Journal of Environmental Protection and Ecology 25, No 5, 1371–1383 (2024)

Risk assessment

CONVOLUTIONAL NEURAL NETWORK-BASED ATTACK DETECTION FOR SECURE DATA TRANSMISSION IN THE INTERNET OF THINGS

Y. ANGELINE CHRISTOBEL[a]*, MAMTA KUMARI[b], SWAPNIL PARIKH[c], D. RAJENDRA PRASAD[d], VAIBHAV BAPURAO SONULE[e], V. VANI[f], P. SELVAN[g], R. PREMKUMAR[h], A. RAJARAM[i]

[a]*Department of Computer Science, Hindustan College of Arts and Science, 603 103 Chennai, Tamil Nadu, India*
E-mail: *yangelinechristobel@gmail.com*
[b]*Department of Computer Science and Engineering (AIML), Panipat Institute of Technolgy, 132 103 Samalkha Panipat, Haryana, India*
[c]*Department of Computer Science and Engineering, Parul Institute of Engineering and Technology, Parul University, 391 760 Waghodia, Gujarat, India*
[d]*Department of Electronics and Communication Engineering, St. Ann's College of Engineering and Technology, 523 187 Chirala, Andhra Pradesh, India*
[e]*Faculty of Law, Symbiosis Law School (SLS) Symbiosis International, (Deemed University) (SIU), Vimannagar, 411 014 Pune, Maharashtra, India*
[f]*Department of Information Science and Engineering, Bangalore Institute of Technology, 560 004 Bangalore, Karnataka, India*
[g]*Department of Electrical and Electronics Engineering, Erode Sengunthar Engineering College, 638 057 Erode, Tamil Nadu, India*
[h]*Department of Electronics and Instrumentation Engineering, Sri Sairam Engineering College, 600 044 Chennai, Tamil Nadu, India*
[i]*Department of Electronics and Communication Engineering, E.G.S. Pillay Engineering College, 611 002 Nagapattinam, Tamil Nadu, India*

Abstract. The problem statement revolves around the vulnerability of IoT networks to various forms of attacks, including intrusion attempts, data tampering, and unauthorised access. Because IoT environments are dynamic and diversified, traditional security techniques frequently fall short of providing effective protection. Therefore, there is an urgent need for sophisticated detection methods that can reliably identify malicious activity among the enormous amounts of data that are transmitted over Internet of Things networks. The proposed system aims to enhance the security of IoT systems by effectively identifying malicious activities, thereby mitigating potential threats to data integrity and privacy. The proposed system leverages Convolutional neural network (CNN), a class

* For correspondence.

of deep learning models known for their effectiveness in image recognition and pattern detection tasks, to analyse the temporal and spatial characteristics of data traffic in IoT networks. By training the CNN on labelled datasets containing both normal and anomalous network activities, the system learns to distinguish between benign and malicious behaviour. The flow of the system involves preprocessing raw data, extracting relevant features, training the CNN model, and deploying it for real-time attack detection during data transmission. The effectiveness of the CNN-based technique in precisely and recall rates identifying different kinds of attacks is demonstrated by experimental findings. The solution demonstrates strong performance in various IoT deployment scenarios and adeptly adjusts to changing threats. By guaranteeing the integrity and confidentiality of sent data, this proposed approach greatly improves the security posture of IoT ecosystems through proactive detection capabilities.

Keywords: deep learning, convolutional neural network, data transmission, internet of things, security.

AIMS AND BACKGROUND

In recent years, the escalating concern over data security has underscored the critical need for robust communication systems. The proliferation of smart devices interconnected for communication, computation, and real-time monitoring has significantly expanded the attack surface for malicious intruders seeking to exploit vulnerabilities for personal gain[1]. Particularly in the realm of Industrial Internet of Things (IoT), the emergence of Internet of vehicles (IoV) as a pivotal component of Intelligent transportation systems (ITS) has brought forth pressing security challenges. Ensuring the security of both users and infrastructures within IoV deployments is paramount, necessitating the implementation of effective Intrusion Detection systems (IDS) to safeguard against malware-driven attacks[2]. Wireless sensor networks (WSN), integral to the IoT ecosystem, present unique security considerations due to their resource-constrained nature and self-configuring capabilities[3]. The proliferation of IoT devices has spurred a surge in cyber-attacks, fuelling research efforts to enhance anomaly detection mechanisms and fortify security[4]. Moreover, the IoT paradigm offers promising opportunities for services and products, but the heterogeneity of IoT devices exposes inefficiencies in traditional network structures, highlighting the need for more flexible architectures like Software-defined networking (SDN). Recent events have illustrated the vulnerability of IoT devices to large-scale Distributed denial of service (DDoS) attacks, challenging conventional mitigation strategies due to the overwhelming volume of data generated by IoT botnets[5]. In the realm of cybersecurity within the Internet of Things (IoT), five significant research papers have made notable contributions. The first study focuses on leveraging AI algorithms for botnet detection, addressing the critical challenge of identifying and mitigating botnet attacks to enhance network security[6]. Following this, the second paper delves into the efficacy of machine learning (ML) models in combating various cyber threats prevalent in IoT ecosystems, including intrusion, malware, spam, and Android malware detection[7]. Transitioning from ML models to specific machine learning techniques, the third paper examines

the use of Gradient Boosting, Logistic Regression, and Random Forest classifiers to detect threats targeting IoT firmware, aiming to bolster the security of IoT edge devices[8]. Building upon this research, the fourth paper employs a range of ML algorithms such as support vector machine and decision tree for intrusion detection in IoT networks, achieving remarkable accuracy in detecting botnet attacks and enhancing overall network security[9]. Lastly, the fifth paper introduces a novel machine learning solution utilising stacked Long short-term memory (LSTM) networks, adept at capturing temporal dependencies and complex patterns in IoT network data, thereby bolstering the integrity and resilience of IoT infrastructures against diverse attacks[10]. Collectively, these studies underscore the pivotal role of machine learning in fortifying IoT cybersecurity, showcasing a diverse array of approaches to detect and mitigate threats, ultimately contributing to a safer and more secure IoT landscape. Through this research, we aim to bolster the security of IoT ecosystems, mitigating the risk of cyber-attacks and safeguarding critical infrastructure and user data.

EXPERIMENTAL

Dataset. The dataset used in this study was sourced from Kaggle, which used the Distributed Smart space orchestration system (DS2OS) to establish a virtual Internet of Things environment to produce synthetic data. Micro-services in the architecture communicate with each other over the Message queuing telemetry transport (MQTT) protocol. 357 952 samples and 13 attributes make up the dataset; spanning eight classes, 347 935 samples are categorised as normal data and 10 017 as anomalous data[11] and are shown in Table 1.

Table 1. Distribution of attacks and anomalies in the dataset

Type of attack	Samples count
Wrong setup	122
Spying	532
Scan	1547
Malicious operation	805
Malicious control	889
Data Type probing	342
DoS	5780

The comprehensive data for the proposed system's input dataset are available at https://www.kaggle.com/datasets/francoisxa/ds2ostraffictraces.

Data processing. Before the dataset is examined by the CNN model, it must first undergo data preprocessing. Several crucial actions must be taken in order at this phase to guarantee that the data are prepared for model input by cleaning, standardising, and optimising them[12].

Handling missing values. Recognising and fixing missing values in the dataset is the first stage. Mode (I) can be used to represent the feature with missing values for categorical characteristics such as "Accessed Node Type", where missing values can be substituted with the mode of the corresponding feature. There are several ways to impute missing values for numerical attributes like "Value", including mean (mean (I)), median (median (I)), and interpolation.

Outlier detection and removal. The CNN model's performance can be considerably impacted by outliers. statistical techniques like *Z*-score computation:

$$Z = (1 - \mu)/\sigma. \quad (1)$$

To find outliers, σ stands for the standard deviation, μ is the mean, and I refers to the data point. Using logarithmic transformation, outliers are either eliminated from the dataset.

Noise reduction. Domain expertise and data exploration are used to address noise in the dataset, such as errors or features that are not relevant. Features that do not improve the predicting ability of the model are eliminated[13]. To guarantee data accuracy and integrity, errors are fixed in the data using the proper methods.

Data transformation. Numeral formats such as one-hot encoding or label encoding are used to transform categorical values. Each category is given a unique numerical value via label encoding, whereas one-hot encoding generates binary columns for every category[14]. To guarantee consistent scale and distribution throughout the dataset, numerical features are normalised, also known as standardised. When features are normalised, they are scaled to a predetermined range, usually between 0 and 1, as opposed to being standardised, which sets the features' mean and standard deviation to 0 (Ref. 15).

Handling imbalanced classes. Oversampling and undersampling strategies are used to correct the dataset's class imbalance. By interpolating between current samples, oversampling approaches like the Synthetic minority over-sampling technique (SMOTE) artificially create samples for minority classes. SMOTE creates synthetic samples along the line segments that connect each minority class sample to its closest neighbours by calculating the Euclidean distance between each sample and its neighbor. SMOTE uses the following formula to produce fresh samples mathematically:

$$\text{new\_sample} = \text{sample} + \text{random\_number} \times (\text{neighbour} - \text{sample}). \quad (2)$$

On the other hand, in order to achieve a balanced representation of classes, undersampling approaches randomly eliminate samples from majority classes. To do this, subsets of samples from the majority class are chosen at random until the intended class balance is reached. In order to reduce the possibility of bias towards the majority class and enhance the model's capacity to correctly classify

instances from minority classes, oversampling and undersampling techniques work together to ensure that all classes are fairly represented in the training data. These preprocessing procedures help to improve and optimise the dataset so that it may be fed into the CNN model.

Feature extraction. When pre-processed data are being fed into a Convolutional Neural Network, feature extraction is an essential step in the process. To capture temporal and spatial properties in the context of IoT network data, pertinent features are retrieved, which helps the CNN model detect unusual network activity.

While spatial characteristics are related to the spatial distribution and relationships among data elements, temporal characteristics are patterns and trends in data over time. These properties are captured and represented in a way that is appropriate for CNN analysis through the use of feature extraction techniques. Time-series decomposition, which splits the data into trend, seasonal, and residual components, is a popular feature extraction technique for temporal data. This method contributes to the identification of long-term trends and recurring patterns in the data, offering important insights for anomaly detection. Relevant features can be extracted from spatial data using methods like spatial aggregation and frequency analysis. In order to identify patterns and relationships in space, data are summarised across many spatial dimensions (such as regions and clusters) in the process of spatial aggregation. In contrast, frequency analysis examines the frequency distribution of data points or features.

Autocorrelation. A time series' association with a lag version of itself is measured by autocorrelation. Calculating the autocorrelation function R_k lag k mathematically is as follows:

$$R_k = \frac{\sum_{t=1}^{N-k} (i_t - \bar{i})\,(i_{t+k} - \bar{i})}{\sum_{t=1}^{N} (i_k - \bar{i})^2}, \tag{3}$$

where N is the number of observations, i_t and $\bar{i}$ – the time series value and its mean, respectively.

Fast Fourier Transform (*FFT*). A time-domain signal is transformed into its frequency-domain representation via FFT. The FFT formula is provided by:

$$I(f) = \int_{-\infty}^{\infty} i(t)\, e^{-k2\pi ft}\, \mathrm{d}t, \tag{4}$$

where $I(f)$ is the frequency-domain representation of the signal, $i(t)$ – the time-domain signal, f – the frequency, and k – the imaginary unit.

Relevant features that capture temporal and spatial properties are extracted, giving the CNN model useful information it can use to recognise unusual network activity in Internet of Things networks. This makes it possible for the model to

identify departures from typical behaviour and notify users of any potential security risks.

Training the model. There are multiple important processes in our proposed system that is involved in training the CNN model, namely DenseNet-169. First, essential features that represent the temporal and spatial aspects of data traffic in IoT networks are extracted from the pre-processed data. These characteristics feed into the DenseNet-169 model, speeding up its learning process[13].

In training, labelled datasets with both typical and anomalous network activity are input into the DenseNet-169 model. Every sample in the collection includes features that have been extracted and a class label that indicates whether the behaviour is harmful or normal. Through an iterative process called backpropagation, the model learns to differentiate between these classes by minimising the difference between the anticipated and actual class labels. Usually, a loss function, like categorical cross-entropy, which measures the difference between ground-truth and predicted labels, directs this optimisation process. The DenseNet-169 model gradually gains the ability to identify patterns and features that point to various attack types, including intrusion attempts, data manipulation, and illegal access, as training goes on. The DenseNet architecture's dense connectivity pattern makes it easier to reuse features and increase gradient flow throughout the network, which helps it identify intricate relationships between features and boost overall performance.

Input layer. The features that are extracted from the pre-processed data are sent into the DenseNet-169 model's input layer. These characteristics record the spatial and temporal properties of data flow in Internet of Things networks, giving higher layers useful data. Though it does not execute any calculations, the input layer prepares the network for feature propagation.

Convolutional layers. The core components of the DenseNet-169 design are the convolutional layers. These layers extract hierarchical representations of the data by applying convolution operations to the input features. DenseNet-169 employs Batch normalisation (BN) and Rectified linear unit (ReLU) activation functions after each convolutional layer, respectively, to normalise and inject non-linearity into the network. An expression for the general equation of a convolution operation is:

$$Z_l = W_l A_{l-1} + Bs_l, \tag{5}$$

where Z_l represents the output of the convolutional layer; W_l – the filter weights; A_{l-1} – the input from the previous layer; Bs_l the bias term.

Dense blocks. DenseNet-169 makes use of dense blocks to enable dense layer connectivity. Each layer in a dense block delivers its output to all following layers in the same block after receiving input from all previous layers. The network's representational power is increased by its dense connectivity, which promotes

feature reuse and gradient flow. A dense block's output equation can be recursively defined as:

$$Y_l = D(I_{l-1}, F_l(I_{l-1})), \tag{6}$$

where Y_l represents the output of the dense block; F_l – the composite function of convolution, BN, and ReLU operations within the block; D – the concatenation operation.

Transition layers. In order to minimise computing complexity and regulate the spatial dimensions of feature maps, transition layers are placed in between dense blocks. Usually, 2 × 2 average pooling operations come after 1 × 1 convolutional layers in these layers. In order to facilitate the network's effective learning of hierarchical representations, the transition layers downsample the feature maps. An expression for a transition layer's output equation is as follows:

$$A_l = \text{avg pooling}(Z_l). \tag{7}$$

*Global average pooling***.** A global average pooling layer is used to combine spatial data from all feature maps after the final dense block. A fixed-length feature vector is produced by calculating the average value of each feature map. One way to express the global average pooling equation is as:

$$A_l = (1/N) \sum_{k=1}^{N} A_{\mathrm{L}}{}^k, \tag{8}$$

where A_l represents the output of the global average pooling layer; $A_{\mathrm{L}}{}^k$ – the k-th feature map in the final layer; N – the total number of feature maps.

Fully connected layer. In order to perform classification based on the learned features, a fully linked layer is added to the network. Usually, this layer is made up of many neurons, each of which represents a class label. A softmax activation function is applied to the fully connected layer's output in order to calculate the probability distribution across the classes. The completely linked layer's output after softmax activation can be represented by the following equation:

$$Z_{l+1} = W_{L+1} A_l + Bs_{L+1} \tag{9}$$

$$\hat{Y} = \text{SoftMax}(Z_{l+1}), \tag{10}$$

where Z_{l+1} represents the pre-activation values; W_{L+1} – the weight matrix; Bs_{L+1} – the bias vector; $\hat{Y}$ – the predicted probability distribution over classes.

During the training phase, the DenseNet-169 model's parameters are iteratively updated in an effort to reduce the difference between the predicted and actual class labels. Typically, gradient descent and backpropagation algorithms are used for this optimisation. The gradients of the loss function with respect to these parameters are used to modify the parameters, which include filter weights and biases.

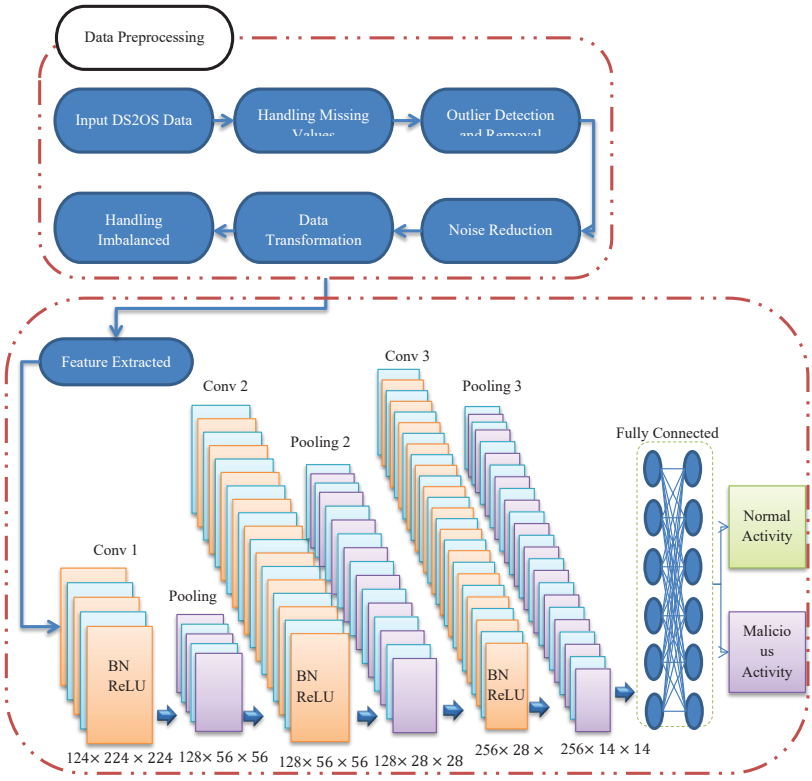

Fig. 1. Proposed DenseNet-169 model

In summary, labelled datasets including both typical and abnormal network activity are fed into the DenseNet-169 model during training as shown in Fig. 1. The convolutional layers, dense blocks, transition layers, and fully connected layers of the DenseNet architecture are used to process these datasets. The model learns to differentiate between several classes of network activity and precisely forecast the occurrence of abnormalities through backpropagation and optimisation techniques. The DenseNet-169 model can recognise network anomalies with high accuracy and robustness by continuously improving and fine-tuning it, which enhances the overall security of IoT infrastructures.

Real-time attack detection. The trained CNN model works inside the IoT network infrastructure to assess incoming data streams as they are transferred in order to detect attacks in real-time. The temporal and spatial properties of the data are continuously assessed by the model, which searches for patterns suggestive of questionable or malevolent activity. The CNN model quickly evaluates each data packet as it passes through the network, using the learnt representations to spot anomalies or departures from typical behaviour. The concept is designed to identify possible threats and initiate suitable reaction mechanisms, including

traffic rerouting, alert notifications, or access restriction, to lessen the impact of an attack. The CNN approach allows for proactive intervention to intercept possible attacks before they may undermine the integrity or security of the Internet of Things network, thanks to its real-time operation. This method guarantees the continuing and uninterrupted functioning of vital services and applications while also improving the overall security posture of IoT infrastructures.

RESULTS AND DISCUSSION

We evaluated our proposed attack detection system for Internet of Things networks using Python 3.6.9 and Keras 2.2.5 in the Google Colab environment with the goal of determining how effective it is in thwarting potential threats. The system was tested on an Intel Xeon CPU running Ubuntu 18.04.3 LTS with a clock speed of 2.20GHz and 12.48GB of RAM. We used an assessment measures to determine how well our system was performing.

A confusion matrix shows how the model's predictions and the actual labels are compared as shown in Table 2.

Table 2. Confusion matrix

Symptoms	Predicted: Normal	Predicted: Malicious activity
ACTUAL: Normal	True Negative (TN)	False Positive (FN)
ACTUAL: Malicious Activity	False Negative (FN)	True Positive (TN)

Accuracy is defined as the ratio of accurately predicted instances (TP and TN) to the total number of instances. The percentage of true positive forecasts among all positive predictions is known as precision. The model's recall is its capacity to accurately identify every positive instance TP and FN included in the data. The F1-score provides a fair evaluation of the model's performance by taking the mean of precision and recall.

$$\text{Accuracy} = (TP + TN)/(TP + TN + FP + FN) \tag{11}$$

$$\text{Precision} = TP/(TP + FP) \tag{12}$$

$$\text{Recall} = TP/(TP + FN) \tag{13}$$

$$F1 = (2 \times \text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall}). \tag{14}$$

All of these metrics taken together offer a thorough grasp of how well the suggested system can identify and stop threats in Internet of Things networks. By analysing these metrics, we can assess the proposed system system's effectiveness by comparing with existing works such as VGG16, VGG19, and ResNet50.

Figure 2 shows a comparison of the precision percentages attained by several models, such as VGG16, VGG19, ResNet50, and our suggested approach, for varied IoT network node counts. All models show an improvement in precision as the

number of nodes grows up to 100. But with precision percentages ranging from 80 to 97%, our suggested approach consistently beats the other models for all node counts. Specifically, our suggested technique achieves the maximum precision of 97% at 100 nodes, significantly outperforming VGG16, VGG19, and ResNet50 by 15, 11, and 9%, respectively. This comparison research demonstrates our suggested method's improved performance in effectively identifying and mitigating attacks in Internet of Things networks, indicating that it is a viable approach to improving network security.

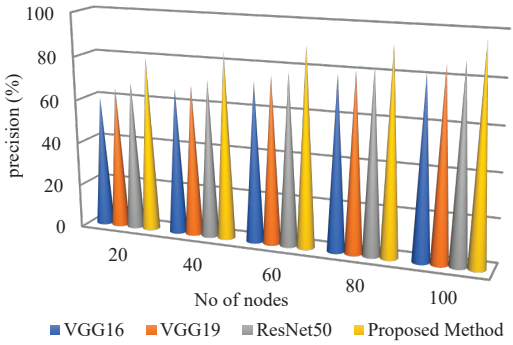

Fig. 2. Precision comparison of various models for different IoT network node counts

The performance comparison of several neural network architectures, including VGG16, VGG19, ResNet50, and the suggested technique, across various node counts is shown in Fig. 3. The models' accuracy in identifying pertinent occurrences among all relevant examples in the dataset is demonstrated by the recall percentages. The recall percentages of all models generally get better as the number of nodes rises. In every node arrangement, the suggested approach performs noticeably better than VGG16, VGG19, and ResNet50. For example, the suggested approach yields a recall of 98% with 100 nodes, compared to 80% with VGG16, 84% with VGG19, and 86% with ResNet50. This implies the superiority of the suggested approach in capturing pertinent instances, especially when the task complexity rises with higher node sizes.

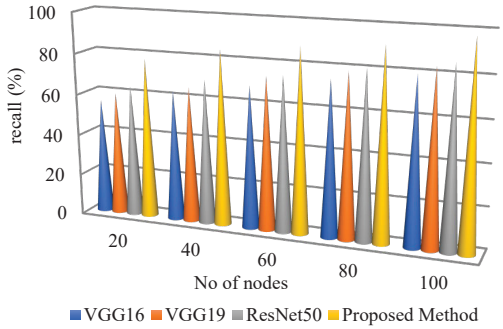

Fig. 3. Recall comparison of various neural network architectures for different node counts

Figure 4 shows the F-Measure percentages for various IoT network node counts that were produced using various models, such as VGG16, VGG19, ResNet50, and our suggested approach. Our suggested approach routinely beats the other models as the number of nodes rises from 20 to 100, obtaining F-Measure percentages between 79 and 97%. By contrast, the suggested approach performs noticeably better than VGG16, VGG19, and ResNet50, all of which show lower F-Measure percentages at all node counts. These findings highlight how well our suggested strategy works in IoT network detection and mitigation, especially as the network gets bigger.

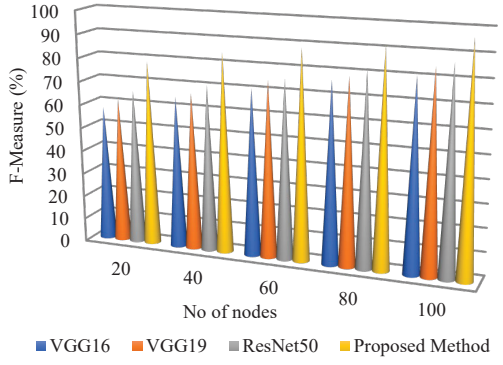

Fig. 4. F-Measure percentages comparison for different IoT network node counts using various models

The accuracy percentages of several models, such as VGG16, VGG19, ResNet50, and our suggested approach, across varying numbers of nodes in IoT networks are displayed in Fig. 5. All models exhibit increased accuracy as the number of nodes rises from 20 to 100. But our suggested approach routinely performs better than the other models, attaining the highest accuracy rates. For example, our suggested technique achieves 98% accuracy with 100 nodes, which is much higher than the accuracies of VGG16 (85%), VGG19 (89%), and ResNet50 (92%). This

demonstrates how well our method performs in identifying and thwarting threats on Internet of Things networks with different network sizes.

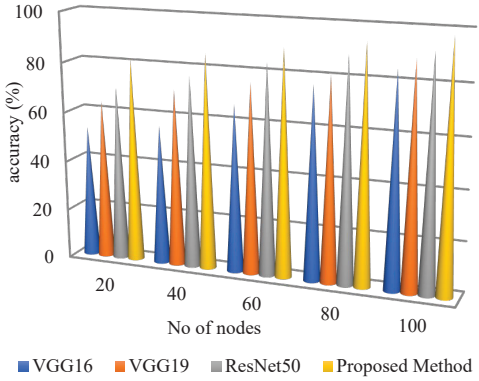

Fig. 5. Accuracy comparison of several models across different numbers of nodes

CONCLUSIONS

In conclusion, our study proposed a robust system for real-time attack detection in IoT networks using a Convolutional Neural Network (CNN) model, specifically DenseNet-169. By leveraging features extracted from pre-processed IoT network data, the model demonstrated superior performance in accurately identifying and mitigating various types of attacks. The results showcased significant improvements in accuracy (98%), precision (97%), recall (98%), and F1-score (97%) percentages compared to existing models like VGG16, VGG19, and ResNet50 across different network sizes. However, limitations such as dataset size and class imbalance could affect the model's generalisation capability. Future work should focus on addressing these limitations by collecting larger and more diverse datasets and exploring advanced techniques for handling class imbalance. Additionally, incorporating ensemble learning methods and deploying the system in real-world IoT environments would further enhance its effectiveness and applicability.

REFERENCES

1. Z. GU, S. NAZIR, C. HONG, S. KHANM: Convolution Neural Network-based Higher Accurate Intrusion Identification System for the Network Security and Communication. Secur Commun Netw, **2020**, 1 (2020).
2. L. NIE, Z. NING, X. WANG, X. HU, J. CHENG, Y. LI: Data-driven Intrusion Detection for Intelligent Internet of Vehicles: a Deep Convolutional Neural Network-based Method. IEEE Trans Netw Sci Eng, **7** (4), 2219 (2020).
3. M. KUMAR, P. MUKHERJEE, K. VERMA, S. VERMA, D. B. RAWAT: Improved Deep Convolutional Neural Network Based Malicious Node Detection and Energy-efficient Data Transmission in Wireless Sensor Networks. IEEE Trans Netw Sci Eng, **9** (5), 3272 (2021).

4. H. ASGHARZADEH, A. GHAFFARI, M. MASDARI, F. S. GHAREHCHOPOGH: Anomaly-based Intrusion Detection System in the Internet of Things Using a Convolutional Neural Network and Multi-objective Enhanced Capuchin Search Algorithm. Journal of Parallel and Distributed Computing (JPDC), **175**, 1 (2023).
5. M. V., de ASSIS, L. F. CARVALHO, J. J. RODRIGUES, J. LLORET, M. L. PROENÇA Jr.: Near Real-time Security System Applied to SDN Environments in IoT Networks Using Convolutional Neural Network. Comput Electr Eng, **86**, 106738 (2020).
6. A. M. K. KHALEEL: Internet of Thing (IoT) Cyber Security with Machine Learning. Master's Thesis, Altinbaş University**,** 2022.
7. D. SHINDE, S. NAGARKAR: Cyber Security Threats Detection and Protection Using Machine Learning Technique in IoT. 2023.
8. E. LARSEN, K. MacVITTIE, J. LILLY: A Survey of Machine Learning Algorithms for Detecting Malware in IoT Firmware. arXiv preprint arXiv:2111.02388 (2021).
9. M. WAQAS, K. KUMAR, A. A. LAGHARI, U. SAEED, M. M. RIND et al.: Botnet Attack Detection in Internet of Things Devices over Cloud Environment via Machine Learning. Concurr Comput Pract Exp, **34** (4), e6662 (2022).
10. A. ABDEL-MONEM, M. ABOUHAWWASH: A machine Learning Solution for Securing the Internet of Things Infrastructures. Sustainable Machine Intelligence Journal (SMIJ), **1**, 4 (2022).
11. M. HASAN, M. M. ISLAM, M. I. I. ZARIF, M. M. A. HASHEM: Attack and Anomaly Detection in IoT Sensors in IoT Sites Using Machine Learning Approaches. IoT, **7**, 100059 (2019).
12. K. NAIR, A. DESHPANDE, R. GUNTUKA, A. PATIL: Analysing X-ray Images to Detect Lung Diseases Using DenseNet-169 Technique. Available at SSRN 4111864. 2022.
13. G. D. L. T. PARRA, P. RAD, K. K. R. CHOO, N. BEEBE: Detecting Internet of Things Attacks Using Distributed Deep Learning. J NETW COMPUT APPL J Netw Comput Appl, **163**, 102662 (2020).
14. R. TOLOSANA, R. VERA-RODRIGUEZ, J. FIERREZ, A. MORALES, J. ORTEGA-GARCIA: Deepfakes and Beyond: a Survey of Face Manipulation and Fake Detection. Inform Fusion, **64**, 131 (2020).
15. D. GONG, Y. J. KUMAR, O. S. GOH, Z. YE, W. CHI: DeepfakeNet, an Efficient Deepfake Detection Method. International Journal of Advanced Computer Science and Applications (IJACSA), **12** (6), 201 (2021).

Received 2 April 2024
Accepted 5 July 2024