Computers, Materials & Continua
DOI: 10.32604/cmc.2022.030831
Article

Tech Science Press

Augmenting IoT Intrusion Detection System Performance Using Deep Neural Network

Nasir Sayed[1], Muhammad Shoaib[2,*], Waqas Ahmed[3], Sultan Noman Qasem[4], Abdullah M. Albarrak[4] and Faisal Saeed[5]

[1]Department of Computer Science, Islamia College Peshawar, Peshawar, 25120, Pakistan
[2]Department of Computer Science, CECOS University of IT and Emerging Sciences, Peshawar, 25000, Pakistan
[3]Department of Electrical Engineering, HITEC University Taxila, Taxila, 47080, Pakistan
[4]Computer Science Department, College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, 11432, Saudi Arabia
[5]DAAI Research Group, Department of Computing and Data Science, School of Computing and Digital Technology, Birmingham City University, Birmingham, B4 7XG, UK
*Corresponding Author: Muhammad Shoaib. Email: shoaib1646@gmail.com
Received: 02 April 2022; Accepted: 15 June 2022

Abstract: Due to their low power consumption and computing power, Internet of Things (IoT) devices are difficult to secure, and the rapid growth of IoT devices in the home increases the risk of cyber-attacks. One method of preventing cyberattacks is to employ an intrusion detection system (IDS), which detects incoming attacks and notifies the user, allowing for the implementation of appropriate countermeasures. Attempts have been made in the past to detect new attacks using machine learning and deep learning, but these efforts have been unsuccessful. In this paper, we classify network attacks using two Convolutional Neural Networks (CNN) models i.e., MyCNN and IoTCNN to automatically detect various kind malignant and benign intrusion in IoT network. The purpose of this research is to evaluate the use of deep learning in IoT IDS. The neural network was trained in this experiment using the NF-UNSW-NB15-v2 dataset, which contains nine distinct types of attacks. The data from the network stream was converted to Red Green and Blue (RGB) images, which were then used to train the neural network. To establish baseline models, we proposed two models with the name of MyCNN and IoTCNN. When compared the proposed MyCNN convolutional neural network for network attack classification, the IoTCNN was outperformed by the MyCNN model. Additionally, it demonstrates that both networks achieve higher accuracy in the majority of categories but the IoTCNN achieved lower than the proposed MyCNN model for network attack detection. We discovered that the MyCNN is generally more suitable to be deployed for intrusion detection in IoT devices.

Keywords: Internet of things; intrusion detection system; deep learning; convolutional neural network; supervised learning

1 Introduction

As the Internet of Things (IoT) has grown in popularity in our daily lives [1], it is a research and industrial trend in the field of Information and Communication Technologies (ICT) that has grown accustomed to being a part of technological advancements in our daily lives. The IoT is a new communication paradigm that refers to devices having the sensors and the actuators that can be used as objects or they are the "things," that sense their surroundings, communicates with each other, and exchange data though the Internet [2]. The IoT requires the platform's development upon which all the products, applications & services can be connected for capturing, communicating, storing, accessing, and sharing/transferring data from real world [3,4]. Around 50.0 billion IoT devices are currently connected to Internet, and the number is still expected for growing exponentially in the coming years [5,6]. Numerous applications can benefit from the massive amounts of data generated by such a large number of connected devices. Numerous IoT application scenarios exist, including those involving food and agriculture (including smart farming), demography, assisted living, e-health, and augmented learning, to name a few. By 2025, for example, there will be 15.3 billion IoT devices in use for smart agriculture [7,8]. Numerous industrial sectors and this is true across all industries, require a large no. of sensors & the actuators for real-time & environmental monitoring so that it can provide the insights that are actionable and also can make timely decisions [9].

Despite this, there are numerous impediments to the IoT widespread adoption in research and industry. Security and trust, reliability, scalability, and portability are just a few of the issues that can cause problems. The IoT introduces a wide variety of potential security threats due to their connection to the global Internet via immature and insecure communication protocols and applications [3,4]. A cyberattack could use unauthorized access to IoT-based applications to introduce anomalies and cause them to make incorrect control decisions, potentially wreaking havoc on people's lives and property, as well as the economy [7,10]. The sophistication of threats to the IoT ecosystem is increasing, and cyber-attacks pose a significant threat to the ecosystem. To compound matters, IoT devices rely on various platforms and network connectivity protocols (such as Ethernet, Wi-Fi, ZigBee, and wired-based technologies) to achieve connectivity. Multiple standards and protocols must be coordinated in order to reduce security threats. The heterogeneous & distributed nature of IoT applications adds to the complexity of IoT networks & thus increasing the likelihood of security breaches in the IoT environment as a result of the various technologies utilized by IoT industry. Numerous security issues and cyber-attacks on IoT networks result from these disadvantages. Accurate anomaly detection IDS models are therefore critical for the success of IoT applications. Numerous IDS have been proposed to safeguard IoT devices from cyber criminals [11–13].

Active security measures and passive security measures are the two types of security solutions that can be implemented. Taking proactive measures to defend against external threats can effectively protect the IoT. Although active security measures are in place, there is a high risk of intrusion because the IoT is connected to the global Internet. IDS are used as a second line of defense and can detect and prevent a wide range of cyberattacks and other threats. A large number of IDS solutions have been proposed [14–16], and IDS solutions have received a great deal of attention from researchers and industry in the context of the IoT. In terms of the methods used to detect them, IDS solutions can be divided into three categories: signature IDS models, anomaly IDS models, and hybrid IDS models, among others. As a general rule, signature-based approaches are more effective against known attacks, whereas anomaly-based approaches are more effective against unidentified attacks. It is inefficient and unproductive to employ traditional signature-based techniques since they demand constant human intervention and expertise for extract attacking patterns and patterns to updating the IDS models [17,18]. This has occurred as a result of the heterogeneity, dynamic nature, and complexity of IoT networks. Using anomaly-based intrusion detection systems in the IoT has several advantages, including detecting zero-day attacks and the need for less manual intervention [18]. In hybrid approaches, signature-based approaches are combined with anomaly-based approaches to provide a complete picture. The use of signature-based intrusion detection systems in the IoT networks, on the other hand, is restricted due to the impracticality of relying on predefined attack patterns (signature-based) for intrusion detection in the networks of IoT [16–18].

Anomaly intrusion detection systems, which are critical components of intrusion detection in IoT environments, are essential in combating this threat. The vast majority of existing IDSs develop detection models using traditional machine learning techniques [19], described in more detail below. Machine learning techniques are frequently used in developing IDS models, and they are particularly effective. When it comes to extracting representative features from the big and unstructured data generated by IoT devices, however, the speed and volume of IoT devices necessitate extensive research efforts. As a result of the large amount of unstructured data generated by IoT devices, traditional machine learning techniques, which require elaborate feature engineering, will necessitate extensive research efforts to extract representative features from the data. Therefore, traditional machine learning-based solutions continue to encounter several challenges when putting them into practice. During the past several years, deep learning (DL) techniques have become increasingly popular in IDSs, particularly in the military. In order to extract relevant information from rapidly changing and real-time data streams, DL accelerates the analysis of fast and real-time data streams to predict the future of the IoT.

In general, digital learning is seen as more trustworthy and accurate than conventional learning since it is easier to extract information and so delivers superior accuracy [20]. As a result of its ease of use and higher degree of accuracy in extracting data, digital learning is used. Therefore, the application of DL techniques in areas such as anomaly and malware detection has been the focus of some research, with the results still inconclusive. DL techniques have been applied to problems that fall under 2 different perspectives, namely, technical & regulatory, in anomaly & malware detection. Most IDS systems have been built using existing computer networks, wireless sensor networks, and self-organizing mobile networks, among other sources. However, while the IDS proposed for these networks is suitable for IoT applications, the unique characteristics of IoT-based networks, such as connectivity to the global Internet and the use of lightweight resources, render it ineffective for IoT applications [11,12]. Only a few studies have been conducted in the IoT domain specifically focused on DL techniques [21]. Deep learning-based IDSs haven't been specifically studied, as far as the authors know, in IoT security research. Consequently, the aim of this study is to fill up this gap and investigate how decentralized techniques may be used most effectively and efficiently in IoT environments. This review, which employs DL techniques to discover robust anomaly-based intrusion detection systems, provides an in-depth, focused, and high-quality analysis that will guide future research in the field of IDS.

We proposed two novel Deep Convolutional Neural Network (DCNN) methods for the detection of various malicious and benign intrusion attacks in a IoT networks. We improved the performance of an automatic intrusion detection and reduced the computational power which may be helpful for low power IoT devices in the network. We identified the subcategory of cyberattacks in the IoT networks. We performed a comparison of two custom created DCNN model in term of accuracy and efficiency the proposed scheme with other DL and traditional Machine Learning (ML) techniques. Unlike many previous works we use a modern dataset that is using new standard feature set for network intrusion detection system datasets. In this manuscript, we use a small convolutional neural network that can fit on IoT devices whereas prior work using CNNs have not been specifically aimed at IoT.

2 Literature Review

The information industry's software and hardware systems are protected from external threats by security, an essential and integral component of the sector. As the amount of new data generated on the Internet continues to grow, the importance of data security to a business's success grows.

2.1 Internet of Things Security

The Internet of Things is not a centralized system core architecture (IoT). The IoT has created a new ecosystem of connected devices and smart cities, breaking away from traditional centralized system core architectures [22]. It is now an essential part of the smart city infrastructure. By using embedded device networks or other wireless methods, anyone with an internet connection can access massive amounts of sensitive and private data in the future [23]. The IoT has many advantages for users, service providers, and businesses, but it also has many drawbacks.

An embedded system communicates via a protocol that varies by device and application, unlike a traditional network system. There is currently no centralized and uniform framework for developing security measures. This increases security risks associated with IoT networks as data volume increases [24]. Most network attacks and anomalous behavior involve malicious code, trojan horses, hackers, and viruses. These are the most common and destructive malicious objects. Device theft, device manipulation, identity theft, and eavesdropping [25] are all serious threats to IoT networks. When the IoT is compromised, it affects both people's lives and data integrity. For example, hackers can hack a person's fitness tracker or IoT-based perimeter defense system, gaining unauthorized access to sensitive workplace areas like Information Technology (IT) infrastructure. Also, IoT flaws make it easier for hackers to access sensitive data and systems. The device can authenticate and encrypt data before transmission and exchange to prevent unauthorized access. Encryption prevents hackers from gaining access to data even if they steal it. A device that cannot be manipulated avoids the autonomous control trap [26].

Signed coding certificates serve as digital signatures to ensure that only authenticated code is executed and that only the editor can break or change the code. Due to encryption, IoT devices often run on different platforms [27]. By 2020, over a billion devices and servers will be online, giving hackers an infinite number of new attack vectors to profit from. Without centralized control, the only way to protect these devices is to protect them at the factory. Because cloud servers are the primary traffic conduit for the IoT, they must act as a secondary perimeter defense. Existing cloud security regulations can be tailored to the IoT environment [28], allowing for the most efficient implementation. The use of Deep Learning (DL) methods such as Deep Reinforcement Learning (DBN) has accelerated the development of IDSs for IoT servers [29].

Network traffic identification is critical to network security because it alerts when an intrusion is detected. Traditional detection systems become ineffective as traffic packets and users grow. Port detection is another long-used technique. The basic Hyper Text Transfer Protocol (HTTP) does not perform well due to the system's limited protocol set compared to other systems worldwide. Other system applications use the signature mode approach to deliver specific payloads. [30] Remember that this strategy can be used in a wide range of situations. Many machine learning algorithms have been proposed for traffic recognition, each with its own set of benefits and drawbacks. Traditional traffic recognition and classification algorithms like plain Bayes [31], random forests [32], and decision trees [33] have long been used. In their work, Yang et al. [34] used Restricted Boltzmann Machine (RBM) and Support Vector Machine (SVM) to detect and identify network traffic. However, machine learning has enabled the development of more sophisticated methods for analyzing security issues.

The researchers say there are numerous ways to apply DL to cybersecurity. A recent study found that hybrid methodologies were used to create IDSs. Various DL methods are used in the overall process, which is divided into three categories. [34] The method has been used to detect numerous malicious anomalies in network traffic. The method uses a variety of statistical techniques to process network data computations. This is done by observing how the system's neutral points interact. To detect network security anomalies, a variety of network security variables must be evaluated.

2.2 Factors in Intrusion Detection

When an IDS attempts to detect a system intrusion, it must first distinguish between normal data and system anomalies. Thus, the IDS should detect malicious data characteristics. The classification system must also accurately distinguish between the two information sets for normal and abnormal data. The system uses a network-specific code to calculate distance between nodes automatically [35]. This technique's use is predicated on the following fundamental assumption: A distance between two nodes' consistency is used to estimate data integrity and normality [36]. As a result, a large distance between nodes indicates anomalous data, which is a red flag. The Manhattan distance can be calculated in two ways: by using the cumulative network size distance or by using the Euclidean distance, which focuses on calculating the vector size. Manhattan Distance is used to calculate the distance between two points in a network. Another issue is sample integrity, which is used to detect anomalies in samples (also known as ordinary poisoning). The output of the feature extraction functions is critical in unsupervised DL because it determines the overall learning outcome. As a result, the method should protect normal data while allowing for continuous self-defense. Manipulation of a network includes adding traffic and performing information operations [37].

2.3 Traditional Detection Methods

Machine learning, on the other hand, can be used to detect malicious anomalies in a network, which is particularly useful for catching hackers. [38] Automatic dimensionality reduction (ADR) is a video streaming technique that uses both the encoder and decoder to reduce dimensionality. There are three more layers: an input layer, an output layer, and a hidden layer. Autoencoders also use pre-training, deployment, and fine-tuning. All of this is done by professionals in a controlled setting. Deep belief networks (DBNs) [39] are also significant because they combine unsupervised RBM and supervised backpropagation network layers, and are considered traditional DL methods. Unsupervised RBM and supervised backpropagation network layers are combined in deep belief networks (DBNs) [39]. Unsupervised RBM processing and supervised RBM processing are two DBN approaches. These two techniques were combined to create a hybrid anomaly detection system. This method's dimensionality reduction is achieved by separating vectors using autoencoder algorithms [40]. As a result, DL systems classify data collected by DBN systems. In conclusion, hybrid systems have less time-related complexity and more detection accuracy. Studies show that protocol identification threatens traditional detection methods, even though over one-eighth of all network traffic is unidentifiable. [41] A system that uses probabilistic operations on unknown flows, such as DL, is more accurate.

2.4 Sequential Detection Method

The language model represents the probability distribution of the word sequence being evaluated in Natural Language Processing (NLP) tasks. The RNN-based language model is widely used in NLP, but it can also be used with other sequential data types. Sequential data, like system routines, network load, and programme code, dominates the cyberspace security domain. As in sequence modelling, the vanishing and exploding gradient of basic RNN models is unstable. Several relevant papers and experiments have used the Long Short-Term Memory (LSTM) model [42]. Because it depicts the interaction between programme and system kernel, system routine sequence data is the most useful and accurate data to have. Real-time traces of the system routine are easy to obtain for data acquisition [43]. In addition, the system routines sequence corresponds to the system's language. The system routine and its sequence are compared to natural language words and sentences. Using this data, we can determine if the system is out of sync. To forecast anomalies, the sequential model's backend uses an ensemble of thresholding classifiers to model the system's normal sequence in language. The language model can estimate a system's sequence's probability distribution. The data is fed into the model using a one-hot encoding form [44]. During the training phase, the Backpropagation Through Time (BPTT) algorithm [45] provides standard samples. The back-end classifiers are Neural Network (NN) and k-means models [46]. A web shell takes more steps to detect than a backdoor. The first step is to convert PHP (Hypertext Preprocessor) files to opcodes using Vulcan Logic Disassembler (VLD). The Bag of Word (BOW) model can then decode the opcode. The final step is to feed the sequence into the language model.

3 Methodologies

This section discusses how the intrusion detection dataset i.e., NF-UNSW-NB15-v2 presented in the paper and how the same was used to develop DL models to detect/identify and classify cyberattacks. Details of our categorization studies and neural network model utilized for testing are described in great depth. Additionally, we discuss classification experiments more broadly.

3.1 UNSW-NB15

The Cyber Range Lab at the Australian Centre for Cyber Security (ACCS) published the dataset in 2015. According to the researchers at the Australian Centre for Cyber Security, it is one of the most extensively utilized NIDS datasets on the market (ACCS). Additional 12 features were created by matching the Argus and Bro-IDS datasets to each other and extracting 35 features [47]. Network traffic generated by the IXIA PerfectStorm tools and captured as pcap files are also included. By comparing the NF-UNSW-NB15-v2 dataset to the UNSWNB15 dataset, we were able to create a new dataset called the NF-UNSW-NB15-v2, which was developed utilising 43 NetFlow characteristics retrieved from the nProbe pcap file as criteria. The NF-UNSW-NB15-v2 dataset is a newly created dataset that is based on the UNSWNB15 dataset but includes extra information. The IoT data is preprocessed by converting it to RGB images using the short-term fast Fourier transform method, the resolution of the constructed RGB image is 256×256 24 bit image with are saved in .png file format. As a result, this dataset [48] was chosen, and the results demonstrated that using this new dataset resulted in significant improvements in multi-class classification and reduced prediction times.

3.2 Data Processing

After data is collected, it must be pre-processed before being used to train our model, which is the next step after data collection. Cleaning the data, transforming it, splitting it into training and testing sets, and creating image representations are all steps in a five-step process. A Comma Separated Values (CSV) file containing 43 NetFlow functions, each with an attack category and a malicious or not label. After cleanup, we have a dataset with 37 features Tabs. 1 and 9 classes Fig. 1. For speed, we used only 40% of the dataset. Hierarchical split ensured equal distribution of attacks. After the image conversion, we needed to split the dataset into training (60%) testing (20%) and validation (20%) sets.

Each attack in each dataset is represented hierarchically. A total of 560927 training, validation, and test samples were used. In general, large datasets are trained with 3-way holdout because high variance is less of an issue than small datasets [49]. Because our dataset is not image-based, we used the same method as Nguyen et al. [50]. Fig. 2 shows the three steps. Our data are normalised using min-max scaling Eq. (3.1). Padding ensures the array is the right size. An 8-bit integer is created by multiplying 255 by a colour map. Color maps help DoS (Denial-of-Service) attacks and malware improve CNN images [51,52]. The Eq. (3.1) is used to normalized the IoT data which is used for model training and validation, the variable x represent a features of an attribute while the max(x) is the maximum value in that attribute and min(x) is the minimum value in the selected attributed used for normalizing an entire attributes values in a dataset.

$$\acute{x} = \frac{x - Min\,(x)}{Max\,(x) - Min\,(x)} \tag{3.1}$$

Table 1: A few of the 37 features from the NetFlow used to train classifiers

Feature	Description
PROTOCOLP	Identifier byte for IP protocol
L7-PROTOCOL	Layer 7 protocol
INPUT-BYTES	No. of bytes (Incoming)
OUTPUT-BYTES	No. of bytes (Outgoing)
INPUT-PACKETS	Income No. of packets
OUTPUT PACKETS	Outgoing No. of packets
No-PKTS-1-128-BYTES	Packets whole IP size <= 128
No-PKTS-128-256-BYTES	Packets whole IP size > 128 and <= 256
No-PKTS-256-512-BYTES	Packets whole IP size > 256 and <= 512

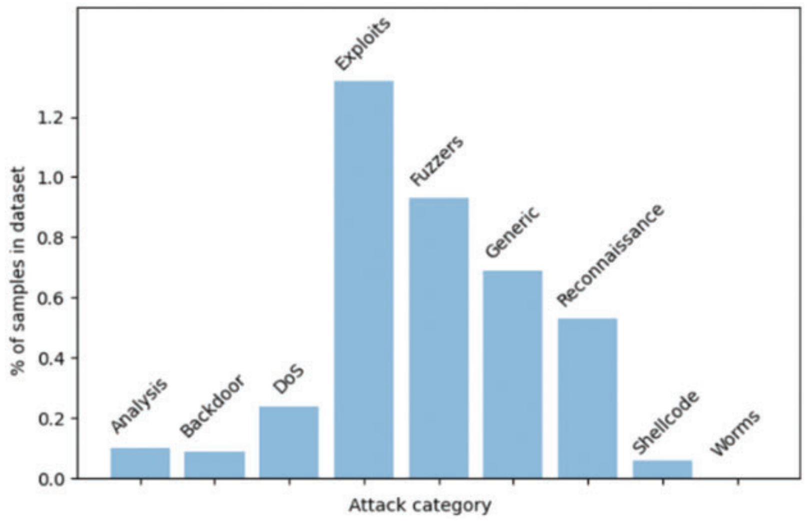


Figure 1: Frequency of instances by class distribution using the histogram method

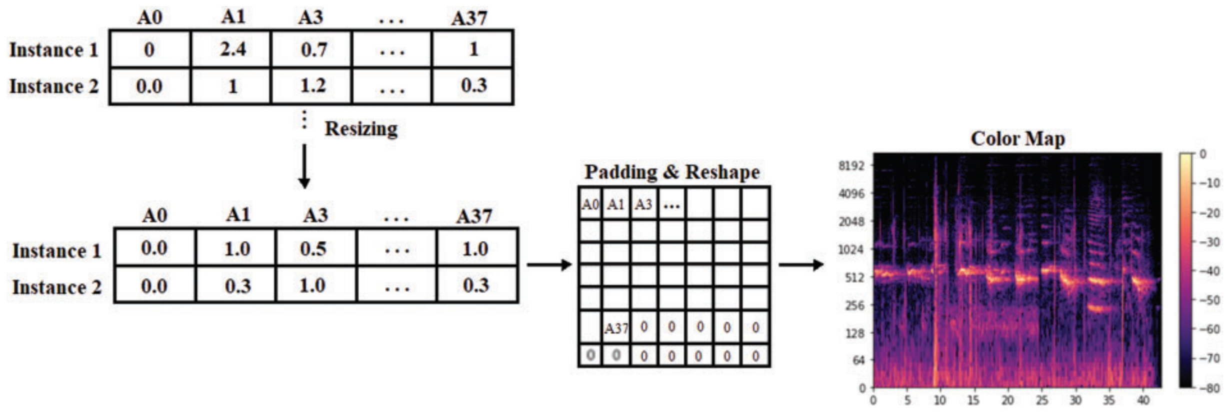


Figure 2: The process of creatin g a 2D RGB image

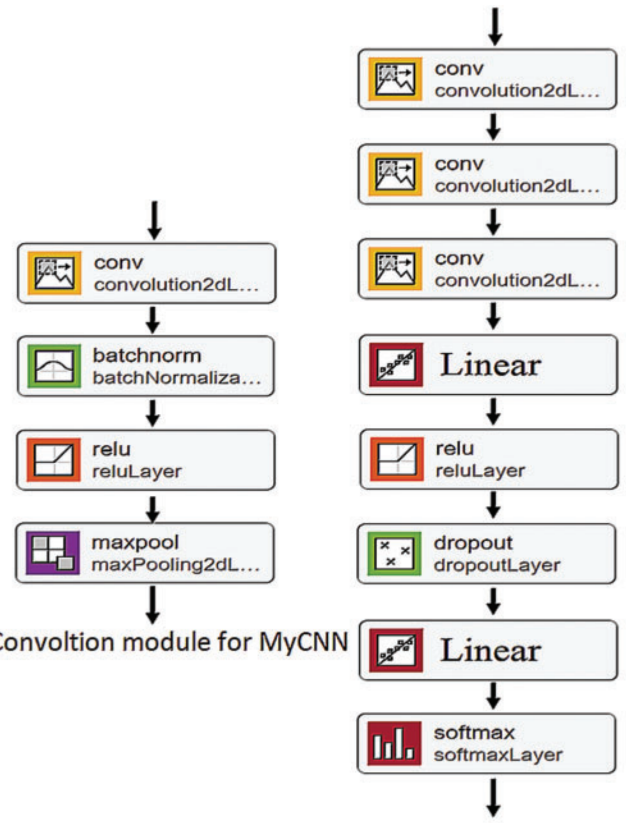
3.3 Models

3.3.1 MyCNN

Fig. 3a shows three convolutional modules, followed by two fully connected layers and a softmax activation layer, which modifies the basic CNN structure slightly from what is shown. IoT and Android malware [53,54] can be categorized using similar smaller models, as well as other types of malware [55]. Given that this network will be used for comparison purposes, it bears little resemblance to the basic CNN structure that will serve as the investigation's starting point. There are three layers total: a 3×3 convolution layer, a batch normalization layer, and a max-pooling layer ($2x2$) for downsampling the input data. On the right side of Fig. 3b, the complete structure of MyCNN can be seen, where the first two convolutional modules tends to increase the width from three to sixteen and finally to thirty-two by increasing the number of convolutional modules between them. As can be seen from the Fig. 3b, there is no representation of the spreading layer between the first fully connected layer and the third convolutional module, which is responsible for converting the matrix to a single array. Between the first fully connected layer and the ReLU and between the first fully connected layer and the subsequent dropout layer, there is a probability of 0.5. The data is then transmitted to the final, fully-connected layer, which extracts the features needed for classification. The final fully-connected layer in MyCNN is the maximum pooling layer.

3.3.2 IoTCNN

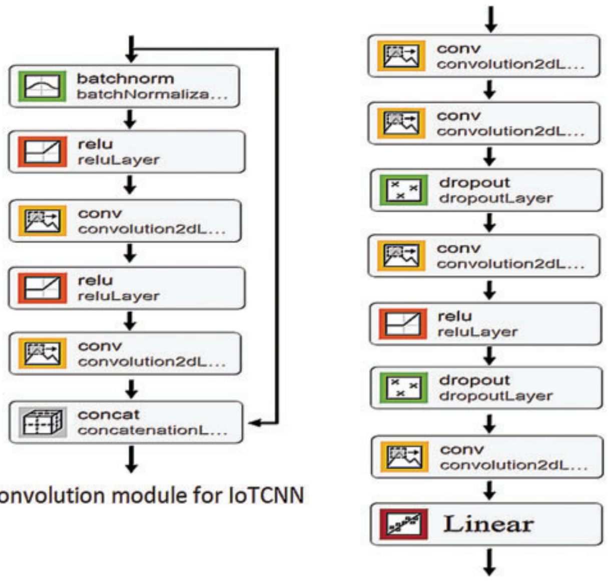
While the standard 3×3 convolution is a good starting point for most applications, T. Lawrence and L. Zhang developed our second model, IoTCNN, for resource-constrained environments such as IoT devices. Convolutional modules (blocks) and groups (multiple blocks) make up the network structure instead of the standard 3×3 convolution [56]. Fig. 4 illustrates a hierarchical organization of blocks, with each block incorporating batch normalization, an activator for the ReLU, and a jump connection to the next block in the hierarchy. The network structure is more fully illustrated in Fig. 4b; to control the network's width the first 3×3 convolution along with the first block of each group of n convolution blocks are used, and the width factor k is calculated using the method proposed by S Zagoruyko and N Komodakis [57]. Due to the model's proclivity for overfitting most classes, we added the dropout with probability of 0.2 between each set of blocks. A final fully connected layer at the top of the hierarchy performs the final classification. The parameters $n = 3$ and $k = 0.2$ were chosen for training the IoTCNN network.



(a) Convoltion module for MyCNN

(b) Structure of MyCNN Groups

Figure 3: MyCNN architecture



(a) Convolution module for IoTCNN

(b) Structure of the IoTCNN Groups

Figure 4: IoTCNN architecture

3.4 Training

For training classifiers on unbalanced datasets, we implemented a number of different techniques, all of which are described in this section. The testing hardware and methods used to select hyperparameters are also described here. It also describes how we selected the hyperparameters that were used for testing.

3.4.1 Hardware

We trained four neural networks concurrently without experiencing any lag when using hardware rented from vast.ai, a service that delivers docker instances over peer-to-peer hardware for rental aggregate. We trained four neural networks concurrently using rented hardware with no lag.

- HP Z-440 Intel® Xeon® Processor E5-2697 v3 CPU @2.90Ghz
- Nvidia RTX 2070 Super 8 GB GPU
- 47 GB of DDR4 RAM

3.4.2 Hyperparameters

Our research focused on two hyperparameters: learning rate and batch size. We deemed these two parameters crucial. In the end, the models with the least validation loss had the best performance. We used this data to calculate our hyperparameters. The learning rate was set to 0.01, 0.001, 0.0001, and 0.00001 to allow the model to converge as quickly as possible without returning a worse solution. Our tests show that when the learning rate is greater than 0.0001, the model achieve to learn and produces useful results, instead classifying all samples into the same category. Because a learning rate of 0.00001 gives significant results for 50 epochs, we used 0.00005 to get a good balance of performance and efficiency. Consider the number of samples used to train the network in each iteration when determining batch size. This determines the length of training and the model's generalization ability. According to S. Keskar and coworkers, using large data sets reduces the generalization ability of models [51,58,59]. As a result, big data models lose generalizability. Using larger batch sizes than 128 rather than 32, 64, and 128 can significantly reduce training time per epoch. As a result, we chose 128 for training because we are working with small images and each batch does not require a lot of graphics card RAM. Small images and low-memory batches are the main reasons.

3.4.3 Sampler

We sample our dataset using weighted random sampling and the official Pytorch implementation of this sampler. We assign a weight to each sample in the dataset we are sampling using this sampler. Eq. (3.2) [60] is used to calculate the weight of a class, where x denotes the class being weighed. We will use Eq. (3.2) to calculate the class weight, where x denotes the class [61]. The training weights are listed in Tab. 2, and they effectively undersample the majority of our classes (for example, benign) while oversampling a few (for example, malignant) (e.g., worms).

$$\propto = \frac{1}{Total\ x} \tag{3.2}$$

3.4.4 Loss Functions

Our loss function is the cross-entropy loss (CELoss), which is commonly utilised in CNNs and is frequently employed in conjunction with softmax activation. The cross-entropy loss, which is perhaps the most often used loss function for CNNs, is also evaluated to see how adding weights to each class affects the loss function in conjugation. The network will try to avoid misclassifying classes with higher weights by adding weights. In order to rebalance the loss, people developed a second loss function known as class balance loss. To determine the type of loss, there are three hyperparameters that control the loss function: softmax, sigmoid, focuses of the loss type, $\beta \in 0.9, 0.99, 0.999, 0.9999$ for adjusting between no re-weighting and inverse class frequency re-weighting, and $\gamma \in 0.5, 1.00, 2.00$ to adjust the focal loss by downward weighting rate [62,63]. These values i.e., $\beta = 0.999$ and $\gamma = 0.5$; a, worked well on all datasets tested, hence they were used in this study. Early tests revealed that when using weighted CELoss or CBLoss with a learning rate of 0.00005, the training and validation losses do not change during training, as a result, we chose the lower learning rate of 0.00001.

Table 2: Weighted random samplers employ class weights

Class	Weight
Analysis	0.190
Backdoor	0.211
Benign	0.807
DoS	0.101
Exploits	0.101
Fuzzers	0.303
Generic	0.104
Reconnaissance	0.603
Shellcode	0.509
Worms	0.820

3.5 *Classification Experiment*

In our studies, we use the NF-UNSW-NB15-v2 dataset to train and evaluate two distinct CNN models for multiclass classification, each with four possible configurations. We train 100 epochs for each model after picking hyperparameters, store the parameters when the validation loss lowers, and then test using the model with the best validation loss. The Adam optimizer [60], one of the most extensively used neural network training optimizer, is employed during the training process. Additionally, we used I. Loshchilov and F. Hutter's decoupled weight decay regularisation with a factor of 0.0005, which has been shown to improve generalisation performance when employing Adam [64]. Tabs. 3 and 4 detail the test parameters utilized for each model.

Table 3: MyCNN CNN training parameters

	MyCNN	MyCNN CE	MyCNN CN	MyCNN Sampler
Batch size	64	64	64	64
Initial LR	0.0001	0.0001	0.0001	0.0001
Loss function	Cross entropy loss	Weighted cross entropy loss	Class balance loss	Cross entropy LOSS

(Continued)

Table 3: Continued

	MyCNN	MyCNN CE	MyCNN CN	MyCNN Sampler
Weight decay	0.0004	0.0004	0.0004	0.0004
Sample	N/A	N/A	N/A	Weighted Random Sampler

Table 4: IoTCNN CNN training parameters

	IoTCNN	IoTCNN CE	IoTCNN CN	IoTCNN Sampler
Batch size	64	64	64	64
learning rate	0.0001	0.0001	0.0001	0.0001
Loss function	Cross entropy loss	Weighted cross entropy loss	Class balance loss	Cross entropy loss
Weight decay	0.0004	0.0004	0.0004	0.0004
Sample	N/A	N/A	N/A	Weighted Random Sampler

4 Results and Analysis

There were 100 epochs of training and validation datasets that were pre-segmented. A neural network's training and validation results are its loss on the training set and validation sets as well, on its validation accuracy, and on its average training time.

On the graph in Fig. 5, the dashed line represents the model's training loss, while the solid line depicts its validation loss over time. If the network is under or overfitted, the correlation between validation and training losses can be used to determine its performance. Underfitting occurs when the validation loss is less than or equal to the training loss, while overfitting occurs when this difference is bigger. All trained models after epoch 20, experience a quick decrease in loss as early in training in a closer examination of Fig. 5a. The only models in which the training loss decreases significantly over time are MyCNN Sampler and MyCNN CELoss, but because the validation loss does not decrease at the same rate, this could indicate overfitting the training set. In comparison to the other models, CBLoss has a lesser loss since it classifies attacks into fewer categories. As demonstrated in Tab. 5, the difference in loss between the IoTCNN and MyCNN models during training is bigger than the difference in loss between the MyCNN models.

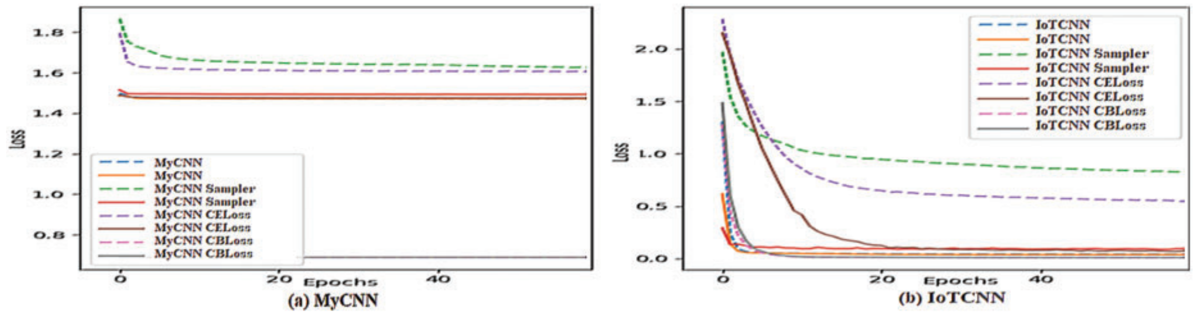
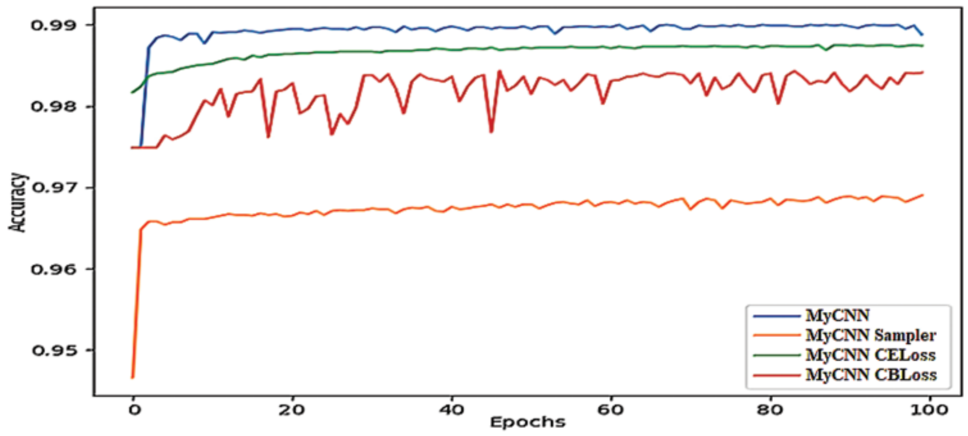


Figure 5: Classification model training and validation loss plots

Table 5: Model training with the best validation loss

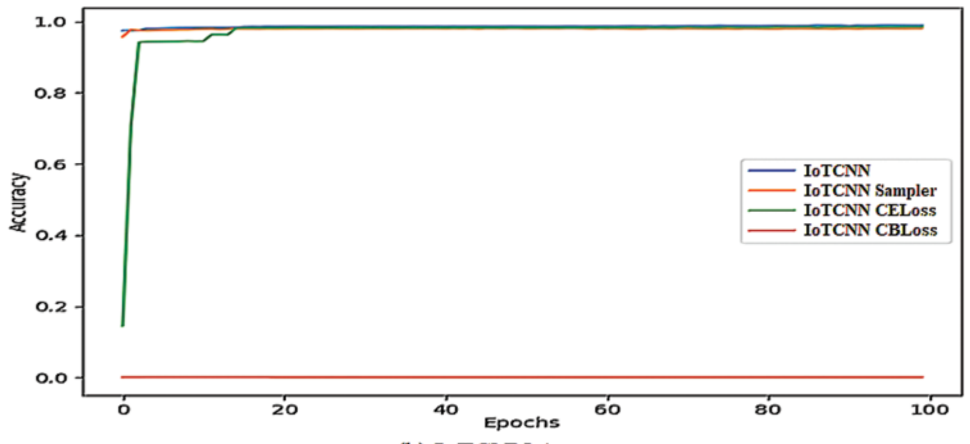
Model	Validation loss	Model	Validation loss
MyCNN	1.45	IoTCNN	0.024
MyCNN sampler	1.47	IoTCNN sampler	0.068
MyCNN CELoss	1.45	IoTCNN CELoss	0.052
MyCNN CBLoss	0.58	IoTCNN CBLoss	0.020

The validation accuracy of the model is shown in Fig. 6, and we can see that it varies more than the full range of training and validation losses. Both MyCNN and IoTCNN can achieve accuracies of close to 99 percent at their peak, which is similar to the losses we see, with most of the improvement occurring in the first 20 periods. MyCNN's resampling accuracy is lower than the other methods, whereas IoTCNN's accuracy is comparable to the other high-performing methods. When using class balance loss, the accuracy of the two networks differs, with MyCNN's validation accuracy erratic and IoTCNN's validation accuracy consistent at 0.01 percent. It is worth noting that 182,282 out of 186,976 samples are benign, or 97.49 percent, implying that classifiers that overfit to most classes will have that accuracy, implying that accuracy is not all the time a good measure for unbalanced datasets. IoTCNN takes more time to train than MyCNN because of its usage of more convolutional layers, as seen in Tab. 6. When using weighted CELoss or CBLoss for any of the four networks, we see no increase in time, but we see an increase in time when using resampling due to oversampling and undersampling.



(a) MyCNN Accuracy

Figure 6: (Continued)



(b) IoTCNN Accuracy

Figure 6: Validation accuracy plots of proposed CNN models

Table 6: Average time taken per Epoch in model training

Model	Time (s)
MyCNN (CELoss/CBLoss)	123
MyCNN Sampler	139
IoTCNN (CELoss/CBLoss)	235
IoTCNN Sampler	244

It took roughly 42 h to train all 8 models because we could only train four models at a time, resulting in a training round lasting more than 10 h each time.

4.1 Classification

The models with the lowest validation loss were selected to evaluate the classification capabilities of our models, and they were then used to classify the test samples. A confusion matrix is used to display and assess the classifying results for each model. In addition, we calculated generic metrics that measure the performance of each type of abnormality detection using the values in the matrix to determine the best model for each neural network. The proportion of correct predictions in all samples is called accuracy. The proportion of true positives to all positives is known as precision. F1 Score, the sum of precision and recall, is the proportion of correct predictions across all relevant samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + TN} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$$F1 - Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{4.4}$$

4.1.1 Cross-entropy Loss

The confusion matrix in Tab. 7 shows the MyCNN trained on an imbalanced dataset and how all samples are sorted into the three biggest classes using the conventional cross-entropy loss. The analysis has been misclassified as secure because many attack samples were classified benign. In terms of recall, the model has an 88.9 percent use rate for the vulnerabilities it categorizes.

Table 7: Confusion matrix of MyCNN proposed model based on Cross-entropy Loss

Class	Analysis	Backdoor	Benign	DoS	Exploits	Fuzzer	Generic	Recon . . .	ShellCode	Worms
Analysis	0.0%		90.0%		6.6%	3.33%				
Backdoor		0.0%	18.5%		37.0%	44.4%				
Benign			99.7%		0.05%	0.14%				
DoS			21.7%	0.0%	61.7%	16.4%				
Exploits			6.13%		88.9%	4.96%				
Fuzzer			5.38%		3.02%	91.5%				
Generic			16.9%		65.9%	17.4%	0.00%			
Recon . . .			0.76%		30.5%	68.6%		0.00%		
ShellCode			5.36%		3.50%	91.2%			0.00%	
Worms					100%					0.00%

Although like MyCNN, IoTCNN only categorizes samples into the largest category, unlike MyCNN, fewer malicious samples are wrongly categorized as benign. In comparison to MyCNN, the fuzzifier has dropped from 91.60 percent to 79.84 percent in the recall.

4.1.2 Re-sampling

Tab. 9 depicts the impact of resampling on a MyCNN using a weighted random sampler on data of successfully classified samples from all categories of interest. Worms with 100 percent recall have Q1 the best results, as this is the smallest category with the most weight. With only 37.08 percent correctly classified, DoS attacks are more difficult to classify than the second-lowest attack type by more than 20 percent.

Table 8: Confusion matrix of IoTCNN proposed model based on Cross-entropy Loss

Class	Analysis	Backdoor	Benign	DoS	Exploits	Fuzzer	Generic	Recon . . .	ShellCode	Worms
Analysis	0.0%		56.6%		40.0%	3.33%				
Backdoor		0.0%	18.5%		33.3%	48.1%				
Benign			99.6%		0.15%	0.15%				
DoS			11.9%	0.0%	76.4%	11.6%				
Exploits			2.93%		89.6%	7.38%				
Fuzzer			0.81%		19.3%	79.8%				
Generic			2.07%		80.9%	17.0%	0.00%			
Recon . . .			0.76%		13.6%	85.5%		0.00%		
ShellCode			5.36%		10.5%	89.4%			0.00%	
Worms					100%					0.00%

Table 9: Confusion matrix of MyCNN proposed model based on Weighted Random Sampler

Class	Analysis	Backdoor	Benign	DoS	Exploits	Fuzzer	Generic	Recon...	ShellCode	Worms
Analysis	93.3%		56.6%		3.33%	3.33%				
Backdoor		66.6%					7.40%	11.1%	14.8%	
Benign	0.10%	0.05%	97.4%	0.06%	2.02%	0.16%	0.07%	0.02%	0.04%	0.00%
DoS	9.36%	2.24%	0.37%	37.0%	23.5%	7.86%	11.6%	4.86%	1.12%	1.87%
Exploits	2.76%	0.90%	0.34%	8.50%	67.1%	3.84%	8.03%	6.51%	0.86%	1.07%
Fuzzer	0.24%	1.38%		1.38%	0.32%	84.7%	2.85%	1.22%	7.75%	0.08%
Generic	0.82%	0.82%		7.46%	12.8%	5.39%	69.7%	0.82%	1.65%	0.41%
Recon...		1.15%	0.19%	0.57%	0.76%	0.57%	0.38%	95.0%		1.34%
ShellCode				1.75%		31.5%		7.01%	59.6%	
Worms										100%

We can see some similarities between IoTCNN and MyCNN in Tab. 10, as this time just to classify DoS correctly both the IoTCNN and MyCNN struggle, with only a 13.86 percent recall. Compared to the resampled MyCNN, IoTCNN can classify the shellcode with a significantly higher recall of 91.23 percent. When resampling was used in both networks, the numbers of samples that are classified as benign was lower than when resampling was not used. Overall, the classifications were more evenly distributed across all ten categories, indicating that the technique successfully addressed some of the dataset's category imbalances.

Table 10: Confusion matrix of IoTCNN proposed model based on Weighted Random Sampler

Class	Analysis	Backdoor	Benign	DoS	Exploits	Fuzzer	Generic	Recon...	ShellCode	Worms
Analysis	93.3%				3.33%	3.33%				
Backdoor		66.6%					7.40%	11.1%	14.8%	
Benign	0.11%	0.09%	99.0%	0.07%	0.13%	0.14%	0.19%	0.00%	0.17%	0.00%
DoS	15.7%	2.99%	0.37%	13.8%	22.8%	8.98%	18.7%	7.11%	5.99%	3.37%
Exploits	4.05%	4.14%	0.43%	4.70%	56.9%	5.18%	12.6%	6.73%	3.36%	1.77%
Fuzzer	0.24%	0.97%		1.46%	2.20%	65.0%	4.65%	1.63%	23.5%	0.24%
Generic	1.24%	4.56%		7.88%	13.2%	21.1%	43.9%	2.07%	5.39%	0.41%
Recon...		0.19%	0.19%	0.38%	0.57%	0.38%	0.19%	95.7%	0.57%	1.73%
ShellCode						1.75%		7.01%	91.2%	
Worms								9.09%		90.9%

4.1.3 *Cost Sensitive Learning*

A. Weighted cross-entropy loss

The performance of MyCNN is affected by weighted cross-entropy loss, as shown in the Tab. 10. We can observe that the network successfully detected any additional categories, that the number of
Q2 benign attacks increases slightly, and that the number of benign samples misclassified decreases.

Table 11: Confusion matrix of MyCNN proposed model based on Weighted Cross-Entropy Loss

Class	Analysis	Backdoor	Benign	DoS	Exploits	Fuzzer	Generic	Recon . . .	ShellCode	Worms
Analysis	0.00%				96.6%	3.33%				
Backdoor		0.00%	14.8%		40.7%	44.4%				
Benign			99.5%		0.32%	0.17%				
DoS			2.62%	0.00%	80.1%	17.2%				
Exploits			0.99%		91.2%	7.03%				
Fuzzer			0.08%		7.83%	92.0%				
Generic			2.07%		79.2%	18.6%	0.00%			
Recon . . .			0.19%		31.3%	68.4%		0.00%		
ShellCode					15.7%	84.2%			0.00%	
Worms					90.9%	9.09%				0.00%

The results of IoTCNN during the use of weighted cross-entropy loss are shown in Tab. 12, which shows that the number increases if the classification class has some samples that are classified as reconnaissance, and all samples classified as reconnaissance are incorrectly classified. We see fewer false positives for benign classes & more false positives for benign ones, similar to MyCNN, and the fuzzers have increased recall and reduced vulnerability exploitation, unlike MyCNN.

Table 12: Confusion matrix of IoTCNN proposed model based on Weighted Cross-Entropy Loss

Class	Analysis	Backdoor	Benign	DoS	Exploits	Fuzzer	Generic	Recon . . .	ShellCode	Worms
Analysis	0.00%				96.6%	3.33%				
Backdoor		0.00%	14.8%		11.1%	70.3%		3.70%		
Benign			99.5%		0.30%	0.29%		0.05%		
DoS			2.62%	0.00%	77.9%	19.1%		1.12%		
Exploits			0.99%		85.2%	13.6%		0.17%		
Fuzzer			0.08%		15.5%	84.4%				
Generic					78.8%	21.1%	0.00%			
Recon . . .			0.19%		6.53%	93.2%		0.00%		
ShellCode					8.77%	91.2%			0.00%	
Worms					90.9%	9.09%				0.00%

B. Class balanced loss

As shown in Tab. 13, MyCNN only classifies samples into the 2 largest classes, with a high number of false positives for benign classes, which is worse than when no technique is used to help train the unbalanced dataset. As depicted in Tab. 7. IoTCNN's class balancing loss results are depicts in Tab. 14, all samples are labeled as backdoor attacks, resulting in the model's worst performance to date.

Table 13: Confusion matrix of MyCNN proposed model based on Class Balanced Loss

Class	Analysis	Backdoor	Benign	DoS	Exploits	Fuzzer	Generic	Recon...	ShellCode	Worms
Analysis	0.00%		93.3%		6.66%					
Backdoor		0.00%	70.3%		29.6%					
Benign			99.9%		0.07%					
DoS			56.5%	0.00%	43.4%					
Exploits			31.9%		68.0%					
Fuzzer			47.1%		52.8%	0.00%				
Generic			40.6%		59.3%		0.00%			
Recon...			24.4%		75.5%			0.00%		
ShellCode			22.8%		77.1%				0.00%	
Worms			27.2%		72.7%					0.00%

Table 14: Confusion matrix of IoTCNN proposed model based on Class Balanced Loss

Class	Analysis	Backdoor	Benign	DoS	Exploits	Fuzzer	Generic	Recon...	ShellCode	Worms
Analysis	0.00%	100%								
Backdoor		100%								
Benign		100%	0.00%							
DoS		100%		0.00%						
Exploits		100%			0.00%					
Fuzzer		100%				0.00%				
Generic		100%					0.00%			
Recon...		100%						0.00%		
ShellCode		100%							0.00%	
Worms		100%								0.00%

4.2 Results Summary

Resampling with a weighted random sampler yield results for MyCNN for each class that differs from those provided by Sarhan et al., as shown in Tab. 15. In parentheses, people are listed [29]. MyCNN had a higher recall in six of the ten classes, with analysis having the highest recall of 0.93, or 62.44 percent higher. With the exception of one class, all F1 scores were lower, MyCNN had a lower backdoor, and the best overall performance class was benign. The disproportionately low precision is because missing classifications are distributed across most classes, as shown in Tab. 9.

Table 15: MyCNN multi-class classification results

Class	Accuracy	Precision	Recall	F_1 Score
Analysis	99.85%	0.88	0.93	0.16
Backdoor	99.92%	0.11	0.67	0.19
Benign	97.51%	1.0	0.97	0.99

(Continued)

Table 15: Continued

Class	Accuracy	Precision	Recall	F_1 Score
DoS	99.72%	0.22	0.37	0.28
Exploits	97.56%	0.29	0.67	0.41
Fuzzers	99.66%	0.70	0.85	0.77
Generic	99.76%	0.30	0.70	0.42
Reconnaissance	99.87%	0.69	0.95	0.80
Shellcode	99.88%	0.15	0.60	0.23
Worms	99.97%	0.19	1.00	0.31

Tab. 16, shows the results of IoTCNN resampling using a weighted random sampler, with the same format for each category as Tab. 15. IoTCNN had greater precision and recall differences than MyCNN, and performed worse on categorical generalizers and fuzzers.

Table 16: IoTCNN multi-class classification results

Class	Accuracy	Precision	Recall	F_1 Score
Analysis	99.82%	0.07	0.93	0.14
Backdoor	99.84%	0.05	0.67	0.11
Benign	99.07%	1.0	0.99	1.0
DoS	99.73%	0.12	0.14	0.13
Exploits	99.27%	0.78	0.57	0.66
Fuzzers	99.52%	0.63	0.65	0.64
Generic	99.52%	0.12	0.44	0.19
Reconnaissance	99.87%	0.70	0.96	0.81
Shellcode	99.61%	0.06	0.91	0.12
Worms	99.96%	0.12	0.91	0.22

Analysis, backdoor, and DoS were the worst-classified categories, consistent with Sarhan et al. People's findings. We improved recall in all of the above classes by using MyCNN and resampling. Our model falls short when classifying vulnerability exploits, shellcode [29] and generalizations.

4.3 Discussion

Small CNNs can classify network attacks using resampling and the NF-UNSW-NB15-v2 dataset. On the contrary, our models have higher F1 scores, as shown in the Tabs. 15 and 16. Our models' successfully classified various kind of intrusion attacks. The proposed models are generalizable, so they can detect attacks efficiently with higher accuracy. To overcome class imbalances, we can use hybrid techniques rather than testing every possible parameter, the most critical ones: learning rates and batch sizes were chosen . The number of groups and blocks in IoTCNN should be adjusted, as should the rate of dropout for each dropout layer. The model training time for IoTCNN is less than 7 h, including tuning the model and hyperparameters. This time was set aside to ensure the model's functionality. Better parameters has shown improvement in our results. The final dataset was too large and unbalanced to be evaluated using a larger dataset. Only one or two samples in the smallest subset make learning the class impossible. Due to the dataset's imbalance, we needed a large dataset to get good representation of each class. So our model is limited, the experimental results show that the trained models' performance is superior to the reference paper's. Even when using a random sampler, class precision is still higher. Resampling and cost-sensitive learning can produce good results, and the worst-performing classes can be given extra weight. Because of the IoT device training takes too long, externally trained cloud models should be distributed to each IoT device. The IoT device can learn a few new assault samples. The IDS should be trained overnight so it can be updated quickly when new attacks are discovered. IoTCNN took longer time in model training and validation process than the MyCNN.

To classify a test set, it took an average of 51 s (0.27 ms/sa When used as an IDS, this speed can result in an infinite backlog. Maybe for low-traffic devices. The difference in inference times between the two networks was only 3 s.

It's important to note that while MyCNN (3.79 MB) is smaller than IoTCNN (3.05 MB), the models' trainable parameters differ significantly. Aside from that, neither MyCNN nor IoTCNN has any untrainable parameters. A classifier's main goal is generalization. Our models are accurate and more generalizable. A better sampler or loss function may help generalization. Using larger input images with larger width factors can also improve generalization.

Because the dataset is unbalanced, resampling and cost-sensitive learning are ineffective. We could avoid hierarchical splitting for training and validation by selecting evenly from classes. One reason for not choosing the small sample size available for the least populated worm class. It is best to analyses the features further during pre-processing to save or gain space in the final image. This should improve the convolutional layer input quality. Add noise or other enhancement techniques to the image to reduce category preference. Pads add too much data to the image due to their small size. Padding can be tested because it can affect the results. The expansion factor, group count, and block count should all be evaluated. A few more lightweight classification models would be effective. Instead of a fixed learning rate, the leaning scheduler could be used. So, the configuring of the learning rate scheduler during training should be skipped. Finally, the resampler and loss function rates were fine-tune to achieve better performance in IoT IDS.

5 Conclusion

We created two convolutional neural networks and trained them on the NF-UNSW-NB15-v2 dataset to classify cyber-attacks. This research aims to look into how CNNs designed for IoT devices perform in cyber-attacks and see if they can be used as anomaly-based intrusion detection systems. Experiments show that the model can classify cyber-attacks accurately and that it can be run as an IDS in real-time on most devices. We used resampling and cost-sensitive learning in this research to test multiple methods for improving the accuracy of training DL models on unbalanced datasets. Finding the optimal weights and the possibility of combining resampling with cost-sensitive learning using a hybrid approach are two areas that need to be investigated further. Unbalanced datasets would be addressed in this way. Different splitting methods or other oversampling methods that introduce more variation in the data could be used to address unbalanced datasets. Use a different preprocessing method that does not include padding, leading to errors. Larger images can be created from a small amount of data because convolutional neural networks use filters that pass through the image. This can result in more connections between features depending on how the image is constructed. The investigation is needed in the field of fine-tuning the hyperparameters in IoTCNN, and in other different DL models. The use of different optimizers for training can also be investigated for improving the classification accuracy of the proposed models, as the Adam optimizer is the only one used in this paper.

Acknowledgement: The authors extend their appreciation to the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University for funding this work through Research Group No. RG-21-07-04.

Funding Statement: This research is funded by Imam Mohammad Ibn Saud Islamic University, RG-21-07-04.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] L. Atzori, A. Iera and G. Morabito, "Understanding the internet of things: Definition, potentials, and societal role of a fast evolving paradigm," *Ad Hoc Networks*, vol. 56, pp. 122–140, 2017.
[2] M. F. Elrawy, A. I. Awad and H. F. A. Hamed, "Intrusion detection systems for IoT-based smart environments: A survey," *Journal of Cloud Computing*, vol. 7, no. 1, pp. 1–20, 2018.
[3] L. Da Xu, S. Member, W. He and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.
[4] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang *et al.,* "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.
[5] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi and A. Razaque, "Deep recurrent neural network for IoT intrusion detection system," *Simulation Modelling Practice and Theory*, vol. 101, no. may 2020, pp. 102031, 2020.
[6] S. J. Moore, C. D. Nugent, S. Zhang and I. Cleland, "IoT reliability: A review leading to 5 key research directions," *CCF Transactions on Pervasive Computing and Interaction*, vol. 2, no. 3, pp. 147–163, 2020.
[7] M. A. Ferrag, L. Shu, X. Yang, A. Derhab and L. Maglaras, "Security and privacy for green IoT-based agriculture: Review, blockchain solutions, and challenges," *IEEE Access: Green Internet of Things*, vol. 8, pp. 32031–32053, 2020.
[8] M. S. Farooq, S. Riaz, A. Abid, K. Abid and M. A. Naeem, "A survey on the role of IoT in agriculture for the implementation of smart farming," *IEEE Access*, vol. 7, pp. 156237–156271, 2019.
[9] J. Ruan, Y. Wang, F. T. S. Chan, X. Hu, M. Zhao *et al.,* "A life cycle framework of green IoT-based agriculture and its finance, operation, and management issues," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 90–96, 2019.
[10] F. A. Ghaleb, M. A. Maarof, A. Zainal, M. A. Rassam, F. Saeed *et al.,* "Context-aware data-centric misbehaviour detection scheme for vehicular ad hoc networks using sequential analysis of the temporal and spatial correlation of the consistency between the cooperative awareness messages," *Vehicular Communications*, vol. 20, no. December 2019, pp. 100186, 2019.
[11] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.
[12] G. Thamilarasu and S. Chawla, "Towards deep-learning-driven intrusion detection for the internet of things," *Sensors (Switzerland)*, vol. 19, no. 9, 2019.
Q3
[13] Y. Yang, K. Zheng, C. Wu and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," *Sensors (Switzerland)*, vol. 19, no. 11, 2019.
Q4

[14] W. C. Shi and H. M. Sun, "DeepBot: A time-based botnet detection with deep learning," *Soft Computing*, vol. 24, no. 21, pp. 16605–16616, 2020.

[15] M. Munir, S. A. Siddiqui, A. Dengel and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019.

[16] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transaction on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[17] S. Hajiheidari, K. Wakil, M. Badri and N. J. Navimipour, "Intrusion detection systems in the internet of things: A comprehensive investigation," *Computer Networks*, vol. 160, pp. 165–191, 2019.

[18] M. Fahim and A. Sillitti, "Anomaly detection, analysis and prediction techniques in IoT environment: A systematic literature review," *IEEE Access*, vol. 7, pp. 81664–81681, 2019.

[19] K. A. P. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz and V. H. C. de Albuquerque, "Internet of things: A survey on machine learning-based intrusion detection approaches," *Compuer Networks*, vol. 151, pp. 147–157, 2019.

[20] H. Albasheer, M. M. Siraj, A. Mubarakali, O. E. Tayfour, S. Salih *et al.,* "Cyber-attack prediction based on network intrusion detection systems for alert correlation techniques: A survey," *Sensors*, vol. 22, no. 4, pp. 1–15, 2022.

[21] B. Sharma, L. Sharma and C. Lal, "Anomaly detection techniques using deep learning in IoT: A survey," in *Int. Conf. on Computational Intelligence and Knowledge Economy ICCIKE 2019*, Dubai, UAE, pp. 146–149, 2019.

[22] W. Wang, F. Xia, H. Nie, Z. Chen, Z. Gong *et al.,* "Vehicle trajectory clustering based on dynamic representation learning of internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3567–3576, 2021.

[23] W. Wang, J. Chen, J. Wang, J. Chen and Z. Gong, "Geography-aware inductive matrix completion for personalized point-of-interest recommendation in smart cities," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4361–4370, 2020.

[24] W. Wang, J. Chen, J. Wang, J. Chen, J. Liu *et al.,* "Trust-enhanced collaborative filtering for personalized point of interests recommendation," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6124–6132, 2020.

[25] E. Schiller, A. Aidoo, J. Fuhrer, J. Stahl, M. Ziörjen *et al.,* "Landscape of IoT security," *Computer Science Review*, vol. 44, no. May, pp. 100467, 2022.

[26] M. Conti, A. Dehghantanha, K. Franke and S. Watson, "Internet of things security and forensics: Challenges and opportunities," *Future Generation Computer Systems*, vol. 78, pp. 544–546, 2018.

[27] D. E. Kouicem, A. Bouabdallah and H. Lakhlef, "Internet of things security: A top-down survey," *Computer Networks*, vol. 141, pp. 199–221, 2018.

[28] K. Gupta and S. Shukla, "Internet of things: Security challenges for next generation networks," in *2016 1st Int. Conf. on Innovation and Challenges in Cyber Security (ICICCS 2016)*, Greater Noida, India, no. Iciccs, pp. 315–318, 2016.

[29] I. S. Gracia, O. D. Suarez, G. B. Garcia and T. K. Kim, "Fast fight detection," *PLoS One*, vol. 10, no. 4, pp. 1–19, 2015.

[30] H. Peng, C. Liu, D. Zhao and J. Han, "Reliability analysis of CPS systems under different edge repairing strategies," *Physica A: Statistical Mechanics and its Applications*, vol. 532, pp. 121865, 2019.

[31] C. Kruegel, D. Mutz, W. Robertson and F. Valeur, "Bayesian event classification for intrusion detection," in *Proc.-Annul Computer Security Applications Conf. ACSAC*, Las Vegas, NV, USA, vol. 2003-January, pp. 14–23, 2003.

[32] C. Sinclair, L. Pierce and S. Matzner, "An application of machine learning to network intrusion detection," in *Proc.-Annul Computer Security Applications Conf. ACSAC*, Phoenix, Arizona, USA, vol. Part F133431, no. 0293, pp. 371–377, 1999.

[33] J. Zhang and M. Zulkernine, "A hybrid network intrusion detection technique using random forests," in *Proc. the First Int. Conf. on Availability, Reliability and Security*, Vienna, Austria, vol. 2006, pp. 262–269, 2006.

[34] J. Yang, J. Deng, S. Li and Y. Hao, “Improved traffic detection with support vector machine based on restricted boltzmann machine,” *Soft Computing*, vol. 21, no. 11, pp. 3101–3112, 2017.
[35] H. Peng, Z. Kan, D. Zhao and J. Han, “Security assessment for interdependent heterogeneous cyber physical systems,” *Mobile Networks and Application*, vol. 26, no. 4, pp. 1532–1542, 2021.
[36] L. Greche, M. Jazouli, N. Es-Sbai, A. Majda and A. Zarghili, “Comparison between Euclidean and manhattan distance measure for facial expressions classification,” in *2017 Int. Conf. Wireless Technologies, Embedded Intellegent Systems (WITS) 2017*, Fez, Morocco, pp. 2–5, 2017.
[37] H. Peng, C. Liu, D. Zhao, H. Ye, Z. Fang *et al.,* “Security analysis of CPS systems under different swapping strategies in IoT environments,” *IEEE Access*, vol. 8, pp. 63567–63576, 2020.
[38] Q. Tian, D. Han, K. C. Li, X. Liu, L. Duan *et al.,* “An intrusion detection approach based on improved deep belief network,” *Applied Intelligence*, vol. 50, no. 10, pp. 3162–3178, 2020.
[39] G. Hinton, “Deep belief networks,” *Scholarpedia*, vol. 4, no. 5, pp. 5947, 2009.
[40] I. Tolstikhin, O. Bousquet, S. Gelly and B. Schölkopf, “Wasserstein auto-encoders,” in *6th Int. Conf. Learning Representation ICLR 2018*, Vancouver, BC, Canada, pp. 1–20, 2018.
[41] A. I. Abubakar, H. Chiroma, S. A. Muaz and L. B. Ila, “A review of the advances in cyber security benchmark datasets for evaluating data-driven based intrusion detection systems,” *Procedia Computer Science*, vol. 62, no. Scse, pp. 221–227, 2015.
[42] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, “LSTM: A search space odyssey,” *Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2016.
[43] I. T. Bowman, R. C. Holt and N. V. Brewster, “Linux as a case study: Its extracted software architecture,” in *Proc.-Int. Conf. on Software Engineering*, Los Angeles, California, USA, pp. 555–563, 1999.
[44] Y. Fu, Y. Du, Z. Cao, Q. Li and W. Xiang, “A deep learning model for network intrusion detection with imbalanced data,” *Electronics (Switzerland)*, vol. 11, no. 6, pp. 898, 2022.
[45] K. Chen, Z. J. Yan and Q. Huo, “A Context-sensitive-chunk BPTT approach to training deep LSTM/BLSTM recurrent neural networks for offline handwriting recognition,” in *Int. Conf. on Document Analysis and Recognition, ICDAR*, Tunis, Tunisia, vol. 2015-Novem, pp. 411–415, 2015.
[46] A. K. Jain, “Data clustering: 50 years beyond K-means,” *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.
[47] Y. Cui, M. Jia, T. Y. Lin, Y. Song and S. Belongie, “Class-balanced loss based on effective number of samples,” in *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, vol. 2019-June, pp. 9260–9269, 2019.
[48] M. Sarhan, S. Layeghy and M. Portmann, “Towards a standard feature set for network intrusion detection system datasets,” *Mobile Networks and Applications*, vol. 27, no. 1, pp. 357–370, 2022.
[49] N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere and M. Smelyanskiy, “On large-batch training for deep learning: Generalization gap and sharp minima,”*5th Int. Conf. on Learning Representations, ICLR 2017-Conf. Track Proc.*, Palais des Congrès Neptune, Toulon, France, pp. 1–16, 2017.
[50] S. N. Nguyen, V. Q. Nguyen, J. Choi and K. Kim, “Design and implementation of intrusion detection system using convolutional neural network for DoS detection,” in *ACM Int. Conf. Proceeding Series*, New York, NY, United States, pp. 34–38, 2018.
[51] J. Kim, J. Kim, H. Kim, M. Shim and E. Choi, “CNN-Based network intrusion detection against denial-of-service attacks,” *Electronics (Switzerland)*, vol. 9, no. 6, pp. 1–21, 2020.
[52] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei *et al.,* “IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture,” *Computer Networks*, vol. 171, no. April 2019, pp. 107138, 2020.
[53] J. Su, V. D. Vasconcellos, S. Prasad, S. Daniele, Y. Feng *et al.,* “Lightweight classification of IoT malware based on image recognition,” in *Proc.-Int. Computer Software and Applications Conf.*, Washington, DC, USA, vol. 2, pp. 664–669, 2018.
[54] T. Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollar, “Focal loss for dense object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[55] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th Int. Conf. on Learning Representations, ICLR 2019*, New Orleans, LA, USA, 2019.

[56] T. Lawrence and L. Zhang, "IoTNet: An efficient and accurate convolutional neural network for IoT devices," *Sensors (Switzerland)*, vol. 19, no. 24, 2019.

Q5 [57] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conf. 2016, BMVC 2016*, York, UK, vol. 2016-Septe, pp. 87.1–87.12, 2016.

[58] S. Ullah, M. A. Khan, J. Ahmad, S. S. Jamal, Z. E. Huma *et al.,* "HDL-IDS: A hybrid deep learning architecture for intrusion detection in the internet of vehicles," *Sensors (Switzerland)*, vol. 22, no. 4, pp. 1–20, 2022.

[59] A. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the internet of things: Techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, no. 1, 2021.

Q6 [60] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd Int. Conf. on Learning Representations, ICLR 2015-Conf. Track Proc.*, San Diego, CA, USA, pp. 1–15, 2015.

[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury *et al.,* "PyTorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019.

Q7 [62] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conf., MilCIS 2015–Proc. 2015*, Canberra, ACT, Australia, 2015.

[63] W. Wang, M. Zhao and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 8, pp. 3035–3043, 2019.

[64] Y. Zhang, P. Li and X. Wang, "Intrusion detection for IoT based on improved genetic algorithm and deep belief network," *IEEE Access*, vol. 7, no. c, pp. 31711–31722, 2019.

Author Query Form

Journal: Computers, Materials & Continua

Article ID: TSP_CMC_30831

Article Title: Augmenting IoT Intrusion Detection System Performance Using Deep Neural Network

Q1 (p. 15)	Table 8 is not cited in the text. Please mark where the citations should be inserted in the text.
Q2 (p. 16)	Table 11 is not cited in the text. Please mark where the citations should be inserted in the text.
Q3 (p. 21)	Reference [12] is incomplete. Please provide the missing information: page range.
Q4 (p. 21)	Reference [13] is incomplete. Please provide the missing information: page range.
Q5 (p. 24)	Reference [56] is incomplete. Please provide the missing information: page range.
Q6 (p. 24)	Reference [59] is incomplete. Please provide the missing information: page range.
Q7 (p. 24)	Reference [61] is incomplete. Please provide the missing information: page range.