IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 December 2022, accepted 30 December 2022, date of publication 2 January 2023, date of current version 5 January 2023.

*Digital Object Identifier 10.1109/ACCESS.2022.3233775*

RESEARCH ARTICLE

# Transfer Learning Approach to IDS on Cloud IoT Devices Using Optimized CNN

**OGOBUCHI DANIEL OKEY[1], (Student Member, IEEE), DICK CARRILLO MELGAREJO[2], (Member, IEEE), MUHAMMAD SAADI[3], RENATA LOPES ROSA[4], JOÃO HENRIQUE KLEINSCHMIDT[5], (Member, IEEE), AND DEMÓSTENES ZEGARRA RODRÍGUEZ[4], (Senior Member, IEEE)**

[1]Graduate Program in Systems Engineering and Automation Engineering, Federal University of Lavras, Lavras, Minas Gerais 37203-202, Brazil
[2]School of Energy Systems, Lappeenranta–Lahti University of Technology, 53850 Lappeenranta, Finland
[3]Department of Electrical Engineering, University of Central Punjab, Lahore 54000, Pakistan
[4]Department of Computer Science, Federal University of Lavras, Lavras, Minas Gerais 37203-202, Brazil
[5]Center for Engineering, Modeling and Applied Social Sciences, Federal University of ABC, Santo André, São Paulo 09210-580, Brazil

Corresponding author: Dick Carrillo Melgarejo (Dick.Carrillo.Melgarejo@lut.fi)

This work was supported in part by the Lappeenranta–Lahti University of Technology, in part by the Forum for Agricultural Research in Africa cum Tertiary Education Trust Fund (FARA/TETFund), and in part by the National Council for Scientific and Technological Development (CNPq) from Brazil, under Grant 404764/2021-5.

**ABSTRACT** Data centralization can potentially increase Internet of Things (IoT) usage. The trend is to move IoT devices to a centralized server with higher memory capacity and a more robust management interface. Hence, a larger volume of data will be transmitted, resulting in more network security issues. Cloud IoT offers more advantages for deploying and managing IoT systems through minimizing response delays, optimal latency, and effective network load distribution. As a result, sophisticated network attack strategies are deployed to leverage the vulnerabilities in the extensive network space and exploit user information. Several attempts have been made to provide network intrusion detection systems (IDS) to the cloud IoT interface using machine learning and deep learning approaches on dedicated IDS datasets. This paper proposes a transfer learning IDS based on the Convolutional Neural Network (CNN) architecture that has shown excellent results on image classification. We use five pre-trained CNN models, including VGG16, VGG19, Inception, MobileNet, and EfficientNets, to train on two selected datasets: CIC-IDS2017 and CSE-CICIDS2018. Before the training, we carry out preprocessing, imbalance treatment, dimensionality reduction, and conversion of the feature vector into images suitable for the CNN architecture using Quantile Transformer. Three best-performing models (InceptionV3, MobileNetV3Small, and EfficientNetV2B0) are selected to develop an ensemble model called efficient-lightweight ensemble transfer learning (ELETL-IDS) using the model averaging approach. On evaluation, the findings show that the ELETL-IDS outperformed existing state-of-the-art proposals in all evaluation metrics, reaching 100% in accuracy, precision, recall, and F-score. We use Matthew's Correlation Coefficient (MCC) to validate this result and compared it to the AUC-ROC, which maintained an exact value of 0.9996. To this end, our proposed model is lightweight, efficient, and reliable enough to be deployed in cloud IoT systems for intrusion detection.

**INDEX TERMS** Internet of Things, convolutional neural network, cloud IoT, intrusion detection systems, transfer learning, MCC.

## I. INTRODUCTION

In recent years, the Internet of Things (IoT) has been in the spotlight of both researchers' and industrialists' activities as more sophisticated IoT systems continue to emerge and are deployed in various environments for different uses. Because of this, many people consider this technology to be a game-changer, capable of opening up previously unimaginable possibilities for human and animal community alike [1]. These emerging technologies have created an enabling environment

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Pau.

for creating, exchanging, collecting, storing, and distributing data within and among heterogeneous devices involving no human interaction and intervention. Currently, IoT devices have diverse application domains, including but not limited to health [2], imaginative home [3], smart grids [4], transportation, environment [5], infrastructure, and public services. More areas of application of this technology are discovered almost daily. Success can be attributed to the creativity of scientists who have spent years studying this topic, leading to discoveries that have positively impacted the way activities are carried out today leading to improved productivity and work-efficiency. Thus, these advancements in telecommunication systems have generated additional opportunities that ensure more effective knowledge transmission in IoT systems and also allowed the expansion into other scenarios for a more comprehensive information exchange [6], [7], [8]. The benefits of IoT devices cannot be overemphasized, as seen in home automation, industrial process automation, military warfare [4], precision agriculture [9], etc.

As a result of the increased number of IoT devices, counting in billions [10], the number of connected devices has also risen, which has caused a proportional expansion of the IoT network architecture. The expansion of the network interface comes with many benefits as well as worries. The benefits include the need for more devices and a workforce, generating jobs, and creating an atmosphere for personal development. However, the downside of this includes the issues of privacy and security concerns regarding data generation and transmission. Many times, the security of both enterprise and individual information is compromised as it is transferred from one device to another over a large expanse of the network.

Regarding storage and processing power, IoT systems are characterized by limited resource availability, and suffer from drawbacks such as security, reliability, integrity, confidentiality, and performance. One way to overcome this challenge has been the integration of IoT with the cloud environment called the Cloud of Things (CoT) [11]. Many authors have acknowledged that cloud computing is a factor in dealing with the issues associated with IoT as it provides reliability, ubiquity, and scalability combined with a high-performance environment for implementing IoT devices [12], [13], [14], [15], [16], [17], [18], [19]. However, many of the IoT applications require high computational power, very low and predictable latency, mobility support, and large-scale distributed systems, which are currently lacking in cloud computing environments. CoT has failed to assist in achieving the desired goal, hence, the need to incorporate more enabling technologies. Fog Computing with cost-effective services is capable of providing solutions to the limitations of the Cloud Computing technology by scaling its functionalities. FC has geographically distributed architecture, low latency, moderate computational resources, and mobility support, which are core needs of IoT devices.

Fog computing, an emerging technology scaling the entire network, creates challenges related to security and privacy in IoT and CoT environments considering the heterogeneity of devices operating in minimally secured environments. Network vulnerabilities increased when CC and FC were implemented, mostly because of the exponential growth in data volume. Researchers have identified Distributed Denial of Service (DDoS) and Denial of Service (DoS) attacks as significant issues that compromise the data privacy of the network usually due to the mode of operation of the attacks where an entire network can be taken down in seconds with illegitimate clients [20]. Other issues include privacy leakage, man-in-the-middle (MitM) attacks, rouge gateways, injection attacks, service manipulation, and privilege escalation. Traditional network intrusion detection and prevention techniques have been applied to mitigate these attacks, but little success has been achieved. One way of dealing with this challenge is the implementation of highly scalable deep learning models implemented in the fog and cloud environments [19].

A network intrusion detection system (NIDS) is one of the many ways to manage computer network security threats. NIDS monitors every network activity for abnormal or malicious intent and generates warnings when such threats are discovered. This mechanism can either be signature-based, where a well-defined rule is used to detect normal and abnormal network flows, or anomaly-based, where profiles of events are used to decide the nature of the such activity. Mostly, the anomaly-based approach is more accurate than signature-based NIDS, as the latter fails to identify activities that have not been previously defined in its database. In implementing NIDS, it can either be network-based or host-based. Deep learning techniques have achieved great success in various fields of artificial intelligence (AI), such as computer vision, image processing, and pattern. recognition, natural language processing, image captioning, pharmaceutical research, etc. Also, Deep Learning has been applied to develop IDS systems for CoT systems. Fig. 1 shows a typical architectural representation of the Cloud IoT connection.

Convolutional Neural Networks (CNN) is a deep learning architecture that has predominantly gained researchers' attention due to its exceptional performance in handling image data in computer vision, image recognition, and segmentation. Hence, it has been applied to various areas, including networking and medical image processing [21] CNN has also worked efficiently on numeric tabular datasets used for modeling IDS. Classical machine learning tree-based algorithms including Decision Tree (DT), Random Forest (RF), Extra Tree (ET), eXtreme Gradient Boosting (XGBoost) etc have shown good performances in classification of network traffic that are not represented as image data [22], [23]. Other deep learning algorithms such as Recurrent Neural Network (RNN) [24] have been used for time series data analysis with good accuracy but this still suffers from the challenge of varnishing and exploding gradient which is overcome with the CNN architecture. Hence, on image data, CNN is preferred due to the high performance in image classification [25].

In this paper, we propose an efficient-lightweight ensemble transfer learning (ELETL-IDS) model based on CNN to detect the most prominent attacks (Bot, Infiltration, DDoS,

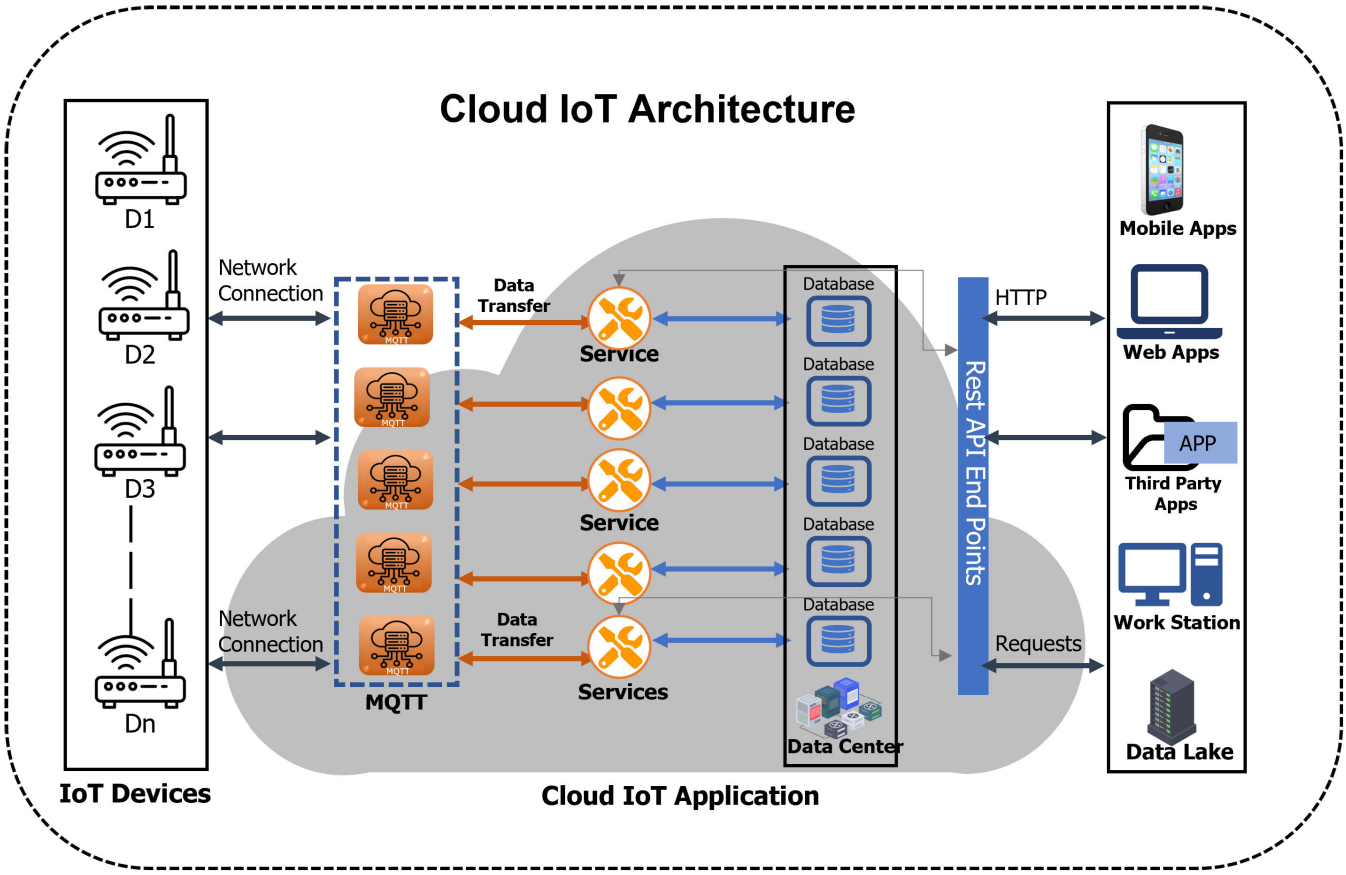



**FIGURE 1.** Typical Representation of the architecture of Cloud IoT connection.

and DoS) in Cloud IoT environments using real-time datasets. We provide an effective way to secure IoT devices by detecting specific attacks. Other works have used machine learning algorithms to develop the models [22], [26], [27], but using ensemble transfer learning (ETL), we present a more accurate and reliable model.

The proposed ELETL-IDS model on evaluation achieves an accuracy, validation accuracy, precision, and F1-score of 100% and 99.98% on the CIC-IDS2017 and CSE-CIC-IDS2018 datasets, respectively. These values show that the model is highly efficient in detecting attacks using multi-class classification. The major contributions of this research can be summarized as follows:

- An efficient-lightweight intrusion detection system is proposed for detecting a wide range of CoT attacks using transfer learning based on CNN architecture.
- To make the input data suitable for the CNN architecture, a method to convert the numeric tabular data into an image representation, was developed.
- A cross-platform IDS capable of operating in the cloud IoT environment is proposed. The model reduces training loss and validation loss with regularization.
- Using Bayesian Optimization - Tree Parzen Estimator (BO-TPE), we perform Hyper-parameter Optimization to determine the best parameters to train the proposed model.

It is important to highlight that the suggested approach has been designed for cloud IoT environments taking into accounts the limitations of IoT systems such as resource constraints. In the experimental performance evaluation, in conjunction with well-known evaluation metrics, we use Matthew's Correlation Coefficient (MCC) to validate the results of the confusion matrix, the ROC-AUC curve. The results demonstrated that the proposed model is reliable, efficient, and outperforms other works in the literature and show that it is possible to use image representation of network traffics to develop a predictive model capable of detecting even zero-day attacks on IoT networks with high accuracy.

## II. RELATED WORKS

Considering the efficiency of DL-based models in accurately providing solutions to IDS problems in IoT and CoT environments, some authors have proposed IDS models using the DL methods. Lin et al., [22] proposed an IDS model based on a special variant of the Recurrent Neural Network (RNN) architecture, the Long Short-Term Memory (LSTM) network for anomaly detection. The LSTM architecture overcomes the varnishing and exploding gradient descent which are common challenges with RNN by learning long term dependencies during training. Combining SMOTE and improved cost functions, the authors handled the data

imbalance in the dataset that was used to train the model, achieving an accuracy of 96.2% on the CSE-CICIDS2018 dataset. In [26], a two-layer multilevel classifier fusion strategy for network intrusion detection was presented, with the upper layer employing an unsupervised feature extraction method utilizing a non-symmetric deep autoencoder and the lower layer employing a random forest classifier with the extracted features. The method experimented with two different datasets, the KDDCUP'99, and the NSL-KDD datasets. For a 5-class classification, the proposed method achieved 97.85% and 85.42% accuracy with the KDDCUP'99 and NSL-KDD datasets, respectively. In experimenting with different hidden layers for binary and multi-class classification of network traffic data in a CNN task, a framework was proposed and evaluated on 6 different datasets, including NSL-KDD and KDDCup'99 in [27]. On evaluation, the obtained results show that the best performing model was composed of 5 hidden layers, which reached an accuracy of 92.7% and 78.9% on the KDDCUP'99 and NSL-KDD datasets for binary classification, respectively, and 92.5% and 78.5% accuracy on the KDDCUP'99 and NSL-KDD datasets, respectively, concerning multi-class classification.

Deep transfer learning (DTL) is a solution that can reuse existing knowledge from the previously trained model and achieve better intrusion detection performance than other models [28], [29]. Yang et al. [28] proposed IDS based on different TL models trained on the imagenet dataset, including VGG-16, VGG-19, InceptionNet, ResNet, and InceptionRestNet to develop the model on the CIC-IDS2017 and Car-Hacking datasets. The transfer learning (TL) model proposed by Taghiyarrenani et al., [30] for intrusion detection shows more efficient performance in both labeled and unlabeled data. To extract the attack invariant from the existing attack data set and transfer the knowledge to the target network system, Yanjie et al., [31] proposed a framework for transfer-learning-based network flow generation for deep-learning-based IDS.

CNN is well known to be very efficient in image processing tasks. Network traffic data is usually in numeric and categorical format, which needs to be converted to image format for faster processing with CNN networks. In this regard, some authors have implemented this technique in their works. Li et al. in [32], proposed an image conversion method of network traffic data for the NSL-KDD dataset. They have used representation learning using ResNet 50 and GoogLeNet with an accuracy of 81.15%. In [33], Harsh Dhillon et al. proposed a deep transfer learning model using CNN-LSTM, CNN, and DNN and evaluated the model on the UNSW-NB-15 dataset for NIDS. The results showed that the CNN-LSTM with an accuracy of 98.43% could help minimize successful network attacks. One drawback of the work was the data imbalance issue, which was not handled. To address the new and emerging challenges related to the accuracy, efficiency, scalability, and dependability of the traditional IDS in a heterogeneous IoT setup, Mehedi et al., [34] proposed deep transfer learning-based dependable IDS for IoT systems using the ResNet model in which accuracy of 88% was achieved. In [25], the NSL-KDD dataset was transformed into image format for transfer learning to develop a model for IDS named Transfer Learning Network Intrusion Detection (TL-IDS). The authors implemented the dataset using the VGG-16 architecture as the base model and transferred weight to a novel CNN interface for the task. According to [35], a TL-based method called TL-ConvNet was introduced that acquires information from a base dataset and applies it to the acquisition of the target dataset. The UNSW-NBI5 and NSL-KDD as the base and target datasets, respectively, for the TL-ConvNet. Experimental results of TL-ConvNet with 87.30% and 81.9% accuracy on the datasets showed significant performance improvement over conventional CNN by the TL-ConvNet, yet has high FPR as a drawback.

To our knowledge, few published papers are in the domain of transfer learning for intrusion detection. Hence, our proposed model was optimized with different parameters and hyperparameters and experimented with two different test datasets from a real-world network intrusion dataset.

## III. PROPOSED FRAMEWORK

### A. PROBLEM STATEMENT AND SYSTEM OVERVIEW

The Internet of Things (IoT) is faced with increasing security concerns, threatening information confidentiality, integrity, and availability (CIA). Cloud-based IoT systems face an unlimited number of malware, spyware, and Man-in-the-middle (MITM) attacks, including ransomware, DDoS, and DoS. A highly efficient yet lightweight intelligent DL model is required to provide deep attack detection and repelling on these systems that communicate 24 hours a day over the internet. The layout of the framework used in this research is shown in Fig. 2.

We develop an optimized, lightweight, ensemble transfer learning model based on CNN for a CoT environment capable of repelling network attacks. In the first phase, data acquisition is carried out by selecting from existing datasets the database that contains IoT-based network attacks as discussed in [36]. The CIC-IDS2017 and CSE-CIC-IDS2018 with the highest numbers of real-time data points were selected for the research. During EDA, we observed that the dataset contains imbalance; therefore, we used two methods of over-sampling technique to obtain balanced datasets. The methods include Synthetic Over Sampling TEchnique (SMOTE), and Borderline_SMOTE with RandomUnderSampling. Since these databases contain information in time-based chunks, they are transformed into images for the CNN architecture using the quantile transform method. In the following stage, classical CNN (CNN-C), and other selected CNN-based models, including VGG16, VGG19, MobileNetV3, EfficientB0, and InceptionNetV3, were trained on the generated images to design the based learners for our ensemble model. Also, for optimum performance and to obtain a highly reliable mode, the CNN models are optimized by Bayesian

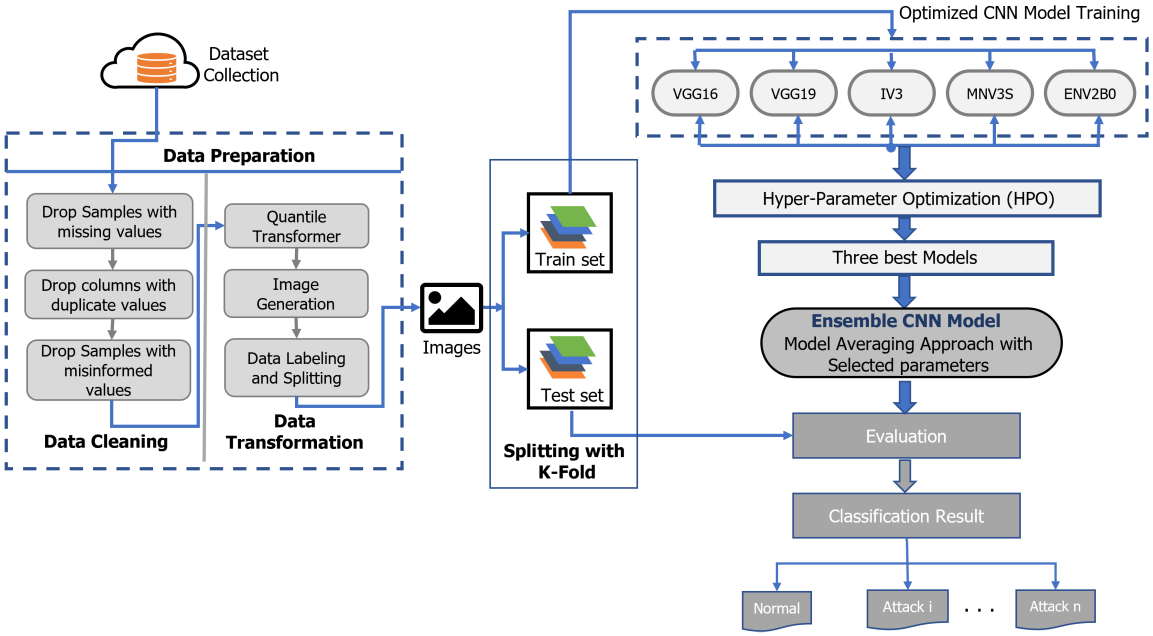



**FIGURE 2.** System Design Framework for the Proposed Ensemble Transfer Learning Model.

Optimization - Tree Parzen Estimator (BO-TPE), a Hyper-parameter Optimization (HPO) method that can automatically tune the hyper-parameters. In the end, the top-3 best-performing CNN models are selected as the three CNN models used to construct the ensemble learning models using the model averaging method.

### *B. DATABASE DESCRIPTION, TRANSFORMATION AND IMAGE GENERATION*

To develop the proposed IDS for the IoT and CoT environments, two different databases (DBs) are used. The first is the CIC-IDS2017 [37], [38] that has 80 features and 15 different labels representing 14 attacks and 1 standard traffic payload that were generated in real-time over 5 days with 25 user behaviors over HTTP, HTTPS, FTP, SSH, and email protocols and contains such attacks as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. The complete CSV file for the dataset is contained in 7 folders and needs to be merged into a single file in order to have all the instances together, thereby improving the dataset and enabling better model learning on a complete dataset. This was achieved, and a total of 85 features were obtained with 2.3 million data instances consisting of 83.3% benign and 16.7% attacks. Out of the 85 features, it was observed that five features were repeated; therefore, they were dropped to arrive at the 80 features earlier stated. The whole labels in the dataset are summarized into 8 attacks, including Bot, Brute Force, DoS, DDoS, Infiltration, Heartbleed, PortScan, WebAttacks, and Benign.

The second dataset used is the CSE-CICIDS2018 [39] that extends the features of the CIC-IDS2017 dataset by including more HTTPS, HTTP, SMTP, POP3, IMAP, SSH, and FTP requests. Using the CICFLOWMETERV3 [40], more than 80 features can be extracted from the PCAP file obtained from the testbed during simulation and data capture. The extracted CSV file format contains six columns of FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol, together with 80 network traffic types. The six columns are not crucial in deciding the network attacks, so they are not used in the modelling. A total of 16 million data instances with 15 different attack types were extracted from the 10 different CSV files obtained from the CICFLOWMETERV3. Finally, the 15 attacks were merged into 6 attacks and benign to include Bot, Brute Force, DoS, DDoS, Infiltration, and Web attack where the Benign has 83% and attack measure 17%.

As part of the data preprocessing phase, the datasets are cleaned of all missing or misinformed data values. The datasets were checked for samples that contain missing values and inadequate values like nan, −inf, +inf, etc. These samples were dropped due to the large volume of the dataset. In addition, we determined that some of the features have constant values and hence do not contribute to the ML learning process during the training. It is not critical to have such features as they will only increase overhead costs and computational resources. These features, such as *Protocol, Fwd PSH Flags, SYN, Flag Cnt, Active Std, FIN Flag Cnt, Fwd URG Flags, CWE Flag Count, Fwd Blk Rate Avg, Bwd Byts/b Avg, Fwd Byts/b Avg, Bwd PSH Flags, Fwd Pkts/b Avg, Bwd Blk Rate Avg, and Bwd URG Flags* were dropped. To confirm that this approach is appropriate, we performed feature selection using Random Forest Feature Importance (RFFI) provided by the Sklearn Library. Using standard deviation, the feature importance ranks the network traffic according to its

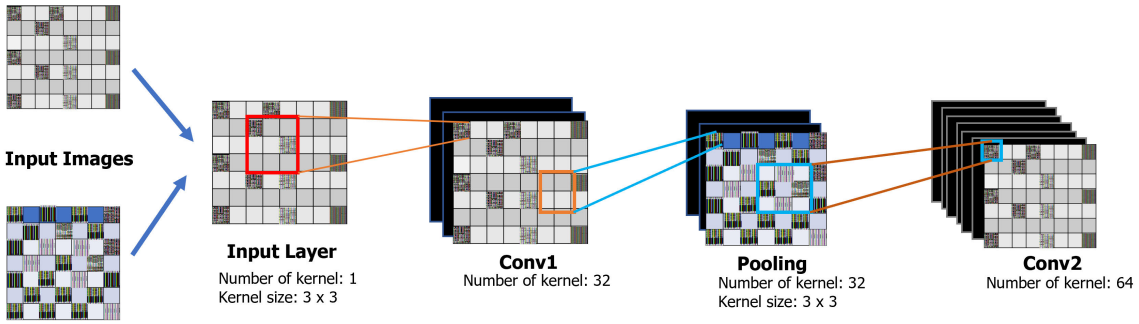



**FIGURE 3.** Convolution Operation on the Image using 3 Channel filters.

importance. Whereas 15 features were dropped, 64 features and labels were used in the model training out of 80 features.

To handle the data imbalance, we generate synthetic samples of the existing data points using the method proposed by Chawla et al. al [41] called SMOTE and Borderline_Smote proposed by [42]. Unlike the undersampling techniques, which randomly remove some of the majority class, the SMOTE and Borderline_Smote generate synthetic instances of the minority class based on the existing data points, thereby ensuring no information loss. Assuming that $T$, $N$ and $k$ defined by SMOTE(T, N, k) are input to the SMOTE algorithm where $T$ is the number of minority class samples; $N$ is the amount of SMOTE, and $k$ is the number of nearest neighbors, the output of the operation can be defined as $(N/100) * T$ synthetic minority class samples [41].

After the preprocessing stage, we must prepare the data to be a suitable input to the CNN architecture. As earlier stated, CNN achieves better performance when working on image data. Since our data is tabular in the *CSV* file format, we need to carry out transformation to obtain image samples according to [28] and [43]. There are mainly three basic transformation algorithms that are frequently used for data transformation. They are Normalization with Min Max Scaler (MMS), Quantile Transformation (QT), and Power Transformation (PT). MMS, defined by Equation 1 is known to be the most commonly used normalization technique, but its drawback is hinged on its inability to handle outliers completely; thus, it may lead to some values being extremely small. For this reason, we adopted the QT method proposed in [44].

$$X' = \frac{X - Min(X)}{Max(X) - Min(X)} \times 255 \tag{1}$$

The QT normalization method transforms the feature distribution to a normal distribution. It then re-calculates all the feature values based on the normal distribution. Therefore, the majority of variable values are close to the median values, which is effective in handling outliers. QT put all features into the same desired distribution based on:

$$Q = G^{-1}(F(x)) \tag{2}$$

where $F$ is the cumulative distribution function of the feature and $G^{-1}$ is the quantile function of the desired output distribution $G$ [45], [46].

During the image generation phase, based on the timestamp and feature size of the network traffic dataset, the data samples are converted into chunks of various sizes. In the case of our dataset, each of them has 64 important features, which we converted into 3D image format of 3 channels representing the color of the images using the pattern shown in Fig. 2. Therefore, each color image generated is transformed to 64 x 64 x 3 total feature values. By implication, the first 64 samples of each chunk were converted into the image matrix of channel 1, the next 64 samples of each chunk were converted into the image matrix of channel 2, and the last 64 samples of each chunk were converted into the image matrix of channel 3, and all are generally mapped into the RGB channels of the image. The conversion of the data into the image matrix is achieved using OpenCV library. The data was converted into a 3D image format to improve the learning capacity of the model since more filters are required in this case, compared to the 2D image representation. During the convolution operation in the 3D representation, three filters of 3 x 3 x 3 are required, resulting in a 4 x 4 x 1 image, as shown in Fig. 3. These filters are edge detectors that allow the model to learn better on the input data. Each dataset chunk consists of 64 X 3 = 192 consecutive data samples. This step is repeated until all the labels in both datasets are correctly transformed. It is important to note here that the time-series correlation of the original network traffic data can be retained since the images generated are based on the timestamps of the sample data; hence, the obtained results are assured of correctness. Furthermore, converting the data into an image vector with three channels is important in this scenario as it helps to easily identify the different patterns in the images through color

Following the transformation, we label the generated images based on the chunks' attack pattern. For instance, the image is labeled "Normal" if the sample in the chunk/image is normal. Also, if the sample in the chunk/image contains attack samples, the image is labeled with the corresponding frequent attack type in the chunk. as either "DDoS Attack" or

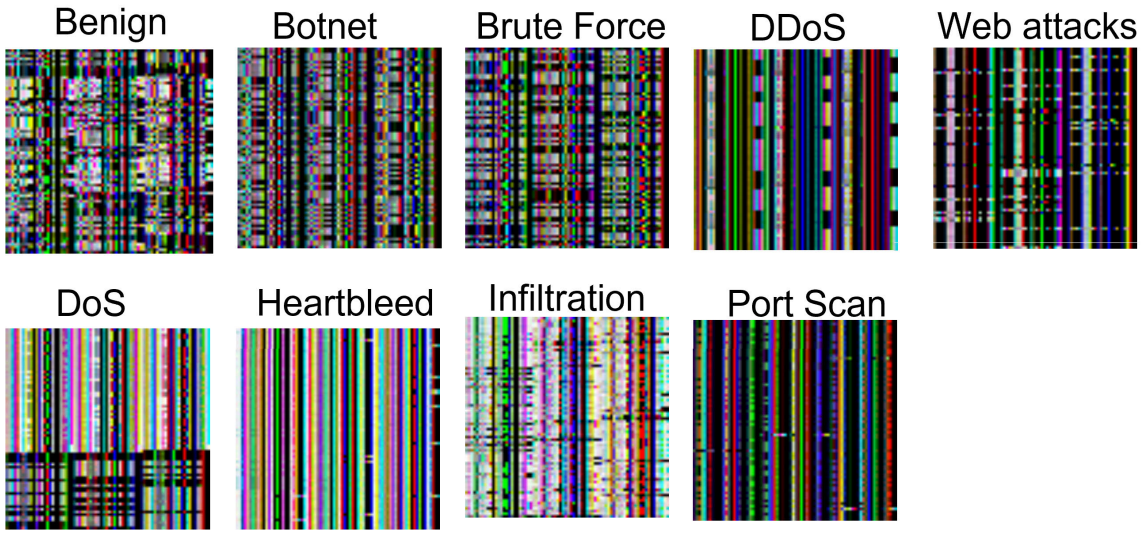



**FIGURE 4.** Images obtained from the CSE-CIC-IDS2017 dataset after conversion.

''Web Attack,'' ''Infiltration'', ''Botnet'', ''Portscan'', ''DoS attack'', ''Brute Force'' as the case may be.

The last pre-processing procedure is re-scaling the image into a suitable input format for the CNN models. The images generated are 64 X 64 X 3, but the pre-trained models such as VGG16, MobileNetV3, and others expect images in the 224 X 224 X 3 range. So, we resized the images to 256 X 256 X 3. This will allow the CNN to learn all the patterns easily in the image and improve the learning speed as the filters convolute the images. Representative samples of the image sample of the CIC-IDS2017 and CSE-CICIDS2018 datasets are shown in Fig. 4 and Fig. 5 respectively.

From both datasets, it can be seen that there are significant differences in the feature patterns between the standard samples and different types of attacks. The feature patterns of Web attack images in Fig. 5 are more random, concentrated at the top and bottom of the images, and sparse at the center, while in Fig. 4, are somewhat denser throughout the image. The difference in the patterns of the images results in the frequency and techniques of each attack strategy employed by the attacker. In addition to these variations, the CNN model can learn more features, making it more robust for attack detection and classification.

### C. CLASSICAL CNN AND TRANSFER LEARNING

CNN is widely used for its exceptional properties of automatically learning and extracting features from images for image recognition and segmentation tasks [21]. Images can quickly be supplied as inputs into the CNN without additional feature description, extraction, and reconstruction processes. A typical CNN comprises convolutional, pooling, and fully connected layers. In convolutional layers, the feature patterns of images are automatically extracted by convolution operations using filters of different sizes, which can be 4 X 4, 3 X 3, etc. In our research, we used the 3 X 3 filter to perform the convolution operation, leading to feature learning and extraction. The data complexity can be reduced in pooling layers without losing essential information through local correlations to avoid over-fitting. Optionally, dropout and batch normalization layers can also be added to improve the performance of the learning process depending on the input parameters and model architecture. The Fully-connected layers serve as a channel to connect all feature maps and generate the output.

Transfer Learning (TL) is a system in DL where the weights of existing models trained on large amounts of data are reused in another dataset [21]. Sometimes, TL may be the best option for image classification tasks where datasets are insufficient, as CNN requires many datasets for good performance. The successful application of TL in image tasks is because the feature patterns learned by the bottom layers of CNN models are usually general patterns that apply to many different tasks, and only the features learned by the top layers are specific features for a particular dataset. Therefore, the bottom layers of CNN models can be directly transferred to different tasks. To improve the effectiveness of TL, fine-tuning can be used in the TL process of DL models. In fine-tuning, most of the layers of the pre-trained model are frozen (i.e., their weights are retained), while a few of the top layers are unfrozen to re-train the model on a new dataset. Fine-tuning enables the learning model to update the higher-order features in the pre-trained model to better fit the target task, or dataset [28]. One of the most critical issues in the TL methods is unifying the distribution of the source and target samples [47]. Maximum mean disparency (MMD) [48] is one of the successful distribution distance estimators used frequently in TL methods. MMD estimates the distance between two distributions based on Reproducing Kernel Hilbert Space (RKHS) [48].

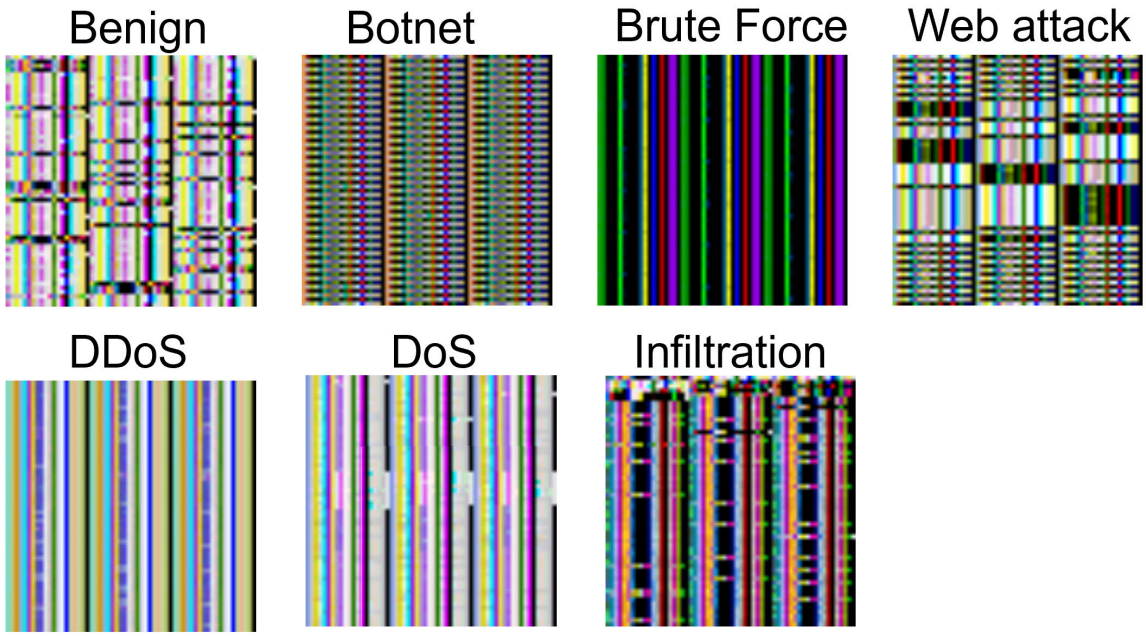



**FIGURE 5.** Images obtained from the CSE-CIC-IDS2018 dataset after conversion.

In the proposed architecture Fig. 2, we have selected VGG16, VGG19, MobileNetV3 (MNV3), EfficientNetV2 (ENV2B0), and InceptionV3 (IV3) as the base learners due to their success in image classification tasks [21], [29]. These are pre-trained CNN models on the ImageNet dataset, a benchmark dataset for image classification tasks consisting of over 14 million images of over 1000 classes [29]. The VGG architecture comprises convolutional layers (CLs) and rectified linear unit (ReLU) activation functions. The VGG comes in two different variations, VGG16 and VGG19, with the VGG16 having a total of 16 layers and the VGG19 having 19 layers; both implement the $3 \times 3$ filter dimensions for the CLs [49]. Due to the wide adoption [29], [49] and ease of implementation of the VGG architecture with both the 16 and 19 layers variations, we have used both in this work to demonstrate its applicability in IDS systems. Another CNN-based pre-trained model architecture that has high relevance in the application is the Inception model published by GoogLeNet, having the V1, V2, and V3 variations [50]. The V2 and V3 are the improved versions of the original V1 architecture. The V2 modified the V1 with the inclusion of batch normalization layers [50] for training streamlining and improved performance, while the V3 included larger spatial features and factoring convolutions for improved computational efficiency. We have used the InceptionV3 (IV3) architecture in our work as it is more flexible and lightweight than the earlier versions regarding memory requirements and trainable parameters. IV3 has a file size of 92 MB, and 23.9 million trainable parameters [51].

As the depth, width, and resolution of the models' architecture increase, more computational cost is required for training. To overcome this, the study by Tan and Quoc [52] proposed the EfficientNet CNN architecture that dynamically grows all the dimensions efficiently using a simple composite coefficient. Based on mobile inverted bottleneck convolution (MBConv), the various versions of the EfficientNet models, ranging from EfficientNetB0 to EfficientNetB7, were developed. The EfficientNetV2 has higher performance advantages, including faster training speed and better parameter efficiency, than previous models [53]. This model family has variations of EfficientNetV2B0-B4 and EfficientNetV2S, EfficientNetV2M, and EfficientNetV2L that were developed using a combination of training-aware neural architecture search and scaling to jointly optimize training speed and parameter efficiency. The models were searched from the search space enriched with new ops such as Fused-MBConv to obtain a model that trained faster than state-of-the-art models while being up to 6.8x smaller. Based on the literature, we selected the EfficientNetV2B0 (ENV2B0) with 79MB of memory capacity and 7.2 million training parameters [52], [53].

Howard et. al., [54] proposed another class of efficient models called MobileNets that target mobile and embedded devices applications. Using a streamlined architecture that operates on depth-wise separable convolutions (DWSC), the authors developed lightweight CNN pre-trained models with better performance on edge devices. Two simple global hyper-parameters that permit a sufficient trade-off between accuracy and latency were implemented, thus, granting model builders the privilege to define and choose the best model sizes for their projects. Like other pre-trained models, the MobileNet comes in different forms, including the MobileNet, MobileNetV2, and MobileNetV3 (Small and Large) [54], [55], [56]. We selected MobileNetV3Small

(MNv3S) for this work, with 2.9 million trainable parameters and 14 MB of memory consumption.

In all, for the TL part of the proposed model training process, the VGG16, VGG19, MNV3S, ENV2B0, and IV3 were trained on the datasets as the base models for IDS. The top three (top-3) performing models are selected to construct the ensemble model proposed in this work.

### D. ELETL-IDS: PROPOSED ENSEMBLE TRANSFER LEARNING MODEL

In this section, we discuss the proposed ensemble model. Ensemble learning is a technique that combines various base learning models to create an enhanced single model. Because an aggregation of several learners performs better than a single learner, ensemble learning is commonly utilized in data analytics challenges, including network attack detection [57], [58].

Model averaging is an ensemble technique where multiple sub-models contribute equally to a combined prediction. This may lead to overfitting, a profound challenge in ML tasks. A variation of this approach, called a weighted average ensemble, weighs the contribution of each ensemble member by the trust or expected performance of the model on a holdout dataset. This allows well-performing models to contribute more and less-well-performing models to contribute less. The weighted average ensemble improves the model average ensemble, which has been used in this work. With the Softmax activation function, which returns the predicted probability for each class in a classification task, we obtain the confidence of each class. The model averaging method calculates the average classification probability of base learners for each class and then returns the class label with the highest average confidence, which is the output of the softmax function as discussed in [59].

$$\text{Softmax}(\mathbf{z})_i = \frac{e^{\mathbf{z}_i}}{\sum_{j=1}^{C} e^{\mathbf{z}_j}} \tag{3}$$

where $\mathbf{Z}$ is the input vector, $\mathbf{C}$ is the number of classes in the dataset, $e^{z_i}$ and $e^{z_j}$ are the standard exponential functions for the input and output vectors, respectively. The predicted class label obtained by the model confidence averaging method can be defined by Equation 4. This is obtained by taking the *agrgmax* of the predicted labels in the selected class

$$\hat{y} = \underset{i\in\{1,\cdots,c\}}{\text{argmax}} \frac{\sum_{j=1}^{k} p_j\left(y = i \mid B_j, \boldsymbol{x}\right)}{k} \tag{4}$$

where $B_j$ is the $j_{th}$ base learner, $k$ is the number of selected base CNN learners, and k = 3 in the proposed IDS; $p_j(y = i|Bj, x)$ indicates the prediction confidence of a class value $i$ in a data sample $x$ using $B_j$.

Given that $N\_m$ is the number of instances, $K\_b$ is the the number of base CNN models and $C\_s$ is the number of classes, we can calculate the computational complexity of the ELETL-IDS and that of the time for the model averaging method as $O(N\_mK\_bC\_s)$. With small values of $K\_b$ and $C\_s$, the speed of execution of the averaging method is usually high.

### E. BEST MODEL THROUGH HYPER-PARAMETER OPTIMIZATION (HPO)

One of the most essential steps in ML/DL tasks is handling hyper-parameters (HPs). While the model adjusts the model parameters during the learning process to achieve optimal results, we can fine-tune the hyper-parameters such that they aid the model in achieving the best performance at the optimal time. Such hyper-parameters as learning rate, epochs, batch normalization, dropout, and weights are usually tuned for the model. In the proposed TL framework, the dropout rate, the learning rate, and the percentage of frozen layers are defined as model-design parameters, while batch size, epochs, and early stop patience level are model-training hyper-parameters aimed at increasing the model training speed and overall performance. These parameters directly influence the CNN models' architecture, efficiency, and effectiveness.

HPO is an automated process of selecting the best parameters for model performance in ML or DL using optimization techniques. Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Random Search (RS), and Bayesian Optimization - Tree Parzen Estimator (BO-TPE) [60] are some of the many search algorithms for performing HPO. The TPE algorithm is implemented on Hyperopt (a library for hyper-parameter tuning with bayesian optimization in Python). We have adopted the BO-TPE [61] technique in this research due to its ability to keep conditional dependencies yet provide optimal performance results with efficiency with all types of HPs.

### F. MODEL TRAINING AND EVALUATION

In the experimental setup for this task, we have used Python, Numpy, Pandas, Matplotlib, and the machine learning library sci-kit learn, as well as deep learning libraries (*Keras*) and (*Tensflow*) for the software. The code were executed on a computer running on Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 3600Mhz, 4 Core(s), 16 GB (15.9 GB usable), Windows 10 Home Single Language 64-bit and NVIDIA GeForce GTX 1050 Ti GPU.

The proposed architecture is evaluated on two benchmark datasets, CIC-IDS2017 and CSE-CIC-IDS2018, as discussed in Section III-B. *StratifiedKFold* split with 10-folds was used to split the dataset into training, test, and validation, ensuring an even distribution of the datasets in the folds, hence overcoming model overfit. Usually, network traffic datasets are composed of a high volume of expected traffic and fewer attacks, so we handled the data imbalance. Table 1 shows the total number of images generated for each dataset. 19,055 images were generated for the CIC-IDS2017 dataset consisting of 9 classes, and 21,230 images comprising 7 classes for the CSE-CIC-IDS2018 dataset.

Generally, for ML or DL classification tasks, the accuracy, precision, recall, and F1-scores, are usually implemented for performance evaluation. We extend the evaluation metrics by

**TABLE 1. Distribution of generating images of the datasets used in the model training and evaluation.**

| Classes | CIC-IDS2017 | CSE-CICIDS2018 |
|---|---|---|
| Benign | 2575 | 4470 |
| Bot | 2062 | 2241 |
| Brute Force | 2057 | 3004 |
| DDoS | 2062 | 4189 |
| DoS | 2060 | 3614 |
| Heartbleed | 2057 | - |
| Infiltration | 2060 | 2036 |
| Port Scan | 2059 | - |
| Web Attack | 2063 | 1676 |
| **Total** | **19055** | **21230** |

including the AUC Score and Matthew's Correlation Coefficient (MCC) metrics. The AUC score can be evaluated using the function defined by Equation 5 and 6

$$AUC = \frac{1}{c(c-1)} \sum_{j=1}^{c} \sum_{k>j}^{c} (\text{AUC}(j \mid k) + \text{AUC}(k \mid j)) \quad (5)$$

where c is the number of classes and AUC($j \mid k$) is the AUC with class j as the positive class and class k as the negative class. In general, AUC($j \mid k$) $\neq$ AUC($k \mid j$) in the multiclass case [62].

Equation 6 extends Equation 5 to be used for calculating roc-auc curve which are weighted by prevalence. This algorithm is used by setting the keyword argument multiclass to 'ovo or OVR' and average to *'weighted'*. The *'weighted'* option returns a prevalence-weighted average as described in [63].

$$AUC = \frac{1}{c(c-1)} \sum_{j=1}^{c} \sum_{k>j}^{c} p(j \cup k)(\text{AUC}(j \mid k) + \text{AUC}(k \mid j)) \quad (6)$$

On the other hand, the MCC provided by Sci-kit learn library is available in the package *K.matthews_corrcoef (y_true, y_pred, *, sample_weight=None)* where $K$ represents the sklearn.metrics function. MCC is used to measure the quality of binary or multiclass classification tasks. It accounts for the true and false positives and negatives and is generally regarded as a balanced measure that can be used even if the classes are of very different sizes. The MCC is a correlation coefficient value between $-1$ and $+1$. A coefficient of $+1$ represents a perfect prediction, 0 is an average random prediction, and -1 is an inverse prediction. The statistic is also known as the phi coefficient [64], [65]. Given that *tp*, *tn*, *fp* and *fn* are the true positive, true negative, false positive, and false negative outputs of a classification problem in a confusion matrix, the MCC for a binary classification can be represented with the function in Equation 7.

$$MCC = \frac{tp \times t\text{n-f}p \times fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}} \quad (7)$$

In the multiclass case, the Matthews correlation coefficient can be defined in terms of a confusion_matrix $C$ for $K$ classes. To simplify the definition consider the following intermediate variables:

- $t_k = \sum_i^K C_{ik}$ the number of times class $k$ truly occur.
- $p_k = \sum_i^K C_{ki}$ the number of times class $k$ was predicted.
- $c = \sum_k^K C_{kk}$ the total number of samples correctly predicted.
- $s = \sum_i^K \sum_j^K C_{ij}$ the total number of samples.

Then, the multiclass MCC can be defined as given in Equation 8.

$$MCC = \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{\left(s^2 - \sum_k^K p_k^2\right) \times \left(s^2 - \sum_k^K t_k^2\right)}} \quad (8)$$

In a situation where more than one labels exist, the value of the MCC will no longer range between $-1$ and $+1$. Instead the minimum value will be somewhere between $-1$ and 0 depending on the number and distribution of ground true labels. The maximum value is always $+1$ [46].

## IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we discuss the results obtained from our experiment. Initially, the selected CNN architectures are trained on the datasets with all the initial model parameters. The performance of the model on the training data and test data are evaluated. Considering the performance, we performed model tuning. After performing the HPO with the BS-TPE algorithm, the best hyper-Parameter were used to re-train the model. During the search, the parameters used are shown in Table 2. The number of epochs, batch size, learning rate, dropout rate, early stopping patience, and the number of frozen layers have been optimized as shown in Table 2. The early stopping patience is used to save training time as the model saves the best performances obtained during the training and stops training when the validation accuracy does not increase between two consecutive epochs. This is because the validation accuracy is monitored during the training as it helps to determine if the model overfits or not.

**TABLE 2. Hyper-Parameters obtained after BS-TPE optimization for the Model Configuration.**

| Hyper-Parameter | Model | Search Range | Optimal Value |
|---|---|---|---|
| Number of Epochs | All CNN pre-trained Models | [5,30] | 15 |
| Batch size | | [32,256] | 128 |
| Early stop patience | | [1,4] | 2 |
| Learning rate | | [0.001,01] | 0.003 |
| Dropout rate | | [0.2, 0.8] | 0.5 |
| Number of layers frozen from the pre-trained layers | MobileNetV3Small | [10,20] | 10 |
| | EfficientNetV2B0 | [5,10] | 16 |
| | InceptionV3 | [80,159] | 148 |
| | VGG16 | [8,16] | 15 |
| | VGG19 | [10,19] | 19 |

Each of the selected CNN-based networks was trained on both datasets, and the results obtained in each case are shown in Table 3 and Table 4 for the CIC-IDS2017 and

**TABLE 3. Performance Evaluation of Optimized and non-Optimized trained Models on CIC-IDS2017.**

| Models | Accuracy | Precision | Recall | F-Score | AUC | MCC |
|---|---|---|---|---|---|---|
| **Base CNN** | 0.9976 | 0.9976 | 0.9976 | 0.9976 | 0.9980 | 0.9984 |
| **VGG16** | 0.9993 | 0.9993 | 0.9993 | 0.9993 | 0.9995 | 0.9998 |
| **VGG19** | 0.9981 | 0.9981 | 0.9981 | 0.9981 | 0.9982 | 0.9988 |
| **IV3** | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9993 | 0.9992 |
| **MNV3S** | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9998 | 0.9995 |
| **ENV2B0** | 0.9982 | 0.9982 | 0.9982 | 0.9982 | 0.9986 | 0.999 |
| **Base CNN - HPO** | 0.9978 | 0.9978 | 0.9978 | 0.9978 | 0.9979 | 0.998 |
| **VGG16 - HPO** | 0.9996 | 0.9996 | 0.9996 | 0.9996 | 0.9995 | 0.9997 |
| **VGG19 - HPO** | 0.9994 | 0.9994 | 0.9994 | 0.9994 | 0.9996 | 0.9992 |
| **IV3 - HPO** | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9999 | 0.9999 |
| **MNV3S - HPO** | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9999 | 0.9998 |
| **ENV2B0 - HPO** | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9989 | 0.9988 |
| **ELETL-IDS** | 1.00 | 1.00 | 1.00 | 1.00 | 0.9999 | 0.9999 |

**TABLE 4. Performance Evaluation of Optimized and non-Optimized trained Models on CSE-CIC-IDS2018.**

| Models | Accuracy | Precision | Recall | F-Score | AUC | MCC |
|---|---|---|---|---|---|---|
| **Base CNN** | 0.9797 | 0.9806 | 0.9797 | 0.9799 | 0.9798 | 0.9800 |
| **VGG16** | 0.9880 | 0.9897 | 0.9897 | 0.9897 | 0.988 | 0.9886 |
| **VGG19** | 0.9887 | 0.9887 | 0.9887 | 0.9887 | 0.9877 | 0.9890 |
| **IV3** | 0.9769 | 0.9777 | 0.9769 | 0.9770 | 0.9796 | 0.9775 |
| **MNV3S** | 0.9832 | 0.9833 | 0.9832 | 0.9831 | 0.9852 | 0.9854 |
| **ENV2B0** | 0.9890 | 0.9889 | 0.9890 | 0.9890 | 0.9870 | 0.9895 |
| **Base CNN - HPO** | 0.9890 | 0.9890 | 0.9890 | 0.9890 | 0.9880 | 0.9892 |
| **VGG16 - HPO** | 0.9899 | 0.9899 | 0.9899 | 0.9899 | 0.99 | 0.9912 |
| **VGG19 - HPO** | 0.9900 | 0.9900 | 0.9900 | 0.9900 | 0.9890 | 0.9932 |
| **IV3 - HPO** | 0.9870 | 0.9870 | 0.9870 | 0.9870 | 0.9880 | 0.9897 |
| **MNV3S - HPO** | 0.9888 | 0.9888 | 0.9888 | 0.9888 | 0.9890 | 0.9899 |
| **ENV2B0 - HPO** | 0.9910 | 0.9910 | 0.9910 | 0.9910 | 0.9916 | 0.9920 |
| **ELETL-IDS** | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9996 | 0.9996 |

**TABLE 5. Time-base model Evaluation.**

| CIC-IDS2017 | | | |
|---|---|---|---|
| **Models** | **Train Time (Tt(s))** | **Test Time (tt(s))** | **Test time per packet (tt/p(s))** |
| **Base CNN** | 39772 | 1341 | 0.2837 |
| **VGG16** | 16669 | 1638 | 0.3466 |
| **VGG19** | 10990 | 1759 | 0.3722 |
| **IV3** | **6578** | **1057** | **0.2237** |
| **MNV3S** | **1170** | **857** | **0.1813** |
| **ENV2B0** | **4172** | **930** | **0.1967** |
| **Base CNN - HPO** | 30932 | 1120 | 0.237 |
| **VGG16 - HPO** | 13679 | 1268 | 0.2683 |
| **VGG19 - HPO** | 8980 | 1420 | 0.3005 |
| **IV3 - HPO** | **5272** | **859** | **0.1817** |
| **MNV3S - HPO** | **1025** | **748** | **0.1582** |
| **ENV2B0 - HPO** | **3282** | **856** | **0.1811** |
| **ELETL-IDS** | **36366** | **2897** | **0.6821** |
| CSE-CICIDS2018 | | | |
| **Models** | **Train Time (Tt(s))** | **Test Time (tt(s))** | **Test time per packet (tt/p(s))** |
| **Base CNN** | 43649.54 | 1048 | 0.2468 |
| **VGG16** | 21229.61 | 1499 | 0.3530 |
| **VGG19** | 21316.19 | 1792 | 0.4220 |
| **IV3** | **7848.12** | **890** | **0.2096** |
| **MNV3S** | **975.67** | **838** | **0.1974** |
| **ENV2B0** | **10032.56** | **875** | **0.2483** |
| **Base CNN - HPO** | 36987.56 | 912 | 0.2588 |
| **VGG16 - HPO** | 18964.59 | 988 | 0.2804 |
| **VGG19 - HPO** | 19121.76 | 1021 | 0.2897 |
| **IV3 - HPO** | **6793.55** | **793** | **0.225** |
| **MNV3S - HPO** | **831.23** | **739** | **0.2097** |
| **ENV2B0 - HPO** | **934.22** | **781** | **0.2216** |
| **ELETL-IDS** | **17695** | **1289** | **0.3036** |

CSE-CIC-IDS2018, respectively, for optimized and non-optimized training. On the CIC-IDS2017 dataset, the base CNN architecture achieved an accuracy, recall, precision, and F-score of 0.9976 in each case whereas the AUC and MCC are 0.9980, and 0.9984, respectively. Notably, this was the lowest performance obtained on the dataset compared to the pre-trained architectures. IV3 outperformed all other models, achieving an overall performance in all metrics of 100% in both optimized and non-optimized conditions. Considering the AUC and MCC, the IV3-HPO also performed better than all other models. Again, MobileNetV3Small ranks second in both optimization and non-optimization with an AUC and MCC of almost 100% *(0.9999)*. As stated in III-F, the MCC value is used to validate the AUC value for the models that best show each model's performance.

As with the results for CIC-IDS2017 in Table 3, the performances for each model on the CSE-CICIDS2018 dataset are shown in Table 4. In this case, the base CNN model also achieved the most minor performance in both scenarios, showing that pre-trained models can perform better than normal CNN architecture. This is because pre-trained models have already learned from many datasets and can quickly learn new patterns from new input data. VGG16-HPO and ENV2B0-HPO are the best-performing models on the dataset, reaching an accuracy of 0.9900 and 0.9910, respectively, against the base CNN with 0.9797 accuracy.

With the BS-TPE optimization algorithm used for the Hyper-Parameter search in the search space, we obtain models with better performances. In Tables 3 and 4, the models trained with selected Hyper-Parameters tend to perform better in evaluation metrics and the time cost function.

In constructing the ensemble model, some critical parameters used in selecting the best three models include the training time (Tt(s) (total time taken for the model to be trained on the dataset), test time (tt(s) (i.e the total time taken to predict all the images in the test set), the overall MCC and AUC values, and the F-Score in addition to all the regular evaluation metrics. Therefore, the Tt(s) and tt(s), including test time per packet (tt/p(s)) were measured during the training and evaluation phases. Test time per packet is the average time to detect a data packet. This is equivalent to the total test time divided by the total number of images in the sample. The results obtained are shown in Tables 5. As shown in Table 5, the training time and testing time for the IV3, MNV3S, and ENV2B0 are the lowest, making them best for the CoT environment and therefore were used for the ensemble model development. While IV3-HPO required 5272 seconds and 1057 seconds to train and test on the CIC-IDS2017, it also required 6793.55 seconds and 890 seconds to be trained and tested on the CSE-CIC-IDS2018 dataset, respectively. Similarly, for MNV2S-HPO, 1025 seconds was the training time on CIC-IDS2017 and 831 seconds to train on the CSE-CIC-IDS2018. In both optimization and non-optimization, the IV3, MNV2S, and ENV2B0 had the least time and resource requirements compared to other pre-trained models.

Also, compared with the base CNN trained from scratch, we can observe that more training and testing time is required for optimum performance. This means high computational power is also required; thus transfer learning presents a more convenient approach to training CNN-based IDS models.

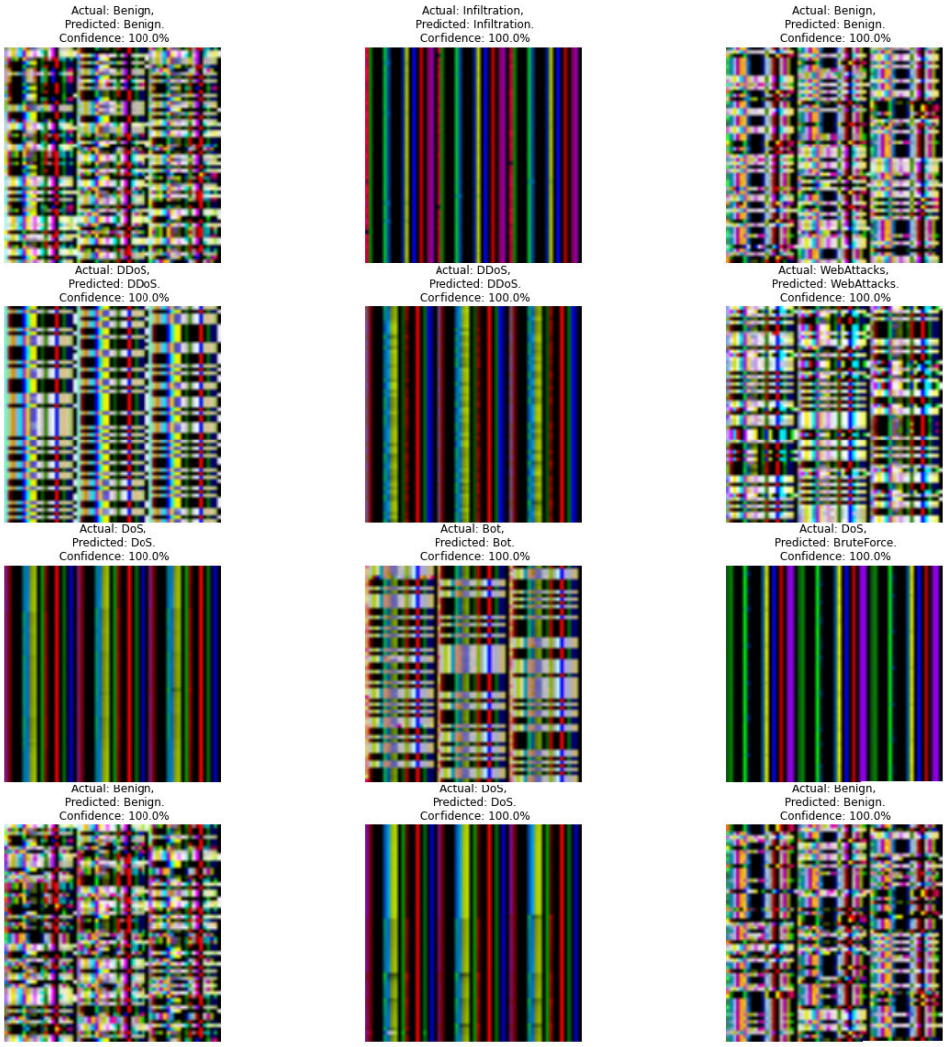
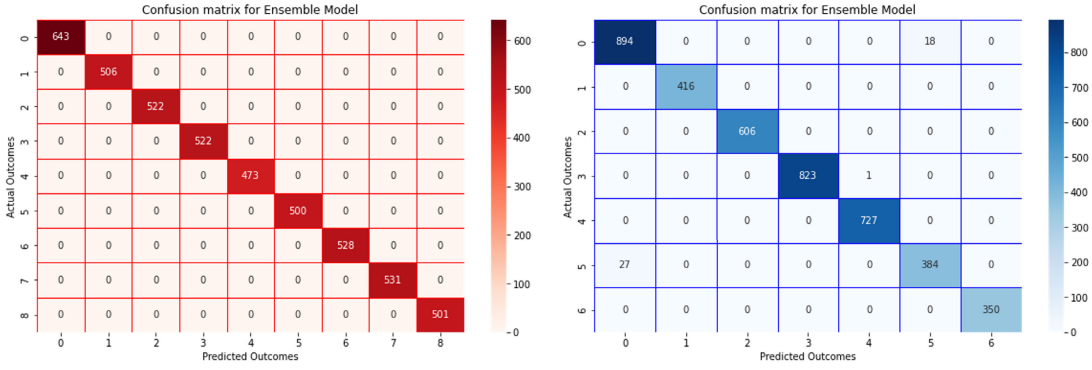


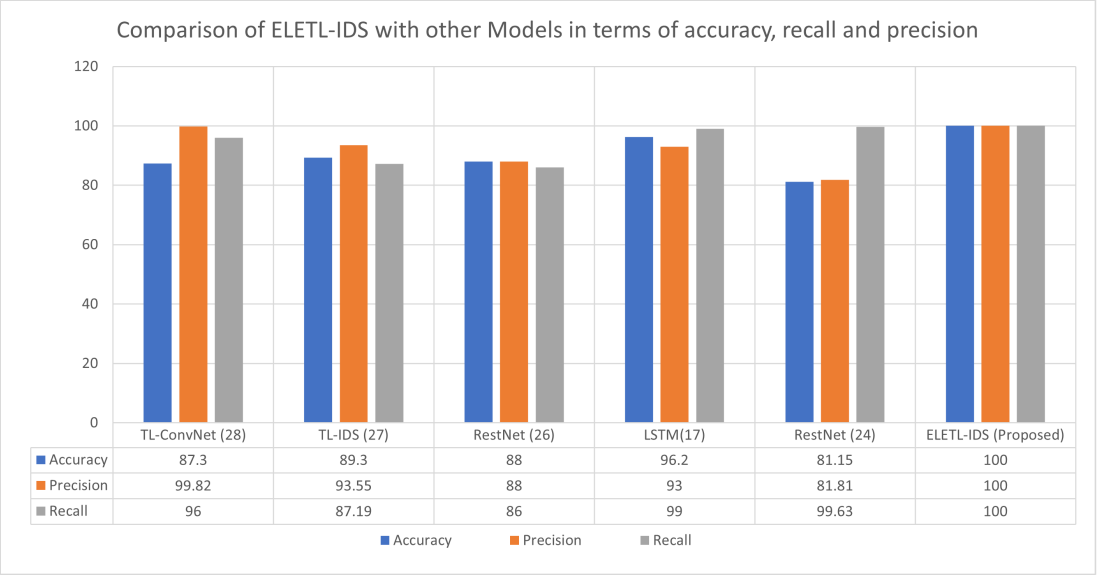

**FIGURE 6.** Prediction Result of the proposed ELETL-IDS model.




a. CIC-IDS2017 b. CSE-CIC-IDS2017

**FIGURE 7.** Confusion Matrix showing the performance of the proposed ELETL-IDS model on selected dataset (a) CIC-IDS2017 and (b) CSE-CIC-IDS2018.

While 43649.54 seconds were needed to train base CNN on the CIC-IDS2017 dataset, only 934.22 seconds were required to train a better-performing model on the same dataset using the ENV2B0-HPO approach. Furthermore, given the test time shown in Table 5, the proposed model is efficient and practically applicable in IoT environments




| | TL-ConvNet (28) | TL-IDS (27) | RestNet (26) | LSTM(17) | RestNet (24) | ELETL-IDS (Proposed) |
|---|---|---|---|---|---|---|
| Accuracy | 87.3 | 89.3 | 88 | 96.2 | 81.15 | 100 |
| Precision | 99.82 | 93.55 | 88 | 93 | 81.81 | 100 |
| Recall | 96 | 87.19 | 86 | 99 | 99.63 | 100 |

**FIGURE 8.** Comparison between our proposed model and other TL-based models.

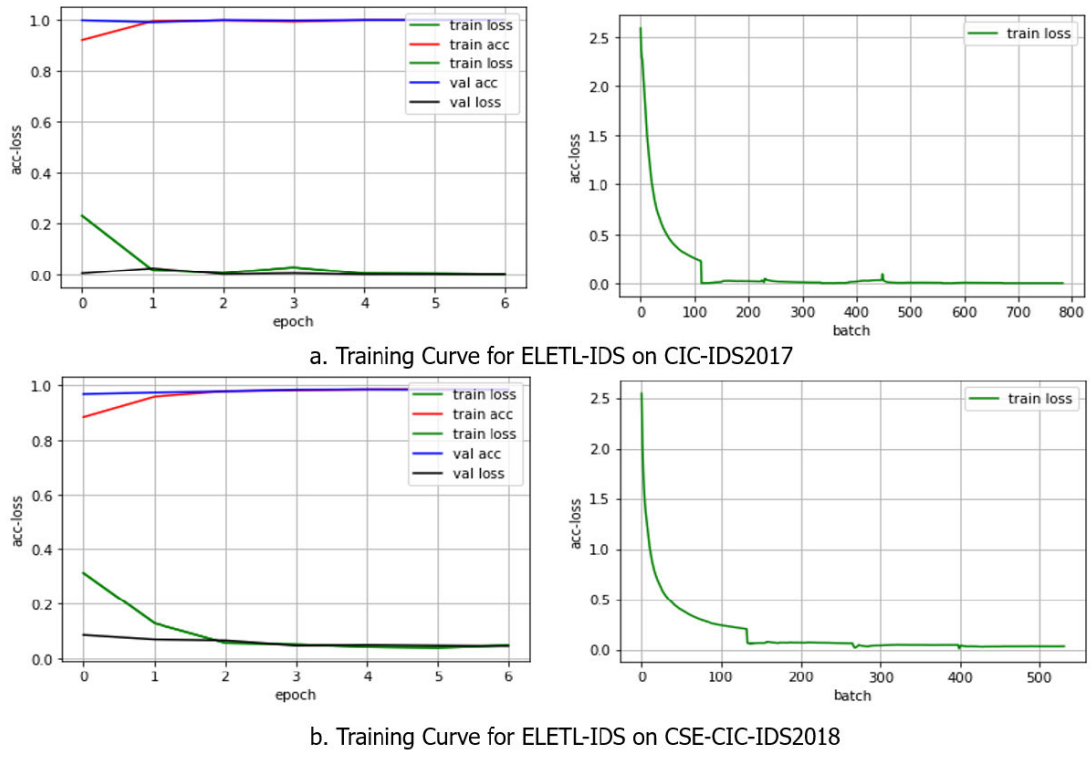



**FIGURE 9.** Learning Curves for the ELETL-IDS Model on the selected Datasets.

because it can detect network flows in a short period of time. For instance, a total of 0.6821 seconds are required to classify a single packet of network flow consisting of 60 features (representing a single image) in the CIC-IDS2017 dataset and 0.3036 seconds in the CSE-CICIDS2018 dataset.

Fig. 6 shows the predicted outcomes together with the confidence level and actual labels. Among the 12 predicted images displayed, our model can classify eleven images accurately, with one wrong prediction showing 0.001 FPR. Thus, FPR, which has been a great challenge in ML/DL tasks, especially concerning IDS, has been reduced to its best minimum.

The confusion matrix helps us understand the models' performance concerning the various tasks being classified by the model. In Fig. 7, we present the confusion matrices for ELETL-IDS obtained on the CIC-IDS2017 and CSE-CIC-IDS2018 datasets. We can observe that ELETL-IDS had no difficulty correctly classifying all the attacks and benign features, reaching an accuracy of 100%. But in classifying the labels of CSE-CIC-IDS2018, the ELETL-IDS identified 18 benign images as infiltration and 27 infiltration attacks as benign, thus, achieving an accuracy of 0.9976. On the confusion matrix are presented the encoded labels for each of the classes of network traffic present in the two datasets used. Accordingly, in the CIC-IDS2017 which contains 9 network profiles, the corresponding label encoding are 0: Benign, 1: Botnet, 2: Brute force, 3: DDoS, 4: DoS, 5: Heartbleed, 6: Infiltration, 7: PortScan, 8: Web attacks. On the other hand, the encoded labels on the CSE-CICIDS2018 dataset are 0: Benign, 1: Bot, 2: DoS, 3: DDoS, 4: Brute force, 5: Infiltration, 6: Web Attacks.

To validate the performance of our proposed model, we make a comparison with other TL-based models as contained in the literature. As shown in Fig. 8, our ELETL-IDS stands out among the other models in accuracy, precision, and recall. We compared our proposed model with the selected related works because they proposed similar methods of detecting network intrusion. Considering that we applied deep learning algorithms to develop our proposed model, we compared the result with more of similar existing deep learning models. Nevertheless, to show that the model is competitive with previous proposals and presents a state of the art functionality, we compared the performance of the proposed model with that of an RNN-based IDS using LSTM [20], [22] and tree-based machine learning algorithms [23]. According to [22], the LSTM-based IDS model with attention mechanism trained on the CSE-CICIDS2018 dataset showed an accuracy of 96.2%. In [20], the proposed model reached an accuracy of 96% while our proposed model achieved a performance accuracy of 100%.

In Fig. 9, we present the learning curves for the proposed model. The learning curve is important in understanding the general behaviour of the model during the training process which in turn aids in evaluating if the model overfits on the training data. In both scenarios, our model during training maintained both training and validation accuracy of almost 100% from the first epoch with 100 batch size using the categorical cross-entropy loss function. This shows that the proposed model is highly efficient and can be deployed for CoT intrusion monitoring.

## V. CONCLUSION

Cloud Internet of Things (CoT) promises to expand the possibilities of application of IoT systems in different domains by leveraging on lower latency requirements and distributed coverage areas. This expansion is not without challenges, as various network attacks tend to exploit the vast network interface and cause harm to the system. To curb the possible damage that successful attacks can cause to the information system and devices, we propose an Intrusion Detection System (IDS) implemented using transfer learning based on the CNN algorithm to detect and prevent network intrusion on CoT devices. First, two primary IDS datasets, CIC-IDS2017 and CSE-CIC-IDS2018, were selected for the research. Since CNN works better on images, the numerical dataset was transformed after cleaning into a set of images measuring 256 X 256 X 3, obtaining 19055 and 16984 images for the CIC-IDS2017 and CSE-CIC-IDS2018 datasets, respectively. Secondly, five different pre-trained models of the CNN architecture, including VGG16, VGG19, InceptinV3, MobileNetV3Small, and EfficientNetV2B0, and one custom CNN algorithm were tested on the image datasets. HPO was carried out using the BO-TPE search algorithm to determine the most crucial Hyper-Parameter for the model fine-tuning. We tuned the pre-trained models using the obtained parameters by freezing and retraining selected network layers to obtain better performances. Ultimately, we developed an ensemble of the three best-performing models with low memory requirements, making it suitable for CoT devices. Results showed that our proposed approach is more efficient, effective, and optimized for edge devices in detecting network intrusions with a very low false positive rate and an accuracy of 100% as shown in Fig. 8.

In addition to this paper, in the future, we intend to explore other optimization algorithms with different pre-trained models to enhance our model's performance especially in identifying the various classes in the CSE-CIC-IDS2018. Evaluating this proposed model on a different network dataset is also a step further in this search.

## REFERENCES

[1] U. O. Matthew, J. S. Kazaure, A. Onyebuchi, O. O. Daniel, I. H. Muhammed, and N. U. Okafor, "Artificial intelligence autonomous unmanned aerial vehicle (UAV) system for remote sensing in security surveillance," in *Proc. IEEE 2nd Int. Conf. Cyberspace (CYBER NIGERIA)*, Feb. 2021, pp. 1–10.

[2] J. Meyer and S. Boll, "Smart health systems for personal health action plans," in *Proc. IEEE 16th Int. Conf. e-Health Netw., Appl. Services (Healthcom)*, Oct. 2014, pp. 404–410.

[3] M. Kasmi, F. Bahloul, and H. Tkitek, "Smart home based on Internet of Things and cloud computing," in *Proc. 7th Int. Conf. Sci. Electron., Technol. Inf. Telecommun. (SETIT)*, Dec. 2016, pp. 82–86.

[4] L. Yushi, J. Fei, and Y. Hui, "Study on application modes of military Internet of Things (MIOT)," in *Proc. IEEE Int. Conf. Comput. Sci. Automat. Eng. (CSAE)*, vol. 3, May 2012, pp. 630–634.

[5] Y. Zixuan, W. Zhifang, and L. Chang, "Research on marine environmental monitoring system based on the Internet of Things technology," in *Proc. IEEE Int. Conf. Electron. Inf. Commun. Technol. (ICEICT)*, Aug. 2016, pp. 121–125.

[6] M. Masoumi, H. R. D. Oskouei, M. M. Shirkolaei, and A. R. Mirtaheri, "Substrate integrated waveguide leaky wave antenna with circular polarization and improvement of the scan angle," *Microw. Opt. Technol. Lett.*, vol. 64, no. 1, pp. 137–141, Jan. 2022.

[7] M. M. Shirkolaei and J. Ghalibafan, "Magnetically scannable slotted waveguide antenna based on the ferrite with gain enhancement," *Waves Random Complex Media*, pp. 1–11, 2021.

[8] M. M. Shirkolaei, H. R. D. Oskouei, and M. Abbasi, "Design of 1∗4 microstrip antenna array on the human thigh with gain enhancement," *IETE J. Res.*, pp. 1–7, 2021, doi: 10.1080/03772063.2021.2004459.

[9] C. Cambra, S. Sendra, J. Lloret, and L. Garcia, "An IoT service-oriented system for agriculture monitoring," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[10] GSMA. *Gsma|the Mobile Economy—The Mobile Economy*. Accessed: Jul. 24, 2022. [Online]. Available: https://www.gsma.com/mobileeconomy/

[11] S. Hakak, S. A. Latif, and G. Amin, "A review on mobile cloud computing and issues in it," *Int. J. Comput. Appl.*, vol. 75, no. 11, pp. 1–4, 2013.

[12] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *Proc. Int. Conf. Future Internet Things Cloud*, Aug. 2014, pp. 464–470.

[13] M. P. K. Shelke, M. S. Sontakke, and A. D. Gawande, "Intrusion detection system for cloud computing," *Int. J. Sci. Technol. Res.*, vol. 1, no. 4, pp. 67–71, 2012.

[14] A. Patel, M. Taghavi, K. Bakhtiyari, and J. C. JúNior, "An intrusion detection and prevention system in cloud computing: A systematic review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 25–41, 2013.

[15] V. Chang, L. Golightly, P. Modesti, Q. A. Xu, L. M. T. Doan, K. Hall, S. Boddu, and A. Kobusińska, "A survey on intrusion detection systems for fog and cloud computing," *Future Internet*, vol. 14, no. 3, p. 89, Mar. 2022.

[16] P. F. De Araujo-Filho, G. Kaddoum, D. R. Campelo, A. G. Santos, D. Macedo, and C. Zanchettin, "Intrusion detection for cyber–physical systems using generative adversarial networks in fog environment," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6247–6256, Apr. 2021.

[17] F. Lin, Y. Zhou, X. An, I. You, and K. Choo, "Fair resource allocation in an intrusion-detection system for edge computing: Ensuring the security of Internet of Things devices," *IEEE Consum. Electron. Mag.*, vol. 7, no. 6, pp. 45–50, Nov. 2018.

[18] L. Nie, Y. Wu, X. Wang, L. Guo, G. Wang, X. Gao, and S. Li, "Intrusion detection for secure social Internet of Things based on collaborative edge computing: A generative adversarial network-based approach," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 1, pp. 134–145, Feb. 2022.

[19] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6882–6897, Aug. 2020.

[20] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, vol. 1, Jan. 2018, pp. 108–116.

[21] A. A. M. Teodoro, D. H. Silva, M. Saadi, O. D. Okey, R. L. Rosa, S. A. Otaibi, and D. Z. Rodríguez, "An analysis of image features extracted by CNNs to design classification models for COVID-19 and non-COVID-19," *J. Signal Process. Syst.*, pp. 1–13, Nov. 2021.

[22] P. Lin, K. Ye, and C.-Z. Xu, "Dynamic network anomaly detection system by using deep learning techniques," in *Proc. Int. Conf. Cloud Comput.* Cham, Switzerland: Springer, 2019, pp. 161–176.

[23] O. D. Okey, S. S. Maidin, P. Adasme, R. L. Rosa, M. Saadi, D. C. Melgarejo, and D. Z. Rodríguez, "BoostedEnML: Efficient technique for detecting cyberattacks in IoT systems using boosted ensemble machine learning," *Sensors*, vol. 22, no. 19, p. 7409, Sep. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/19/7409

[24] Y. Fu, F. Lou, F. Meng, Z. Tian, H. Zhang, and F. Jiang, "An intelligent network attack detection method based on RNN," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace (DSC)*, Jun. 2018, pp. 483–489.

[25] M. Masum and H. Shahriar, "TL-NID: Deep neural network with transfer learning for network intrusion detection," in *Proc. 15th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2020, pp. 1–7.

[26] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[27] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[28] X. Li, Z. Hu, M. Xu, Y. Wang, and J. Ma, "Transfer learning based intrusion detection scheme for Internet of vehicles," *Inf. Sci.*, vol. 547, pp. 119–135, Feb. 2021.

[29] D. Petrov and T. M. Hospedales, "Measuring the transferability of adversarial examples," 2019, *arXiv:1907.06291*.

[30] Z. Taghiyarrenani, A. Fanian, E. Mahdavi, A. Mirzaei, and H. Farsi, "Transfer learning based intrusion detection," in *Proc. 8th Int. Conf. Comput. Knowl. Eng. (ICCKE)*, Oct. 2018, pp. 92–97.

[31] Y. Li, T. Liu, D. Jiang, and T. Meng, "Transfer-learning-based network traffic automatic generation framework," in *Proc. 6th Int. Conf. Intell. Comput. Signal Process. (ICSP)*, Apr. 2021, pp. 851–854.

[32] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2017, pp. 858–866.

[33] H. Dhillon and A. Haque, "Towards network traffic monitoring using deep transfer learning," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Dec. 2020, pp. 1089–1096.

[34] S. T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed, and R. Islam, "Dependable intrusion detection system for IoT: A deep transfer learning based approach," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 1006–1017, Jan. 2023.

[35] P. Wu, H. Guo, and R. Buckland, "A transfer learning approach for network intrusion detection," in *Proc. IEEE 4th Int. Conf. Big Data Anal. (ICBDA)*, Mar. 2019, pp. 281–285.

[36] M. Erfani, F. Shoeleh, S. Dadkhah, B. Kaur, P. Xiong, S. Iqbal, S. Ray, and A. A. Ghorbani, "A feature exploration approach for IoT attack type classification," in *Proc. IEEE Intl Conf Dependable, Autonomic Secure Comput., Intl Conf Pervasive Intell. Comput., Intl Conf Cloud Big Data Comput., Intl Conf Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Oct. 2021, pp. 582–588.

[37] *IDS 2017|Datasets|Research|Canadian Institute for Cybersecurity|Unb*. Accessed: Jul. 23, 2022. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[38] *Index of/Cicdataset/Cic-IDS-2017/Dataset*. Accessed: Sep. 8, 2022. [Online]. Available: http://205.174.165.80/CICDataset/CIC-IDS-2017/Dataset/

[39] *A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)—Registry of Open Data on AWS*. Accessed: Sep. 8, 2022. [Online]. Available: https://registry.opendata.aws/cse-cic-ids2018/

[40] *Applications|Research|Canadian Institute for Cybersecurity|UNB*. Accessed: Jul. 23, 2022. [Online]. Available: https://www.unb.ca/cic/research/applications.html

[41] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Dec. 2002.

[42] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intell. Comput.* Cham, Switzerland: Springer, 2005, pp. 878–887.

[43] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "IoT DoS and DDoS attack detection using ResNet," in *Proc. IEEE 23rd Int. Multitopic Conf. (INMIC)*, Nov. 2020, pp. 1–6.

[44] S.-F. Lokman, A. T. Othman, M. H. A. Bakar, and S. Musa, "The impact of different feature scaling methods on intrusion detection for in-vehicle controller area network (CAN)," in *Proc. Int. Conf. Adv. Cyber Secur.* Penang, Malaysia: Springer, 2019, pp. 195–205.

[45] Wikipedia. *Quantile Function—Wikipedia*. Accessed: Jul. 23, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Quantile-function

[46] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *Proc. ECML PKDD Workshop, Lang. Data Mining Mach. Learn.*, 2013, pp. 108–122.

[47] K. Weiss, T. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, pp. 1–40, May 2016.

[48] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Scholkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, Jul. 2006.

[49] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from AlexNet: A comprehensive survey on deep learning approaches," 2018, *arXiv:1803.01164*.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[52] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[53] M. Ttan and Q. Le, "EfficientNetV2: Smaller models and faster training," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10096–10106.

[54] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[55] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.
[56] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[57] L. Yang, A. Moubayed, and A. Shami, "MTH-IDS: A multitiered hybrid intrusion detection system for Internet of vehicles," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 616–632, Jan. 2022.
[58] S. Das, S. Saha, A. T. Priyoti, E. K. Roy, F. T. Sheldon, A. Haque, and S. Shiva, "Network intrusion detection and comparative analysis using ensemble machine learning and feature selection," *IEEE Trans. Netw. Service Manage.*, early access, Dec. 27, 2021, doi: 10.1109/TNSM.2021.3138457.
[59] J. Large, J. Lines, and A. Bagnall, "A probabilistic classifier ensemble weighting scheme based on cross-validated accuracy estimates," *Data Mining Knowl. Discovery*, vol. 33, no. 6, pp. 1674–1709, Nov. 2019.
[60] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020.
[61] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 115–123.
[62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Jan. 2011.
[63] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Dec. 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786550500303X
[64] *A Comparison of MCC and CEN Error Measures in Multi-Class Prediction|PLoS ONE*. Accessed: Jul. 25, 2022. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0041882
[65] *Phi Coefficient—Wikipedia*. Accessed: Jul. 25, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Phi_coefficient

**OGOBUCHI DANIEL OKEY** (Student Member, IEEE) received the bachelor's degree in computer engineering from the Michael Okpara University of Agriculture, Umudike, Abia, Nigeria, in 2017, and the master's degree in systems engineering and automation from the Federal University of Lavras, Brazil, in 2022. He is currently pursuing the Ph.D. degree with the Graduate Program of Information Engineering, Federal University of ABC, Santo André, São Paulo. His current research interests include computer systems, machine learning, quantum computing, cybersecurity, and systems security and automation.

**DICK CARRILLO MELGAREJO** (Member, IEEE) received the B.Eng. degree (Hons.) in electronics and electrical engineering from the National University of San Marcos, Lima, Peru, in 2004, and the M.Sc. degree in electrical engineering from the Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil, in 2008. He is currently pursuing the Ph.D. degree in electrical engineering with the Lappeenranta–Lahti University of Technology. From 2008 to 2010, he contributed to WIMAX (IEEE 802.16m) Standardization. Since 2022, he has been a Senior Standardization Specialist with Nokia Bell Labs, where he is contributing on shaping the 3GPP release 18 standard (5G-Advanced). His research interests include mobile technologies beyond 5G, energy harvesting, intelligent metasurfaces, cell-free mMIMO, and RAN slicing.

**MUHAMMAD SAADI** received the B.Sc. degree from the National University of Computer and Emerging Sciences, Pakistan, in 2007, the M.Sc. degree from the National University of Malaysia, Malaysia, in 2009, and the Ph.D. degree in electrical engineering from Chulalongkorn University, Bangkok, Thailand. He was at the National Electronics and Computer Technology Center, Aimagin Ltd., Thailand. He is currently an Associate Professor with the Department of Electrical Engineering, University of Central Punjab, Pakistan. His current research interests include visible light communication, indoor localization, machine learning, deep learning, and next-generation networks.

**RENATA LOPES ROSA** received the M.S. degree from the University of São Paulo, in 2009, and the Ph.D. degree from the Polytechnic School of the University of São Paulo (EPUSP), in 2015. She is currently an Adjunct Professor with the Department of Computer Science, Federal University of Lavras, Brazil. She has a solid knowledge in computer science based on more than ten years of professional experience. Her current research interests include computer networks, telecommunication systems, machine learning, quality of experience of multimedia service, cybersecurity, social networks, and recommendation systems.

**JOÃO HENRIQUE KLEINSCHMIDT** (Member, IEEE) received the B.S. degree in computer engineering and the master's degree in computer science from the Pontifical Catholic University of Paraná, in 2001 and 2004, respectively, and the Ph.D. degree in electrical engineering from the State University of Campinas, in 2008. He is currently an Associate Professor with the Federal University of ABC, Santo André, São Paulo, Brazil. His research interests include computer networks, wireless communications, the Internet of Things, blockchain, and information security.

**DEMÓSTENES ZEGARRA RODRÍGUEZ** (Senior Member, IEEE) received the B.S. degree in electronic engineering from the Pontifical Catholic University of Peru, Peru, and the M.S. and Ph.D. degrees from the University of São Paulo, in 2009 and 2013, respectively. From 2018 to 2019, he had a postdoctoral position at the Technical University of Berlin, specifically at the Quality and Usability Laboratory. He is currently an Adjunct Professor with the Department of Computer Science, Federal University of Lavras, Brazil. He has a solid knowledge in telecommunication systems and computer science based on 15 years of professional experience in major companies. His research interests include QoS and QoE in multimedia services, architect solutions in telecommunication systems, intrusion detection systems, and cybersecurity. He is a member of the Brazilian Telecommunications Society.