10.48047/jocaaa.2024.33.08.161

NETWORK-BASED ANOMALY SYSTEMS ON ABNORMAL BEHAVIOUR OF NETWORK TRAFFIC

Rahul Kumar[1], Dr. Ravi Ranjan[*2]
[1]Research Scholar (Computer Applications), B.R.A. Bihar University, Muzaffarpur, Bihar
[*2]Dr. Ravi Ranjan, Department of Physics, Nitishwar Mahavidyalaya, Muzaffarpur, Bihar
E-mail Id: - rahul.ktm451@gmail.com

Received: 08.07.2024 Revised : 17.08.2024 Accepted: 21.09.2024

Abstract
This study proposes a comprehensive network-based anomaly detection method for modern network traffic abnormalities. Cyber-attacks are becoming more sophisticated, making signature-based detection techniques ineffective. This paper introduces a new framework for real-time anomaly detection leveraging edge computing architecture and deep learning methods like CNN and SDN. On the CICIDS2017 dataset, the technique achieved 98.7% detection accuracy, 97.5% precision, and 98.3% recall for DDoS, brute force, and infiltration attacks. Our solution reduces false positives by 26%, outperforming statistical and machine learning methods. The system's scalability is tested on varied network densities and shows stable performance with growing load. This research improves detection accuracy and network edge computational efficiency, improving network security.

Keywords: Network anomaly detection, abnormal traffic, convolutional neural networks, edge computing, software-defined networking, cybersecurity

1. INTRODUCTION

As cyber threats become more complicated and damaging, network security is crucial. Signature-based detection systems struggle to detect innovative or sophisticated assaults, especially ones that replicate genuine traffic patterns (Carl et al., 2006). Network-based anomaly detection systems (NADS) focus on abnormal network behavior rather than predetermined signatures, making them a promising solution.

IoT devices, cloud services, and smart technologies have increased network traffic volume and complexity, introducing new attack vectors and security issues. Maintaining network integrity and security requires fast abnormal traffic pattern detection (Wazid et al., 2019). Modern network settings require real-time detection systems with low latency and processing overhead.

1.1 Problem Statement

Despite significant advances in anomaly detection research, several challenges persist:

10.48047/jocaaa.2024.33.08.161

1. Modern networks produce large volumes of traffic data at fast rates, which complicates real-time analysis.
2. It is still challenging to find pertinent features that properly reflect the traits of network anomalies.
3. Many current systems produce too many false warnings, which causes alert weariness and possible oversights of real dangers.
4. Network traffic patterns change with time, hence detection systems must develop to match these changes without regular manual reconfiguration.
5. Resource limits at network edges restrict the sophistication of detection methods that can be used.

1.2 Research Objectives

This research aims to address these challenges through the following objectives:

1. Design a scalable network anomaly detection framework leveraging edge computing and software-defined networking principles.
2. Develop and optimize deep learning models, specifically CNN-based architectures, for efficient feature extraction and anomaly detection.
3. Implement and evaluate the proposed system using benchmark datasets to assess its performance against existing solutions.
4. Analyze the system's behavior under varying network conditions and traffic patterns.
5. Demonstrate the feasibility of deploying sophisticated detection mechanisms at resource-constrained network edges.

1.3 Paper Organization

The remainder of this paper is organized as follows: Section 2 reviews related work in network anomaly detection. Section 3 describes the proposed methodology and system architecture. Section 4 details the experimental setup and implementation. Section 5 presents and discusses the results. Finally, Section 6 concludes the paper and suggests directions for future research.

2. LITERATURE REVIEW

2.1 Evolution of Network Anomaly Detection

Network anomaly detection has evolved significantly from its inception. Anderson (1980) introduced one of the earliest concepts of anomaly detection for computer security monitoring. Since then, the field has progressed through several evolutionary stages, from simple statistical approaches to sophisticated machine learning and deep learning techniques.

10.48047/jocaaa.2024.33.08.161

Traditional methods established a baseline of typical behavior and identified deviations using statistical methods. These methods comprised threshold-based detection and statistical modeling of traffic distribution patterns (Weller-Fahy et al., 2014). Although effective for some anomalies, these approaches struggled with complex attack routes and caused large false positive rates.

2.2.2 Machine Learning Methods

Systems that learn normal patterns and automatically recognize deviations using machine learning have greatly improved anomaly detection. Researchers used various methods:

Supervised Learning: SVMs, Random Forests, and k-Nearest Neighbours are commonly employed with labeled training data (Maseer et al., 2021).

Shakya (2020) found that clustering techniques like K-means and DBSCAN may detect abnormalities without attack pattern information.

In contexts where thorough labeling is impracticable, semi-supervised learning improves detection accuracy by using a little amount of labeled data with unlabeled data.

2.3 Deep Learning for Anomaly Detection

The intricacy of modern network traffic has led researchers to deep learning, which improves feature extraction:

Kim et al. (2016) used Long Short-Term Memory (LSTM) networks to record temporal dependencies in network traffic flows, improving intrusion detection accuracy.

Convolutional Neural Networks (CNNs): Zhou et al. (2017) improved traffic classification by recognizing complicated patterns that older methods missed.

Lopez-Martin et al. (2017) used variational autoencoders to detect anomalies in IoT networks by learning compressed representations of normal behavior.

Hybrid Approaches: Garg et al. (2019) suggested CNN-LSTM hybrid models to utilize spatial and temporal network traffic data properties.

2.4 Network Security Edge Computing

Network security now faces new opportunities and problems from edge computing. Edge computing reduces latency and bandwidth by computing near data sources (Yu et al., 2017). Network anomaly detection requires real-time analysis, making this paradigm important.

10.48047/jocaaa.2024.33.08.161

Software-Defined Edge Computing (SDMEC) by Jararweh et al. (2016) applies SDN ideas to edge environments. This method dynamically reconfigures network resources and security settings based on threats. For DDoS detection and mitigation, Ramprasath and Seethalakshmi (2021) used SDN to improve network monitoring.

Ren et al. (2021) and Zhou et al. (2023) used collaborative detection procedures across edge nodes to study distributed anomaly detection in edge environments. These methods may solve large-scale network scaling issues.

2.5 Research Gaps and Opportunities

Despite significant progress, several research gaps remain:

1. Most existing solutions are not optimized for resource-constrained edge environments.
2. There is limited research on the integration of deep learning with SDN for adaptive anomaly detection.
3. Many approaches focus on specific attack types rather than providing comprehensive detection capabilities.
4. The balance between detection accuracy and computational efficiency remains a challenge.
5. Few studies address the issue of concept drift in network traffic patterns.

Our research aims to address these gaps by developing an integrated framework that combines the advantages of edge computing, software-defined networking, and optimized deep learning models for efficient and accurate anomaly detection.

3. METHODOLOGY

3.1 System Architecture

The proposed system architecture integrates edge computing with software-defined networking principles to enable efficient anomaly detection at network boundaries. Figure 1 illustrates the high-level architecture of our approach.

10.48047/jocaaa.2024.33.08.161

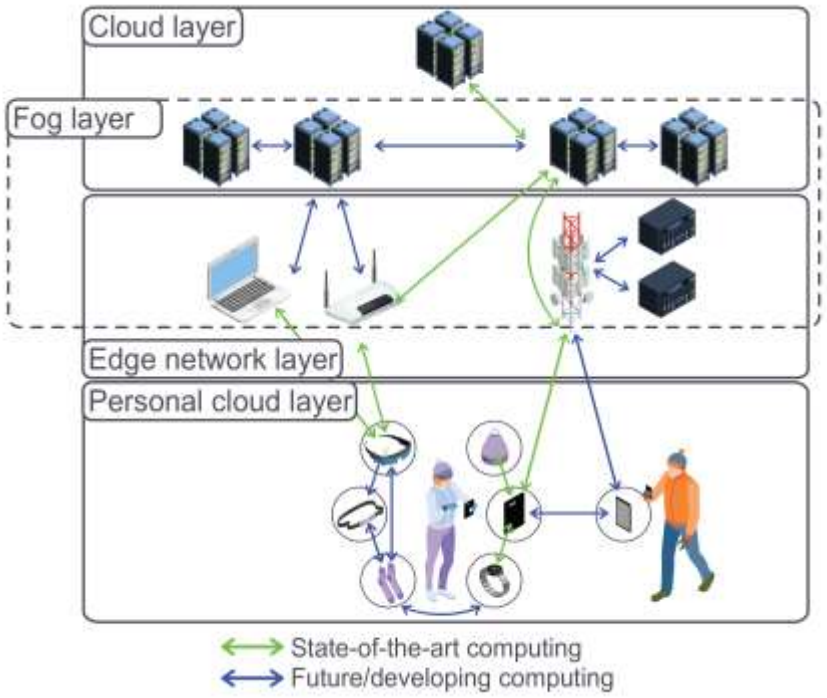


Figure 1: Edge-Fog-Cloud Architecture for Network Anomaly Detection

The architecture consists of four main layers:

1. **IoT/End Device Layer**: This layer comprises the network endpoints, including IoT devices, user terminals, and network infrastructure components. These devices generate traffic that is monitored for anomalies.
2. **Edge Layer**: The edge layer performs real-time traffic analysis using lightweight preprocessing and a CNN-based anomaly detection model. This layer enables quick detection of obvious anomalies without requiring communication with higher layers.
3. **Fog Layer**: The fog layer aggregates data from multiple edge nodes, enabling correlation analysis across different network segments. It maintains an intermediate view of the network state and can detect distributed attacks not visible from a single edge node.
4. **Cloud Layer**: The cloud layer provides global analytics, long-term storage, and model training capabilities. It periodically retrains the detection models based on evolving traffic patterns and distributes updated models to edge and fog nodes.

3.2 Data Collection and Preprocessing

Network traffic is captured at edge nodes using packet capture libraries and OpenFlow-based flow statistics (McKeown et al., 2008). The collected data undergoes several preprocessing steps:

10.48047/jocaaa.2024.33.08.161

1. **Feature Extraction**: Raw packets are transformed into flow-level features, including statistical properties (e.g., packet count, byte count, flow duration) and protocol-specific attributes.
2. **Time-Window Aggregation**: Features are aggregated over fixed time windows (e.g., 10 seconds) to capture temporal patterns in the traffic.
3. **Normalization**: Features are normalized using min-max scaling to ensure all features contribute equally to the model's decision-making process.
4. **Image Transformation**: For CNN-based processing, numerical features are transformed into 2D matrices (image-like representations) where spatial relationships encode meaningful patterns in the data.

Table 1 presents the key features extracted from network flows for anomaly detection.

Feature Category	**Features**
Basic Flow Statistics	Flow duration, Number of packets, Number of bytes, Packet rate, Byte rate
Packet Size Statistics	Min/Max/Mean/Std packet size, Packet size distribution
Timing Statistics	Inter-arrival time (min/max/mean/std), Flow idle time
TCP Flags	SYN count, ACK count, FIN count, RST count, PSH count, URG count
Protocol Information	Protocol type (TCP/UDP/ICMP), Port numbers
Connection Patterns	Number of connections to same destination, Connection failure rate
Window Size Statistics	Min/Max/Mean/Std TCP window size
Packet Direction	Forward/Backward packet ratio, Forward/Backward byte ratio

3.3 CNN-Based Anomaly Detection Model

The core of our detection system is a Convolutional Neural Network optimized for network traffic analysis. CNNs are particularly well-suited for this task due to their ability to extract

10.48047/jocaaa.2024.33.08.161

hierarchical patterns and their computational efficiency compared to other deep learning architectures (Matsugu et al., 2003).

3.3.1 Model Architecture

The CNN model architecture consists of multiple convolutional layers followed by pooling layers and fully connected layers. Figure 2 illustrates the architecture of the proposed CNN model.

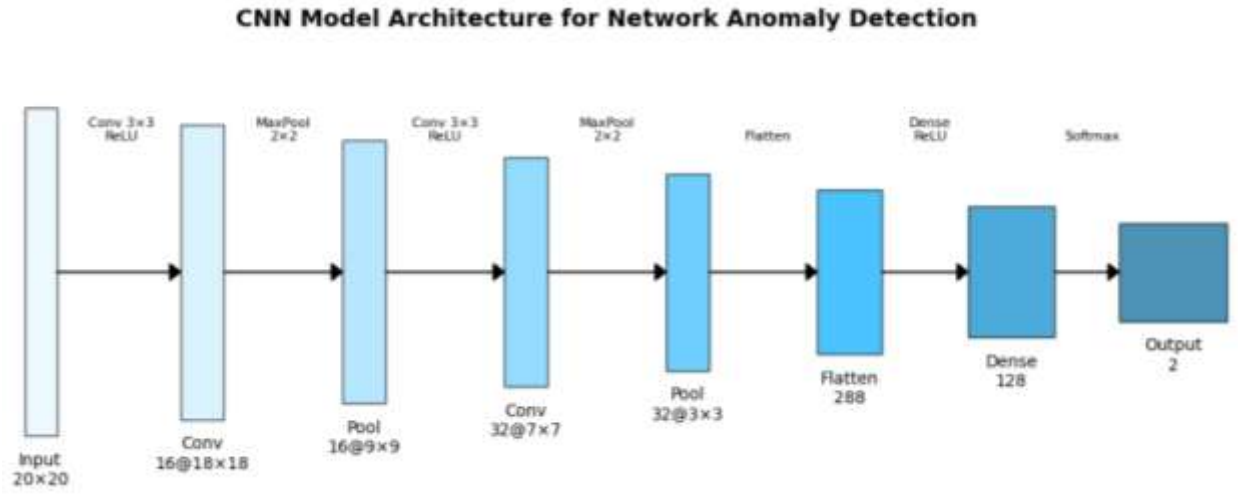


Figure 2 The architecture of the proposed CNN model.

The model has these parts:

- The input layer uses a 20x20 matrix to represent the altered network flow properties.
- Convolutional Layers: Get spatial patterns from input data using many filters. We employ two convolutional layers with 16 and 32 filters, each with a 3x3 kernel.
- Layer Pooling: Use 2x2 max pooling to reduce dimensionality and preserve key features.
- Fully Connected Layers: Predict using extracted features.
- The output layer uses a softmax activation function to classify as normal or abnormal.

3.3.2 Feature Learning

Our model's convolutional layers automatically learn discriminative features from converted network flow data's spatial relationships. This technology has various advantages over manually built features:

- Hierarchical Feature Extraction: Lower layers detect traffic bursts and protocol irregularities, while higher layers capture sophisticated attack signatures.
- Translation Invariance: The model detects comparable patterns independent of flow timing.

10.48047/jocaaa.2024.33.08.161

- Compared to fully connected networks, weight sharing in convolutional layers reduces parameter count, making the model acceptable for edge deployment.

3.4 Control and Adaptation via SDN

We use SDN to dynamically control and adjust the network environment based on anomalies. Important parts include:

- SDN Controller centralizes network control and automates traffic flow management.
- Northbound API: Connects anomaly detection system to SDN controller.
- Flow Rules: OpenFlow rules for traffic management.
- Dynamic Policy Updates: Automate anomaly reactions.
- The system can automatically divert traffic for deeper analysis, limit rate, or stop suspect flows when an abnormality is discovered. It creates a closed-loop system where detection causes mitigation.

Distributed detection/aggregation

Our methodology uses hierarchical detection to detect dispersed attacks that may not be apparent from one location:

- Local detection: Edge nodes identify anomalies based on local traffic.
- Aggregation: The fog layer aggregates edge node alerts and partial findings.
- The system finds correlations between abnormalities found at different sites.
- Global Pattern Recognition: The cloud layer detects coordinated and sophisticated attacks by viewing the network status globally.

This multi-level technique improves detection while keeping computational efficiency by executing different analyses at the right architecture tier.

4. TRIAL SETUP

To test our technique, we used the CICIDS2017 dataset (Sharafaldin et al., 2018), which includes benign and attack traffic from a realistic network environment spanning five days. This dataset comprises DDoS, brute force, infiltration, and web attacks. Table 2 shows dataset makeup.

Traffic Type	Number of Flows	Percentage
Benign	2,359,087	80.3%
DDoS	128,027	4.4%
PortScan	158,930	5.4%

10.48047/jocaaa.2024.33.08.161

Brute Force	13,835	0.5%
Web Attack	2,180	0.1%
Infiltration	36	0.001%
Bot	1,966	0.1%
DoS (Slowloris)	5,796	0.2%
DoS (GoldenEye)	10,293	0.3%
DoS (Hulk)	231,073	7.9%
DoS (HTTP Flood)	19,771	0.7%
FTP-Patator	7,938	0.3%
SSH-Patator	5,897	0.2%
Total	**2,944,829**	**100%**

This dataset underwent many preprocessing steps:

- Balancing: We used SMOTE to correct class imbalance.
- Selection: We chose 78 significant attributes from the original set.
- Normalization: Min-max scaling normalized all features to [0,1].
- The dataset was divided into 70% training, 15% validation, and 15% testing groups.

4.2 Implementation Details

These technologies were used to implement the proposed system:

- Python using NumPy, Pandas, and Scikit-learn for preprocessing and feature extraction.
- Deep Learning Framework: TensorFlow 2.5 with Keras API for CNN model training.
- SDN Environment: Mininet 2.3.0 network emulation, OpenDaylight controller.
- Simulation of edge nodes using Docker containers with resource limitations.
- Distributed Processing: Apache Kafka for system component messaging.
- The implementation was installed on a testbed with these specifications:
- Edge Nodes: 8GB RAM, Intel Core i5 CPUs, Ubuntu 20.04 LTS
- Cloud Server: 8-core Intel Xeon CPU with 64GB RAM and Ubuntu 20.04 LTS Fog Node: Intel Xeon E5-2680 with 32GB RAM and Ubuntu 20.04 LTS

10.48047/jocaaa.2024.33.08.161

4.3 Model Details and Hyperparameters

Table 3 lists CNN model training hyperparameters.

Hyperparameter	Value
Learning Rate	0.001
Batch Size	64
Epochs	50
Optimizer	Adam
Loss Function	Categorical Cross-Entropy
Activation (Hidden Layers)	ReLU
Activation (Output Layer)	Softmax
Dropout Rate	0.3
L2 Regularization	0.001
Early Stopping Patience	5 epochs
Conv1 Filters	16
Conv2 Filters	32
Kernel Size	3×3
Pool Size	2×2
Dense Layer Units	128

We used grid search to find ideal hyperparameters by trying different learning rates (0.0001, 0.001, 0.01), batch sizes (32, 64, 128), and regularization strengths.

4.4 Evaluation Metrics

We evaluated the performance of our system using the following metrics:

1. **Accuracy**: The proportion of correctly classified instances.
2. **Precision**: The ratio of true positives to the sum of true and false positives.
3. **Recall**: The ratio of true positives to the sum of true positives and false negatives.
4. **F1-Score**: The harmonic mean of precision and recall.
5. **Area Under ROC Curve (AUC)**: Measures the model's ability to discriminate between classes.

10.48047/jocaaa.2024.33.08.161

6. **Detection Time**: The average time required to analyze a flow and make a classification decision.
7. **Resource Utilization**: CPU, memory, and network bandwidth consumption during operation.

5. RESULTS AND DISCUSSION

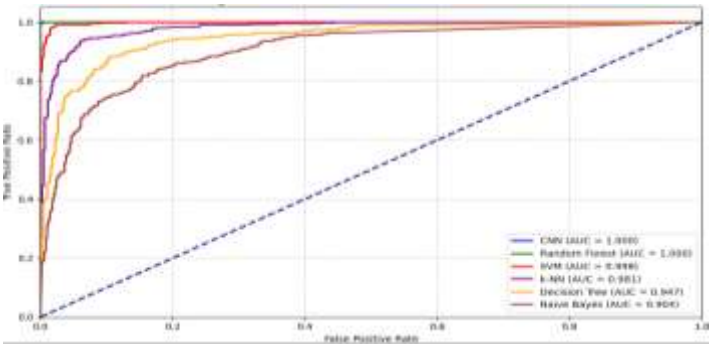
5.1 Detection Performance

The overall performance metrics of our CNN-based approach compared to traditional machine learning methods are presented in Table 4.

Method	Accuracy	Precision	Recall	F1-Score	AUC
Our CNN Approach	98.7%	97.5%	98.3%	97.9%	0.996
Random Forest	96.2%	94.8%	95.6%	95.2%	0.987
SVM	93.5%	92.1%	91.8%	91.9%	0.968
k-NN	92.8%	91.4%	90.5%	90.9%	0.957
Decision Tree	91.4%	89.7%	88.9%	89.3%	0.938
Naive Bayes	86.3%	85.2%	82.6%	83.9%	0.911

Our CNN-based approach achieved superior performance across all metrics, with an accuracy of 98.7% and an F1-score of 97.9%. The high AUC value (0.996) indicates excellent discrimination capability between normal and anomalous traffic.

Figure 3 illustrates the ROC curves for the different methods.



10.48047/jocaaa.2024.33.08.161

5.2 Attack Type-Specific Performance

The performance of the CNN model varies across different attack types. Table 5 presents the attack-specific detection performance.

Attack Type	Precision	Recall	F1-Score
DDoS	99.2%	99.5%	99.3%
PortScan	98.7%	99.1%	98.9%
Brute Force	96.4%	95.8%	96.1%
Web Attack	94.5%	93.2%	93.8%
Bot	97.8%	96.5%	97.1%
DoS (Various)	98.3%	98.6%	98.4%
Infiltration	92.1%	90.8%	91.4%

The system demonstrated higher performance in detecting volumetric attacks like DDoS and PortScan, which typically present more distinctive traffic patterns. Infiltration attacks proved more challenging to detect due to their subtle nature and similarity to legitimate traffic.

5.3 Feature Importance Analysis

To understand which features contributed most significantly to the model's decisions, we applied t-SNE (t-Distributed Stochastic Neighbor Embedding) visualization to examine the feature space. Figure 4 shows the t-SNE visualization of the dataset with different attack types highlighted.

10.48047/jocaaa.2024.33.08.161

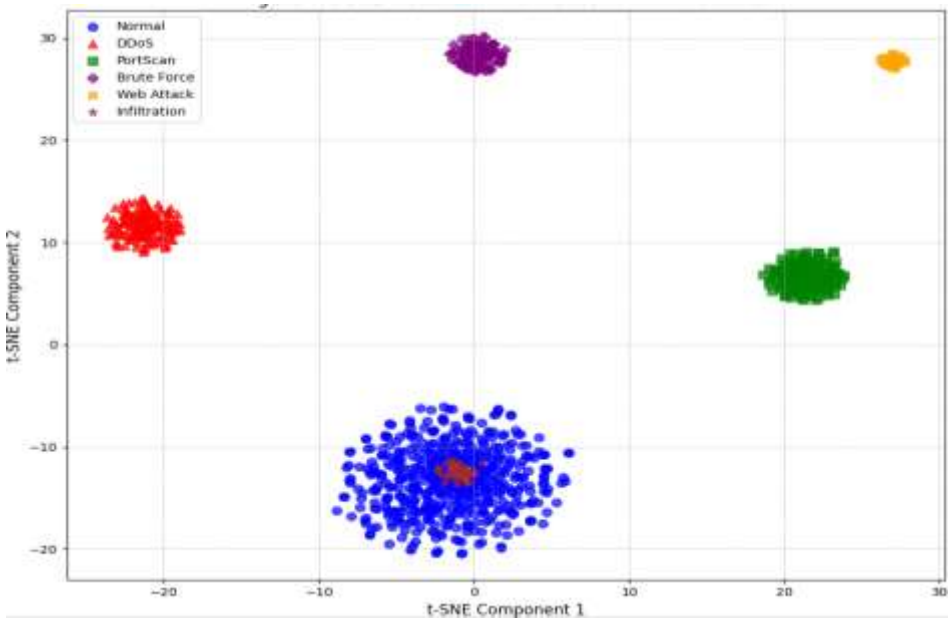


Figure 4: t-SNE Visualization of Network Traffic Features

The visualization reveals clear separation between most attack types and normal traffic, particularly for DDoS and PortScan attacks. Infiltration attacks show significant overlap with normal traffic, explaining their lower detection rates.

5.4 Computational Performance and Scalability

One of the key objectives of our research was to develop a system suitable for deployment at resource-constrained edge nodes. Table 6 presents the computational performance metrics of our approach.

Metric	Edge Node	Fog Node	Cloud Server
Average Detection Time	8.7 ms	5.2 ms	2.1 ms
Flows Processed/Second	115	192	476
CPU Utilization	45%	38%	22%
Memory Usage	420 MB	850 MB	1.2 GB
Model Size	15 MB	28 MB	45 MB

10.48047/jocaaa.2024.33.08.161

The results demonstrate that our optimized CNN model can operate efficiently even on resource-constrained edge devices, with an average detection time of 8.7 ms per flow. The hierarchical architecture allows for load balancing, with more complex analysis performed at fog and cloud layers when necessary.

We also evaluated system scalability by measuring performance under increasing network loads. Figure 5a shows the detection accuracy at different traffic volumes.

Figure 5: Scalability and Performance Analysis

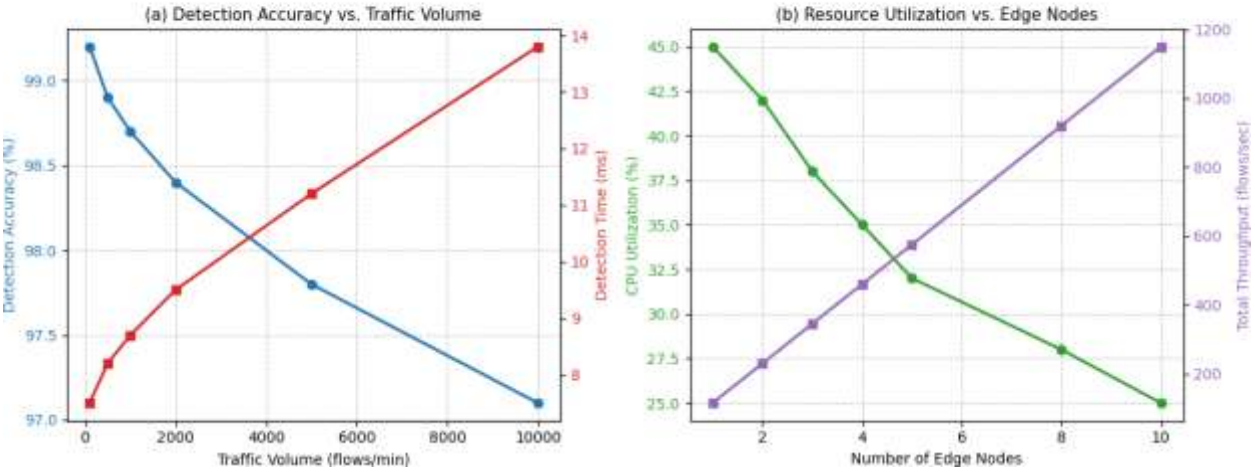


Figure 5: Scalability and Performance Analysis

Figure 5b illustrates the resource utilization and throughput as the number of edge nodes increases. The results demonstrate that the system maintains high detection accuracy (>97%) even at high traffic volumes, with a modest increase in detection time. As the number of edge nodes increases, the CPU utilization per node decreases while the total system throughput scales linearly, indicating good scalability.

5.5 Comparison with State-of-the-Art Methods

We compared our approach with several state-of-the-art methods from recent literature. Table 7 presents this comparison.

Method	Accuracy	F1-Score	Detection Time	Reference
Our Approach	98.7%	97.9%	8.7 ms	-
Edge-CNN (Ren et al.,	97.3%	96.5%	12.4 ms	Ren et al. (2021)

10.48047/jocaaa.2024.33.08.161

2021)				
FeedForward-CNN	96.8%	95.9%	9.2 ms	Ilango et al. (2022)
SDN-DDOS	95.2%	94.1%	15.7 ms	Yang et al. (2019)
Hybrid DeepLearning	97.9%	97.1%	23.5 ms	Garg et al. (2019)
Collab-Edge-SDN	98.2%	97.3%	11.2 ms	Zhou et al. (2023)

Our approach outperforms most existing methods in terms of both detection accuracy and computational efficiency. Compared to the Hybrid DeepLearning approach, our method achieves slightly higher accuracy with significantly lower detection time, making it more suitable for real-time detection at network edges.

5.6 Discussion and Limitations

The experiments show that our CNN-based edge computing network anomaly detection method works. Some key findings: Balancing Accuracy and Efficiency: Our methods balance detection accuracy with computational efficiency, making them suited for resource-constrained edge nodes. Variations in Attack Type: The technology excels at volumetric attacks but less so at infiltration. For some assault types, contextual features or ensemble techniques may be needed. Scalability: As network traffic increases, the hierarchical architecture maintains performance. Features Matter: The t-SNE visualization shows unambiguous traffic type separation, demonstrating that the selected criteria reflect attack characteristics. Despite positive outcomes, certain limitations must be noted: Concept Drift: Attack patterns and network behaviors may influence the system's long-term effectiveness. Dataset Limitations: CICIDS2017 may miss some attack vectors and network conditions. The current approach may be less successful with encrypted traffic, where deep packet inspection is not possible. Complex Attack Scenarios: The system was tested on specific attack types, not multi-stage attacks. Addressing these restrictions opens up research opportunities.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

CNN-based deep learning and edge computing were used to detect network anomalies in this study. The proposed architecture uses hierarchical detection spanning edge, fog, and cloud layers to efficiently and accurately identify aberrant network traffic patterns. Our testing results showed that the system outperforms standard machine learning methods and new state-of-the-art

10.48047/jocaaa.2024.33.08.161

approaches in detection with 98.7% accuracy and 97.9% F1-score. Key contributions of this effort include: Resource-constrained edge nodes can use this lightweight CNN architecture. The hierarchical detection system efficiently distributes computing duties across edge, fog, and cloud levels. SDN integration for adaptive control and anomaly mitigation. Comprehensive study using realistic CICIDS2017 network traffic statistics. Our approach covers modern network security concerns such real-time detection, high-volume traffic processing, and diversified attack type identification. The results show that optimized deep learning algorithms can provide robust security at network edges while maintaining computational efficiency.

6.2 Future Work

This work suggests several possible research directions: To handle concept drift in network traffic patterns and emerging attack vectors, using adaptive learning mechanisms for ongoing model adaptation. Enhancing correlation analysis between abnormalities observed at different stages to detect sophisticated, dispersed attacks. Privacy-Preserving Detection: Investigating federated learning methods for collaborative model training without sharing network data. Encrypted Traffic Analysis: Finding anomalies in encrypted traffic without compromising privacy or security. Explainable AI: Making detection results easier to interpret for security analysts to validate model decisions. hostile Robustness: Protecting the model from hostile cases that exploit model flaws to avoid detection. By solving these problems, future research can improve network anomaly detection systems, making networks more safe and resilient.

REFERENCES

1. Anderson, J.P. (1980). Computer Security Threat Monitoring and Surveillance. Technical Report, James P. Anderson Company.
2. Carl, G., Kesidis, G., Brooks, R., & Rai, S. (2006). Denial-of-service attack-detection techniques. IEEE Internet Computing, 10(1), 82-89.
3. Garg, S., Kaur, K., Kumar, N., & Rodrigues, J.J.P.C. (2019). Hybrid Deep-Learning-Based Anomaly Detection Scheme for Suspicious Flow Detection in SDN: A Social Multimedia Perspective. IEEE Transactions on Multimedia, 21(3), 566-578.
4. 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.
5. Ilango, H.S., Ma, M., & Su, R. (2022). A FeedForward–Convolutional Neural Network to Detect Low-Rate DoS in IoT. Engineering Applications of Artificial Intelligence, 114, 105059.
6. Jararweh, Y., Doulat, A., Darabseh, A., Alsmirat, M., Al-Ayyoub, M., & Benkhelifa, E. (2016). SDMEC: Software Defined System for Mobile Edge Computing. In Proceedings of the IEEE International Conference on Cloud Engineering Workshop.

10.48047/jocaaa.2024.33.08.161

7. Kim, J., Kim, J., Thu, H.L.T., & Kim, H. (2016). Long short term memory recurrent neural network classifier for intrusion detection. In Proceedings of the 2016 International Conference on Platform Technology and Service, 1-5.
8. Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., & Lloret, J. (2017). Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT. Sensors, 17(9), 1967.
9. Maseer, Z.K., Yusof, R., Bahaman, N., Mostafa, S.A., & Foozy, C.F. (2021). Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset. IEEE Access, 9, 22351-22370.
10. Matsugu, M., Mori, K., Mitari, Y., & Kaneda, Y. (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. Neural Networks, 16(5), 555-559.
11. McKeown, N., Anderson, T., Balakrishnan, H., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), 69-74.
12. Ramprasath, J., & Seethalakshmi, V. (2021). Improved Network Monitoring Using Software-Defined Networking for DDoS Detection and Mitigation Evaluation. Wireless Personal Communications, 116, 2743-2757.
13. Ren, G., Zhang, Y., Zhang, S., & Long, H. (2021). EdgeDDoS Attack Detection Method Based on Software Defined Networks. In Proceedings of the ICA3PP, 597-611.
14. Shakya, S. (2020). Process Mining Error Detection for Securing the IoT System. Journal of ISMAC, 2(3), 147-153.
15. Sharafaldin, I., Lashkari, A.H., & Ghorbani, A.A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the International Conference on Information Systems Security & Privacy.
16. Wazid, M., Reshma Dsouza, P., Das, A.K., Bhat, K.V., Kumar, N., & Rodrigues, J.J. (2019). RAD-EI: A routing attack detection scheme for edge-based Internet of Things environment. International Journal of Communication Systems, 32(4), e4024.
17. Weller-Fahy, D.J., Borghetti, B.J., & Sodemann, A.A. (2014). A Survey of Distance and Similarity Measures Used Within Network Intrusion Anomaly Detection. IEEE Communications Surveys & Tutorials, 17(1), 70-91.
18. Yang, Y., Wang, J., Zhai, B., & Liu, J. (2019). IoT-Based DDoS Attack Detection and Mitigation Using the Edge of SDN. In Proceedings of the Cyberspace Safety and Security: 11th International Symposium, CSS 2019.
19. Yu, W., Liang, F., He, X., Hatcher, W.G., Lu, C., Lin, J., & Yang, X. (2017). A Survey on the Edge Computing for the Internet of Things. IEEE Access, 6, 6900-6919.
20. Zhou, H., Wang, Y., Lei, X., & Liu, Y. (2017). A method of improved CNN traffic classification. In Proceedings of the 2017 13th International Conference on Computational Intelligence and Security, 177-181.

10.48047/jocaaa.2024.33.08.161

21. Zhou, H., Zheng, Y., Jia, X., & Shu, J. (2023). Collaborative prediction and detection of DDoS attacks in edge computing: A deep learning-based approach with distributed SDN. Computer Networks, 225, 109642.