the competition. The accuracy, recall, and f1-score of the suggested model were 98.11, 96.93, and 97.52%, respectively, in these evaluations. These findings show great promise for real-time Industry 4.0 systems, since they are more accurate, precise, and capable of detecting network irregularities with the utmost precision.

Keyword: Cloud Computing, Anomaly Detection, Deep Learning, Security, Cloud Monitoring

1. Introduction

The management and creation of information technology services by businesses has recently been altered by cloud computing. Connecting data and applications to complex cloud conditions offers advantages such as scalability, adaptability, and cost-effectiveness. But, new problems have emerged as a result of this change, particularly in the areas of intrusion detection and network security. As our dependence on computer networks grows and cyber attacks become more sophisticated, there is an urgent need for reliable Network Intrusion Detection Systems (NIDS). The ability to detect and react to intrusion attempts, hostile movements, and potential security breaches is critical for keeping computer networks safe [1].

Due to the scattered and ever-changing nature of cloud environments, conventional network intrusion detection approaches are rendered ineffective. When it comes to identifying new and unknown threats, traditional NIDS systems have often depended on signature-or rule-based approaches, which have their limitations. Because of their diverse service offerings, virtualized tools, and multi-tenant designs, cloud environments' ever-expanding attack surfaces make it harder to quickly identify and respond to network incursions [2]. On the flip side, identifying network abnormalities in Industry 4.0 is complicated and plagued by problems due to data fluctuations, real-time needs, loud and diverse settings, and the merging of operational and information technologies. Furthermore, the massive velocity and quantity of cloud data, together with the high network traffic, continue to make intrusion detection a tough task. An efficient and adaptable detection system is required for a massively scalable cloud network that can simultaneously collect and analyze enormous volumes of network data [3]. In order to tackle these difficulties in detecting network intrusion in complicated cloud settings, Industry 4.0 has recently made extensive use of modern machine learning and artificial intelligence approaches. Also, a new crop of methods for improving the efficacy and precision of network intrusion detection has emerged: deep learning techniques,

particularly deep convolutional neural networks (CNNs) and Recurrent Neural Networks (RNNs) [4].

This article focuses on network intrusion detection using deep convolutional neural networks. In particular, we discuss the architecture and design of the deep convolutional neural network (CNN) NIDS model. Network intrusion detection systems may benefit from the use of deep learning models to improve their accuracy, speed of response, and resistance to new threats. Users may better protect their sensitive data and systems from unwanted access and criminal actions by mixing deep learning technologies with NIDS, which improves computer network security [5].

In order to distinguish distinctive characteristics from complex data, we first construct a robust random forest model by renovating the data and removing undesirable values from an inconsistent dataset. These are the major contributions of the current study. Since intrusion detection is not a linear problem, we suggest using a deep convolutional neural network (CNN) model to detect and categorize network threats. This method uses a combination of decision trees to classify features and minimize data dimensionality. We train the model using many layers and hyperparameters, which improves its performance, and then (4) we utilize near-real-time datasets to assess the model's efficiency and compare the findings to previous literature [6].

Part two of the study covers related work, and part three presents the datasets and technique, which includes data pre-processing, feature selection, and model construction. Results and accuracy achieved using the proposed deep learning model are discussed in Section 4. In the last part, the current research is summarized [7].

2. Associated Research

Researchers have made sporadic attempts to employ machine learning and deep learning techniques for network intrusion detection. The Apache Spark model, which was utilized in the research, serves as an illustration of a deep learning model. It achieved 97.8% accuracy on the CSE-CICIDS2018 dataset, 97.7% accuracy on the same dataset, and 99.9% accuracy overall [8]. However, the training time for long short-term memory (LSTM) models is 125.29 minutes, while that for convolutional neural network (CNN) models is 150.26 minutes. The LSTM-SGDM, LSTM-ADAM, CNN, CNN-LSTM, RC-NN, and DKNN models achieved 66.38, 63.69, 58.52, 63.34, 94.61, and 97.11 % accuracies, respectively, in the study on network intrusion using deep hybrid learning conducted on the CSECICIDS2018 and DARPA-IDS datasets. Their research demonstrates that the accuracy of the CNN model is low, at 58.52%, and that it can only be improved by combining the best features found using ensemble learning methods [9]. RNN and CNN models were used in studies that used the KDD and CSE-CICIDS2018 datasets. However, they tested the models to the best of their abilities and produced the RGB images of the utilized datasets. When applied to NSL-KDD datasets, the deep learning model achieved an accuracy of just 83.87%. A deep learning model with an accuracy of 90.73 percent was also constructed using the NSL-KDD dataset [10]. The accuracy of the deep neural network model for intrusion detection with NSL-KDD datasets was only 81.87%. In addition, the UNSWNB15 dataset was utilized to train LSTM and deep neural network models, which achieved very low accuracies of 88.75 and 95.05 percent, respectively, for binary classification. Furthermore, the deep learning-based CNN model utilized in the UNSW-NB15 study was only able to identify the ten network assaults with 95.6% accuracy in only 25% of the testing dataset [11]. Prior studies either used inadequate or oversimplified datasets, leading to inaccurate results. Also, prior research mostly employed complicated datasets for binary classification, as CSE-CICIDS2018 or UNSW-NB15. On the other hand, deep learning-based models were able to achieve an accuracy of over 95% even with relatively simple datasets like NSL-KDD data. Despite improvements, detecting and implementing changes in the CSE-CICIDS2018 datasets remains a formidable challenge due to their complexity and relatively large feature count. Our motivation for doing this study is the ongoing need to enhance the accuracy of deep learning models when applied to this data set [12].

3. Materials and Methods

A. The Datasets

The CSE-CICIDS2018 dataset, which was created by the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick, is a popular option for cybersecurity researchers working in intrusion detection and network analysis [13]. It allows academics and practitioners to build and test intrusion detection algorithms and systems by capturing a variety of network data, both benign and malicious. It covers a wide range of network attack scenarios and contains tagged data for a variety of attacks [14]. We utilized four days of statistics from the CICFlowMeter, including "Wednesday-14-02-2018 and 21-02-2018," "Thursday-15-02-2018," and "Friday-16-02-2018," for this investigation. The dataset is over 1.30 GB in size, consisting of four CSV files containing 5,43,589 items and 80 characteristics each [15].

B. Data Pre-processing

During the pre-processing of the raw data, unwanted or duplicate values were removed. Furthermore, in order to eradicate any data inaccuracies, the null, missing, or zero data was cleansed and repaired. Finally, everything was set up so that the data could be processed. Additionally, the data was cleansed of certain missing values. The dataset is processed in the final step by being converted to the float data format and then divided into target variables (y) and input features (x). The pre-processing steps produced a numerical dataset with no discrepancies [16].

C. Feature selection using Random Forest

In order to categorize the characteristics, the random forest algorithm employs an ensemble learning technique that combines many decision trees. It is widely used in applications for feature selection and categorization due to its adaptability and effectiveness. The random forest evaluates a feature's relevance based on the information gain or decrease in impurity brought about by splitting on a particular feature. This data may be useful for feature selection [17]. To learn and choose the important characteristics, sci-kit-learn uses the RandomForestClassifier. Using n estimators=1000 (decision trees) and random state=40, the dataset separations for training and testing are 70% and 30%, respectively. The random forest method took 653.2871 seconds to train the model. Figure 2 shows the results of using constant threshold values of 0.015 to choose the most important features from the random forest classifier model. Seventeen crucial characteristics were supplied by the random forest selection model [18].

D. Data Preparation

After cleansing the data of any bias, the chosen data was divided evenly across three thousand samples with a random state of 123, and it was prepared for convolutional neural networks (CNNs) using three distinct datasets for each feature label.

E. Deep Convolutional Neural Network

The deep CNN model, which is the most prevalent type of neural network, is ideal for analyzing complex characteristics and the enormous volumes of network traffic data generated by cloud settings. Using deep representations learned from training data, these models are able to identify and classify network assaults with greater accuracy and fewer false positives. A major benefit of deep CNNs is that they can learn features automatically from raw data on network traffic, rather than relying on manually created features. Additionally, deep CNNs are a good fit for zero-day attack detection due to their adaptability and generalizability to novel attack types [19]. A convolutional neural network's (CNN) architecture is enhanced by activation functions and multiple layers. A convolutional neural network (CNN) model's input, 1D convolutional, max-pooling, full-connection, and feedforward neural network–based output layers are its primary hyperparameters. This study used four distinct datasets to test our CNN model, which has a 1D convolutional layer with 128 filters, increasing kernel sizes, and the ReLU activation function. We used MaxPooling1D with filters=64, kernel size=3, strides=2, padding='same,' and flattened layers to create the 1D sequential models. The ReLU activation algorithm was used to run the 128-unit fully linked layer with Dense-units=64 and activation='SoftMax.' To further avoid overfitting, we added an output layer with the SoftMax activation function and set the dropout layer to 0.5. The model was last built using an Adam optimizer and a categorical cross-entropy loss function [20].

4. Results and Discussion

The machine learning and deep learning models were implemented and tested on personal computers running Windows 10 Home 64-bit, with Intel(R) Core (TM) i5-7200U processors, graphics 2.0 GHz, and 8 GB of RAM. The technique that was recommended in the current study was calculated using Python 3.10 in Anaconda Jupyter Notebook. Numpy, Pandas, Matplotlib, Sklearn, Keras, TensorFlow, and many more were among the Python libraries. Identifying and removing any null or infinite values, as well as determining the frequency of each assault type, were the first steps in pre-processing the raw data. There was one benign class and five bad ones in the original data: FTP-BruteForce, DoS attacks-SlowHTTPTest, DDOS attack-HOIC, DoS attacks-GoldenEye, and DoS attacks-Slowloris. The frequency of each kind of assault can be found in Figure 1 [21].

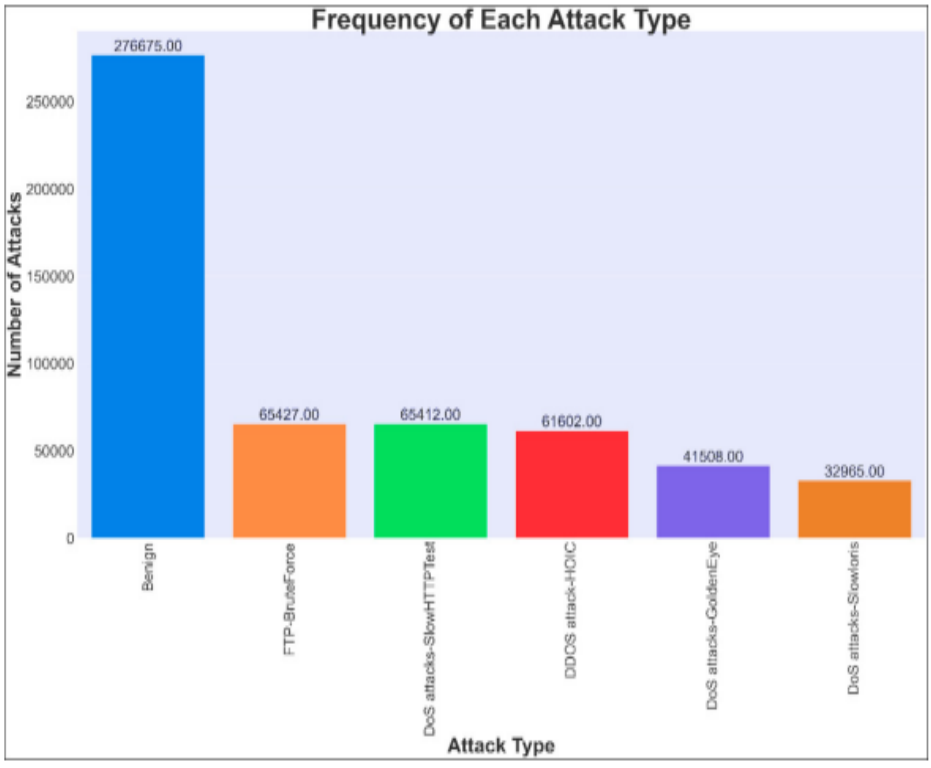

Fig 1: The number of instances of each kind of assault.

The process of selecting the necessary seventeen traits from a pool of eighty using the random forest method is shown in Figure 2. Therefore, the undesirable characteristics were removed from future processing without losing any crucial information. The resulting dataset contained just seventeen significant features and 5,43,589 rows following feature selection using the random forest method. Figure 2 shows the important traits that were chosen using the random forest algorithm.

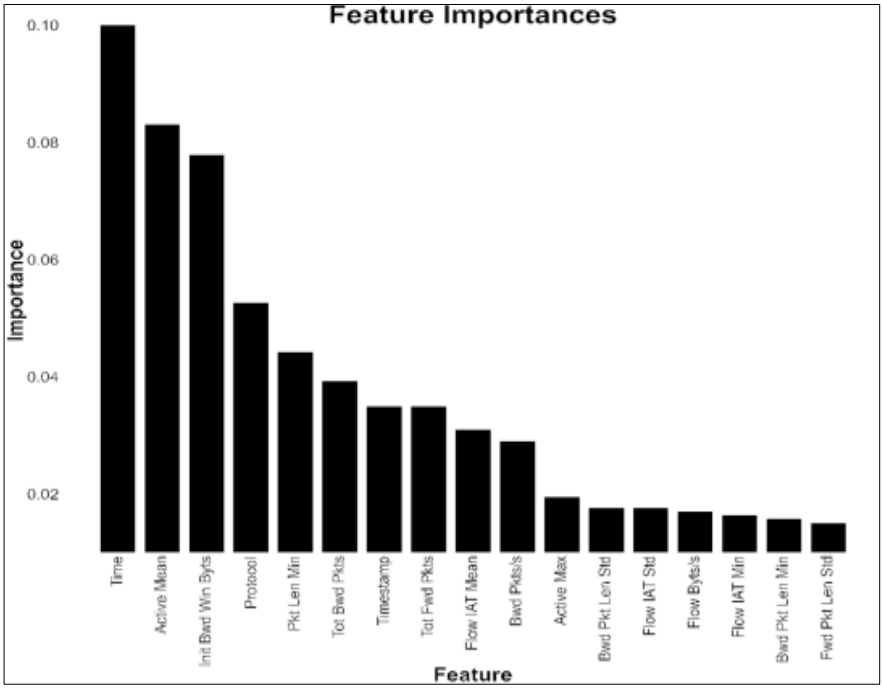

Fig 2: The seventeen essential characteristics that were selected using the random forest method.

Models were trained and evaluated using data that was divided into train and test sets, with the former containing 70% and the latter 30%. With the hyperparameters of the suggested CNN model—"192 hidden nodes," "64 batch size," "learning rate=0.0001," "categorical cross-entropy loss," "SoftMax" for the activation function, and the "Adam" optimizer—the model was put into action. There were 29,830 parameters that were not trainable and 30,214 that were trainable. The CNN model took "20.66" minutes to train and "0.21" seconds to test as a result. The accuracy and loss for validation were 0.3591, while the accuracy and loss for training were 0.9880, 0.0531, and 0.9707, respectively. The training and testing accuracy and loss of the implemented CNN model are shown in Table 1 [22].

We used an error matrix, 10-fold cross-validation, and measures such as recall, accuracy, precision, and the F1-score, which were calculated using Eqs. (1), (2), (3), and (4), respectively, to assess the CNN model's performance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1 - Sccore = 2 * \frac{Recall * Precision}{Recall + Precision} \quad (4)$$

When TP, TN, FP, and FN are all positive, the actual and predicted values are both one. Both the actual and true negative predictions are zero. A false negative constantly forecasts zero but is really one, whereas a false positive implies one but the essential is zero.

Table 1: Loss of models in convolutional neural networks (CNNs) and training accuracy

Metric	Train-the-CNN (%)	Reliability (%)
Precision	98.80	97.07
	0.531	3.591

Table 2 displays the results of the training and testing, while Figure 3 demonstrates that the CNN model achieved the best results in both areas, with a training accuracy of 98.80% and a testing accuracy of 97.07%. While training datasets had values of 99.89%, 97.83%, and 98.94%, respectively, testing datasets had scores of 98.11% for accuracy, 96.33% for recall, and 97.51% for F1. Table 2 and Figure 3 show the results of the testing and training datasets, respectively, where the maximum accuracy was attained [23].

Table 2: Using both training and testing datasets, the CNN model produces its output**.**

Performance Indicators	Assessment Outcomes (%)	Results of the Tests (%)
Accuracy	98.80	97.07
Precision	99.89	98.11
Recall	97.83	96.93
F1-Score	98.94	97.51

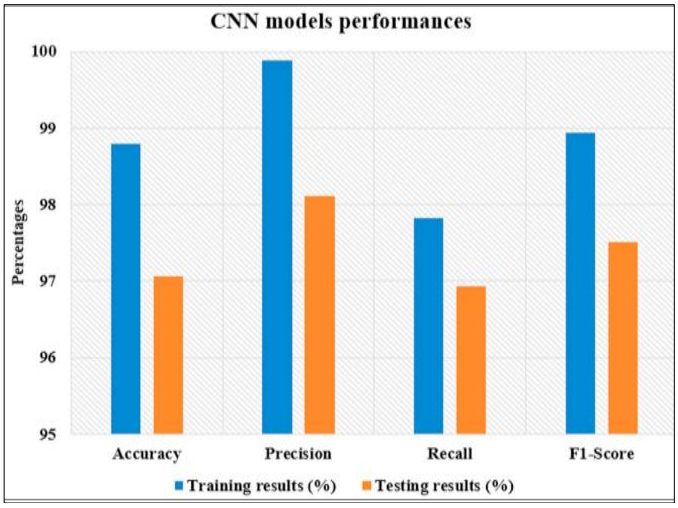

Fig 3: CNN model-produced outcomes

Table 3 displays the class-specific precision, recall, and F1-values of the calculated CNN model using testing datasets. The CNN model's accuracy and loss were assessed by running the training and testing sets for each epoch. Given the intricacy of the data used to display the train and test

accuracy and loss, we used forty epochs to appropriately train the CNN model in our situation. The accuracy plot shows that the CNN model had an accuracy of more than 98% after 13 training epochs. On the other hand, Figure 4 shows that the testing accuracy was at its peak on epochs 20, 34, and 40. This indicates that the model has acquired the ability to correctly identify each assault. On the other hand, the loss reveals the overall error rate for each training sample. Results show that our research had the lowest loss, the best training accuracy, and the best testing accuracy (Figures 4 and 5). With forty epochs, the CNN model's training and testing accuracy is shown in Figure 4. The training loss, which was a pitiful 2%, remained constant across all thirteen iterations. Figure 5 shows that the testing loss was minimal at 20, 34, and 40 epochs. With forty epochs, the loss results reveal a considerable decrease in the erroneous classification. In Figure 5, we can see the loss curve as it progresses through the testing and training epochs. With little training and testing time, the suggested CNN model has achieved accuracy across all classes.

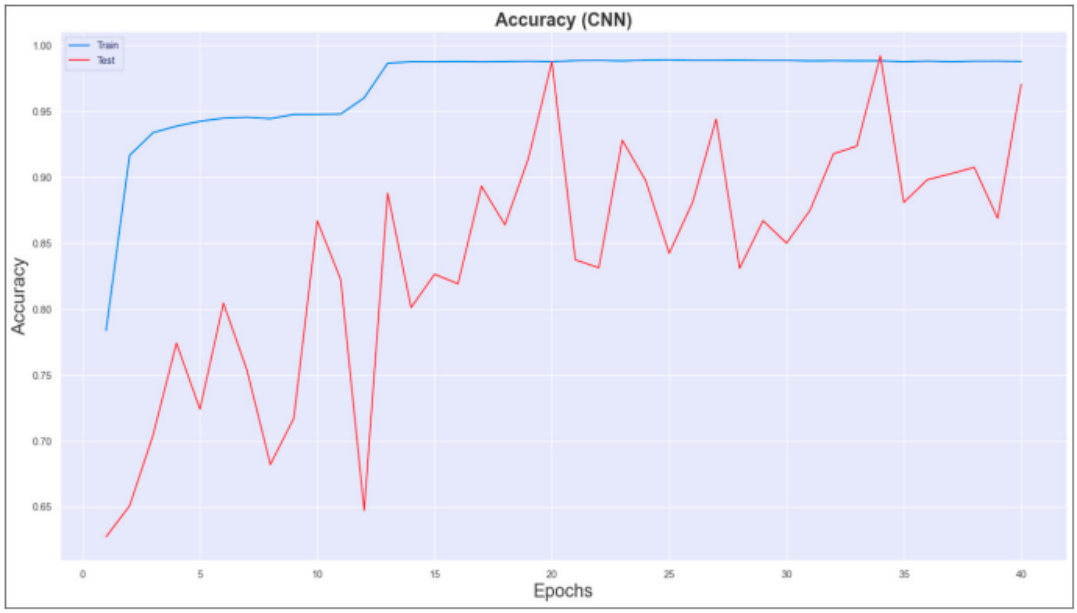

Fig 4: For both training and testing data, the CNN models' accuracy is shown across forty epochs.

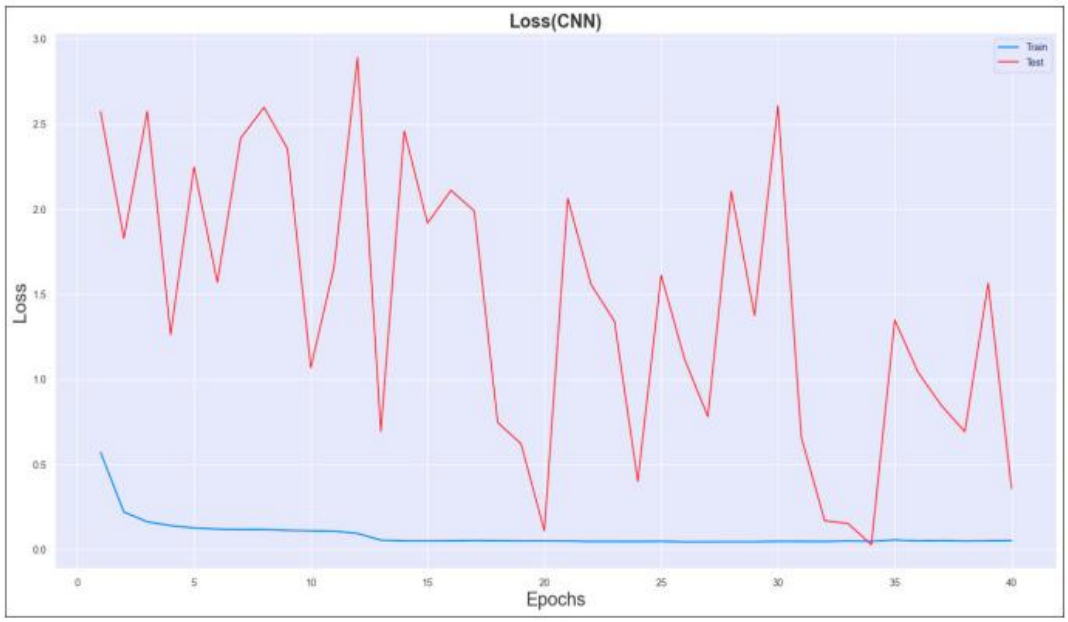

Fig 5: The CNN model's loss plot is displayed using forty training and testing epochs of data.

The loss curve clearly shows that there are almost no erroneous estimations in the test set. So, after forty epochs of training, the CNN model was accurate. The CNN model's error matrix, shown in Figure 6, demonstrates that the cases that were correctly categorized are acceptable, indicating its resilience. Figure 6 depicts the CNN model's error matrix based on the selected datasets. The error matrix is useful for resolving classification issues due to its clear indication of actual and estimated classes. Only 2.94 percent of the data was incorrectly classified by the CNN model, according to our findings. When it came to accurately recognizing and categorizing the various types of assaults from complicated datasets, the deployed CNN model achieved the greatest predicted accuracy of 97.06%. This proves that the suggested CNN model outperforms previous studies in accurately identifying and categorizing network assaults from complicated datasets.

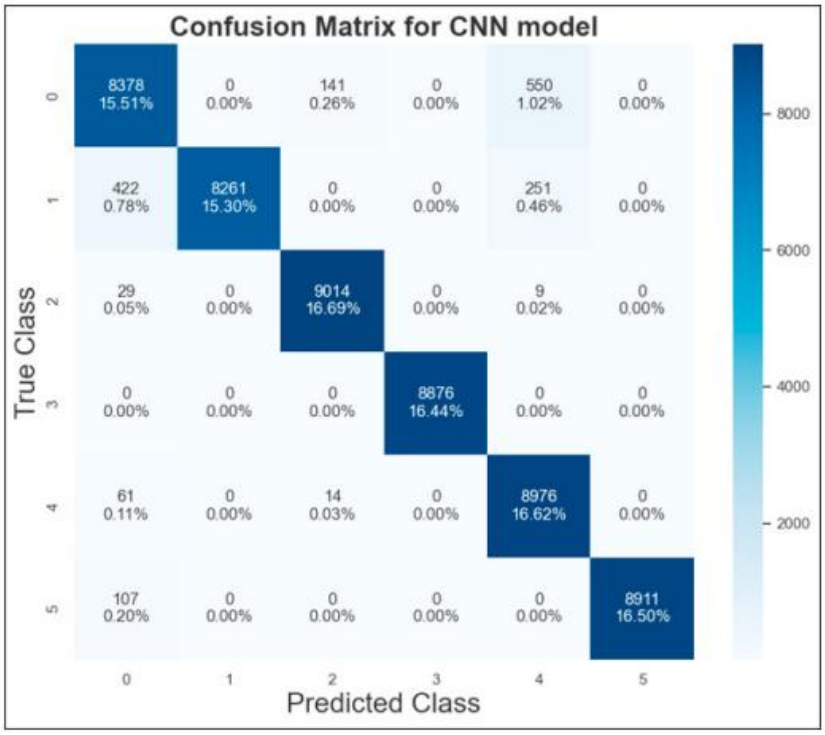

Fig 6: Using the testing dataset, a CNN-derived confusion matrix was used.

5. Conclusion

Improved raw data, better feature selection from complicated data using random forest, and a convolutional neural network (CNN) model for identifying and classifying network threats are all outcomes of the current research. When building the CNN model that relies on deep learning, the random forest technique yielded the most useful features. Even with complex datasets, the suggested CNN model successfully identifies and categorizes network threats. The datasets were tested in 0.21 seconds and trained in 20.66 minutes by the CNN model. The experimental results indicated that the CNN model had an accuracy of 97.07 percent on the testing dataset and 98.80 percent on the training dataset with only forty epochs. As a result, we know that the random forest approach is the most effective way to minimize data dimensionality while still selecting the finest features, all while preserving crucial information. In addition, when it comes to complicated datasets, the deep CNN model outperforms the others with the best detection rate. Businesses can use the findings of the current study to monitor and respond in real time to cyber threats and assaults rather than waiting for the facts to emerge. The results of this research are also crucial for Industry 4.0 since they will help with things like making data-driven choices quickly, responding to new problems as they arise, and adapting to changing market circumstances. Furthermore, the current research has the potential to play a pivotal role in reducing risks to digital information and safeguarding it as the Industry 4.0 environment evolves. The suggested CNN model has the potential to achieve a future accuracy of 100%.

6. References

1. Jaber AN, Rehman SU. FCM–SVM based intrusion detection system for cloud computing environment. Cluster Computing. 2020;23:3221-3231.
2. Atefinia R, Ahmadi M. Network intrusion detection using multi-architectural modular deep neural network. The Journal of Supercomputing. 2021;77:3571-3593.
3. Krishnaveni S, Sivamohan S, Sridhar SS, Prabakaran S. Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing. Cluster Computing. 2021;23(3):1761-1779.
4. Pooja TS, Shrinivasacharya P. Evaluating neural networks using Bi-Directional LSTM for network IDS (intrusion detection systems) in cyber security. Global Transitions Proceedings. 2021;2(2):448-454.
5. Shahzad F, Mannan A, Javed AR, Almadhor AS, Baker T, Al-Jumeily D. Cloud-based multiclass anomaly detection and categorization using ensemble learning. Journal of Cloud Computing. 2022;11(1):1-12.
6. Laghrissi F, Douzi S, Douzi K, Hssina B. Intrusion detection systems using long short-term memory (LSTM). Journal of Big Data. 2021;8(1):65.
7. Fu Y, Du Y, Cao Z, Li Q, Xiang W. A deep learning model for network intrusion detection with imbalanced data. Electronics. 2022;11(6):898.
8. Ashiku L, Dagli C. Network intrusion detection system using deep learning. Procedia Computer Science. 2021;185:239-247.
9. Thirimanne SP, Jayawardana L, Yasakethu L, Liyanaarachchi P, Hewage C. Deep neural network based real-time intrusion detection system. SN Computer Science. 2022;3(2):145.
10. Besharati E, Naderan M, Namjoo E. LR-HIDS: logistic regression host-based intrusion detection system for cloud environments. Journal of Ambient Intelligence and Humanized Computing. 2019;10:3669-3692.
11. Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Al-Nemrat A, Venkatraman S. Deep learning approach for intelligent intrusion detection system. IEEE Access. 2019;7:41525-41550.
12. Gao Y, Liu Y, Jin Y, Chen J, Wu H. A novel semi-supervised learning approach for network intrusion detection on cloud-based robotic system. IEEE Access. 2018;6:50927-50938.

13. Khan AR, Kashif M, Jhaveri RH, Raut R, Saba T, Bahaj SA. Deep learning for intrusion detection and security of Internet of Things (IoT): current analysis, challenges, and possible solutions. Security and Communication Networks. 2022;2022:1-19.
14. Hagar AA, Gawali BW. Apache Spark and deep learning models for high-performance network intrusion detection using CSE-CIC-IDS2018. Computational Intelligence and Neuroscience. 2022;2022:1-14.
15. Mayuranathan M, Saravanan SK, Muthusenthil B, Samydurai A. An efficient optimal security system for intrusion detection in cloud computing environment using hybrid deep learning technique. Advances in Engineering Software. 2022;173:103236.
16. Kim J, Kim J, Kim H, Shim M, Choi E. CNN-based network intrusion detection against denial-of-service attacks. Electronics. 2020;9(6):916.
17. Parampottupadam S, Moldovann AN. Cloud-based real-time network intrusion detection using deep learning. In: 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security). IEEE; 2018. p. 1-8.
18. Archana HPC, Khushi N, Nandini P, Sivaraman, Honnavalli P. Cloud-based network intrusion detection system using deep learning. In: Proceedings of the 7th Annual International Conference on Arab Women in Computing in Conjunction with the 2nd Forum of Women in Research. 2021. p. 1-6.
19. Canadian Institute for Cybersecurity. CSE-CIC-IDS2018 dataset. Available from: UNB CIC IDS 2018 Dataset [Accessed 2023 May 2].
20. Li X, Chen W, Zhang Q, Wu L. Building auto-encoder intrusion detection system based on random forest feature selection. Computers & Security. 2020;95:101851.
21. Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. ACM Computing Surveys. 2009;41(3):1-58. doi:10.1145/1541880.1541882.
22. Cai Z, Zhang Z, Lin M, Zheng Y, Yu Y. A data-driven approach for real-time insider threat detection. IEEE Transactions on Dependable and Secure Computing. 2016;14(2):205-216. doi:10.1109/TDSC.2016.2565566.
23. Xu X, Tian Y. A comprehensive survey of clustering algorithms. Annals of Data Science. 2015;2(2):165-193. doi:10.1007/s40745-015-0040-1.

How to Cite This Article
Sharma A, Soni A, Gupta P, Saxena K. Enhanced Anomaly Detection in Cloud Environments Using Deep Learning. International Journal of Multi Research. 2026; 2(3): 27-33.