A Deep Transfer Learning Approach to Enhance Network Intrusion Detection Capabilities for Cyber Security

Abhijit Das
Research Scholar, VTU-Belagavi, CSE, PESITM, Shivamogga
Asst. Prof., Dept. of CSE, BNMIT, Bangalore, India

Pramod
Associate Professor, Dept. of ISE
PES Institute of Technology & Management
Affiliated to VTU, Shivamogga, India

Abstract—Cyberattacks are on the rise, making technology companies increasingly prone to data theft. Recent research has focused on constructing cognitive models for traffic anomaly detection in a communication network. Many of these experiments resulted in data packets recorded by technologies like Wireshark. These datasets provide high-dimensional data relating to benign and malicious data packets. Recent research has mostly focused on developing machine learning, and deep learning systems to detect attack data packets in a network. Also, machine learning algorithms are currently trained to detect only known threats. However, with the growth of new cyberattacks and zero-day attacks, current algorithms are unable to detect unknown attacks. This research focuses on detecting rare attacks using transfer learning from a dataset of known attacks. Deep learning outperforms explicit statistical modelling approaches by at least 21% for the dataset used. A preliminary survey of candidate deep learning architectures has been performed before testing for transferability and proposes a Convolutional Neural Network architecture that is 99.65% accurate in classifying attack data packets. The suggested CNN architecture trained with a known attack and then tested its performance on unknown attacks to assess transferability. For this model to extract sufficient information for transferability, the training samples must have more information. Only 20% of the dataset represents recent threat data. Several strategies, such as innovative synthetic dataset-based training and bootstrapped dataset training, have been developed to overcome small training sets. A subset of training attacks is determined to optimise learning potential. This study finds training-testing attack pairings with good learning transferability. The most robust and stable relationships are found in DoS attack training-testing pairings. This study also presents model generalisation hypotheses. The dataset features and attack characteristics were analysed using the Recursive Feature Elimination (RFE) algorithm to validate the results.

Keywords—Transfer learning; convolutional neural network; intrusion detection system; cyber security; machine learning and deep learning

I. INTRODUCTION

A neural network uses knowledge learned from previous training to improve generalization on another similar task in transfer learning. The primary goal of this study is to leverage learning transferability to select attacks from the training dataset that causes the model to generalize for other attacks, which can be extended to rare attacks.

In computer networks that employ machine learning algorithms for network security, it is difficult to provide guarantees about the kind of attacks that the network can expect to see and is susceptible to, especially with the rise in the number of novel attacks. Current machine learning algorithms are usually trained to detect a set of known attacks or learn from attacks performed in the past. Therefore, it is hard to predict the range of attacks against which the employed machine learning algorithm is robust. In the literature, there is proof of concept for both classical ML and DL approaches for anomaly detection [1].

Transferability studies can help us understand the range of attacks against which the system is actually robust based on the attacks that the algorithm has been trained to detect and how those attacks enable the model to generalize for other attacks. This is especially useful when the trained model is able to generalize to unknown attacks without being trained on them explicitly because this provides a more extensive range of attacks that the network may be secure against attacks. Studying these attack correlations indicates the attacks that the model can detect and the accuracy of these unknown attacks. This helps us evaluate the risks of what the model can predict if the observed correlations are consistent and what kind of protection the algorithm provides at what computational cost. Transferability studies could discover that the model scales to other different attacks it has learned to detect without being trained on them explicitly. It assures detection for an even broader range of attacks than the ones it has seen. For groups of similar attacks or correlated attacks, transfer learning enables us to find representative attacks from that group. This helps us to train the model with a smaller number of attacks yet achieve the same level of security as it would have if trained with all the attacks in the group.

The increasing machine learning applications have also seen deep learning models going into hardware chips. The advantage of transferability is that we could identify smaller training sets yet make the model generalize well for a broader range of attacks. This decreases training times, making the process of training computationally efficient and having lower memory requirements. This is important, especially if this algorithm is deployed on a resource-constrained device while performing at par with a model trained on a full training set.

Increasing research in this domain has led to an increase in network traffic datasets. Some of the common datasets used are CAIDA 2007, DARPA 98, KDD 99, CSE-CIC-IDS 2018, and UNSW-NB15, to name a few. This work uses the traffic
dataset CICIDS 2017 for training the proposed model. Traffic monitoring software like Wireshark has been used for logging and monitoring network traffic packets for creating these datasets. While these datasets provide a complete description of network packets, they have a high dimensional feature space. In a preliminary study of a potentially suitable neural network, a model has been developed to extract meaningful data from high dimensional training data and a limited number of attack data packets. After identifying a suitable network, the work focuses on attack correlations and comparisons for why training the model with one attack scales for another and if these correlations are symmetric or asymmetric. This study describes the work’s contributions in three sections: first, Developing a DNN architecture; second, Testing the proposed architecture for transferability and third, hypothesis for attack correlations.

Developing a DNN architecture: A comparative study of a Hidden Markov Model has been carried out as a statistical modelling-based approach and a candidate Convolutional LSTM Deep Neural Network (CLDNN) architecture and shows that the deep learning approach scales better for the used dataset. Further, deep learning models have been explored for the classification task. The proposed model’s candidate DNN architectures include a Convolutional LSTM Deep Neural Network (CLDNN), Convolutional Neural Network with BGRU recurrent layers, and One Class Neural Networks (OCNN), and a Convolutional Neural Network (CNN). The performance of these models has been evaluated when trained and tested on all attacks in the dataset. Also, the learning transferability has been evaluated. All the above architectures had overall accuracies of above 90%; however, this dataset is highly unbalanced, with a majority of the dataset consisting of benign data packets. The models are able to detect benign attack packets with high accuracy, thereby boosting the overall accuracy of the models. Therefore, a major criterion for architecture selection is not the overall accuracy but the percentage of attack data packets it is able to classify correctly. Attack transferability has been studied with some of these architectures; however, none of the candidate models other than the proposed architecture showed very strong correlations between training-testing attack pairs. The proposed architecture is the model that shows maximum attack data classification accuracy for the multi-class classification problem. This architecture is able to perform deep feature extraction using flow-based features not only for adequately sampled attacks but also for severely underrepresented attacks.

Testing the proposed architecture for transferability: Attack transferability has been tested with the proposed model architecture by training the model with one kind of attack from the training dataset and testing it on another attack. The entire dataset consists of 20% of attack data that are further divided into fourteen categories of attacks. Therefore, the representation of each attack class is low. For learning transferability, the model requires a larger number of attack data packets. Two techniques have been used to increase the number of attack data packets to address this problem: 1) Using SMOTE generated data and 2) Using a bootstrapped dataset. A Synthetic Minority Oversampling Technique (SMOTE) was used for the first method to generate more attack data packets in each attack class synthetically. For the second method, the original attack data is resampled, shuffled and added to the training dataset to match the number of benign data packets. Furthermore, exploring the possibility that the model may exhibit higher testing accuracies for a particular attack if it is trained with a subset of other attacks in the dataset. An attack boosting algorithm is employed to select a subset of attacks from the training dataset to maximize the classification accuracy when tested on a particular attack. Hypotheses for correlated attacks: The study of attack correlations reveals training tested attack pairs that exhibit strong correlations with each other. A hypothesis has been proposed for these observed attack correlations. The attack characteristics have been studied and important features to validate the hypotheses. The Recursive Feature Elimination (RFE) algorithm is used to identify the dominant features for learning. This not only reduces the dimensionality of the training data but also reduces training times. This study makes inferences about the properties of the features selected and the number of features selected for a pair of correlated attacks. This results in identifying usable attacks in this dataset for transfer learning and explaining why they scale for certain attacks.

II. BACKGROUND AND RELATED WORK

The dataset used is the Canadian Institute of Cybersecurity Intrusion Detection Systems dataset: CICIDS 2017. Overall, the dataset consists of 8 separate CSV files, with data corresponding to attacks simulated in 8 sessions included. The entire dataset consists of data corresponding to 14 types of attacks and benign traffic. The dataset describes 78 flow features. 80% of the data entries are benign data, and 20% of the dataset consists of attack data. The dataset was created by simulating 14 different types of attacks that are shown below in Table I.

| Classes | Attack | Percentage of data |
|---------|--------|--------------------|
| 0       | BENIGN | 80.3               |
| 1       | Botnet | 0.009              |
| 2       | DDoS   | 0.52               |
| 3       | DoS GoldenEye | 0.36 |
| 4       | DoS Hulk | 8.16 |
| 5       | DoS SlowHTTP | 0.19 |
| 6       | DoS Slowloris | 0.2 |
| 7       | FTP-Patator | 0.28 |
| 8       | Heartbleed | 0.00038 |
| 9       | Infiltration | 0.00012 |
| 10      | PortScan | 5.61 |
| 11      | SSH-Patator | 0.2 |
| 12      | Web Attack Brute Force | 0.053 |
| 13      | Web Attack SQL Injection | 0.00074 |
| 14      | Web Attack XSS | 0.023 |

An attacker gains control of several machines and servers connected to the internet and uses these servers to carry out cyberattacks against a target. Due to multiple synchronized attacking machines, the attack data volumes are huge, thereby facilitating a strong attack against the target machine. Botnets are often used to carry out DDoS attacks. Denial of Service (DoS) attacks are attacks aimed at making a server unresponsive to the requests of legitimate users. They usually work by sending enormous traffic, thereby flooding the server and hitting its resource pool by requesting resources or sending obfuscated data that make the server unavailable to real requests.

Distributed Denial of Service (DDoS) attacks are DoS attacks conducted by different servers at once on a victim server. These servers have different IP addresses, thus making...
it difficult to track down one single IP address for attacker identification, which makes it hard to mitigate the attack. In the case of flooding attacks, the volume of data generated to flood the victim server is larger, making the DoS attack strong and harder to terminate[2]. DoS: A goldenEye is an attack tool that identifies vulnerabilities in a target server. It exploits the capability of victim servers to form multiple HTTP connections, thereby using up all possible connections on that server. This attack can be operated as a distributed attack as well. DoS GoldenEye uses Keep-Alive headers and Caching Control options to keep the connection alive, preventing the target server from shutting down the connections and making the server unresponsive to non-attack requests.

**DoS: HULK** is HTTP Unbearable Load King attack. A flooding attack floods the target servers with a large volume of HTTP requests, requesting data or resources or sending unclear HTTP packets. This floods the target machine with HTTP data packets, and the load is unbearable for the server, which makes the server unreachable.

DoS: Slowhttptest and DoS: Slowloris are attack tools for carrying out Slow HTTP attacks. **Slow HTTP attacks** form multiple connections with the target server and try to keep the connections open. Some slow HTTP attacks keep the connection open by declaring a large amount of data to be sent and sending the data at very large intervals of time, almost equal to the timeout period. However, the server cannot close the connection, and the connection does not take time out. This keeps all available connections alive and keeps the server from responding to legitimate requests. Slow HTTP attacks can be made using Slow Header attacks, where the packet header arrives at large intervals or Slow Body attacks, where the body of the data arrives very slowly.

**Brute Force attacks** are targeted towards gaining access to authentication keys by trying all possible combinations of the key. File Transfer Protocol is a network protocol that can be used to transfer files. Users connect to the server using an FTP client using the username and password authentication. Brute Force FTP attacks are an attack on the username and password. Brute Force SSH gains access to valid login credentials to authenticate SSH access to a server by trying all possible combinations.

**Heartbleed** could be classified as a protocol based attack where the attack utilizes a packet header field required for the transport layer security protocol that causes a machine to dump out its entire memory, including confidential and protected data. It does so without leaving traces in the target machine, making it undetectable. In an infiltration attack, the attacker gains access to a protected network or system and finds vulnerabilities in the machines or devices connected to the network. After identifying vulnerabilities, the attacker attacks the machine or device to steal private information.

In an **Port Scan attacks**, the attacker scans the ports of the target server and sends requests to a range of ports on the target server. The attacker finds an active port on the server and exploits a known vulnerability of that service to attack the target server.

Web Attacks identify weaknesses in web applications to gain access to private or protected data. These weaknesses could be exploited using data entry or using injection attacks where malicious scripts are injected into data entries of otherwise harmless websites or by using brute force. For example, for a web application that uses an underlying SQL database and has an option for user inputs, malicious inputs could be given that cause the database to dump out confidential information. DDoS attacks are also a kind of web attack.

It can be observed from Table 1 the CICIDS 2017 dataset is highly unbalanced, with 80% of the dataset consisting of benign data and 20% of the dataset consisting of attack data packets. The attack data is further divided unequally into fourteen different types of attacks. Because of the unequal division, certain attack classes have a very sparse representation, for example, just 11 or 36 data packets throughout the entire dataset. This makes it difficult to generate realistic synthetic data to augment the number of data packets and limits model learning even when using a bootstrapped dataset for these classes of attacks. Therefore, when the proposed architecture has been tested for attack transferability, It eliminates those classes of attacks that cannot use for the transferability of learning.

In the literature, there are different machine learning techniques that deal with intrusion detection using flow characteristics. Alkasassbeh et al. explore MLP, Naive-Bayes, and Random Forest classification algorithms for Distributed Denial of Service attack detection and show that MLP achieves the highest accuracy [3]. Lopez et al. presents a study of machine learning techniques for traffic anomaly detection and proves that a Random Forest-based decision classifier is the best model for anomaly detection, and a Dense Neural Network is a good classifier for some types of DDoS attacks with methods to boost a number of attack samples of underrepresented attack types [1], Vinayakumar et al. carry out a comprehensive study of DNNs and Machine Learning classifiers that learn abstract and high dimensional data representation using the KDDCup99 dataset and test their model performance on other datasets, such as NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CI-CIDS 2017. They also propose a hybrid DNN framework which can be deployed in real-time to monitor network traffic and events to detect possible network attacks [4]. Sharafaldin et al. generate a dataset consisting of benign and seven types of attack data. They also evaluate the performance of machine learning algorithms to identify the best subset of features for certain types of attacks [5]. Ferrag et al. analyze RNNs, DNNs, restricted Boltzmann machines, Deep Belief Networks, CNNs, Deep Boltzmann machines and deep autoencoders for traffic data classification using the CSE-CIC-IDS2018 dataset and the BotIoT' dataset [6].

To find out about an undiscovered attack, researchers have chosen TL-based IDS more, Zhao et al. [7] proposed an algorithm that used the NSL-KDD dataset for binary classification mapped the source and target domains into the latent space K. In their method, the R2L attack was found by looking at DoS data from the source domain. The same author made their TL even better in their next work by adding clusters [8]. They found that 500 data points are enough to train the NSL-KDD dataset well. It was shown by Taghiyarenani et al. [9] that transfer learning can work even if the source and target data are different from each other. They were able to tell the difference between normal traffic and traffic that wasn’t normal. Their results were better when the source domain had a lot of labels.
and the target domain didn’t have as many labels.

Wu et al. [10] also used a CNN to pass on knowledge learned from one dataset to another. When they researched, the UNSW-NB15 dataset was used as the source data, and the NSL-KDD dataset was considered the target data. As the datasets were different, they used two CNNs in their method. Afterwards, a fully connected layer used the data to make the final classification for the data from NSL-KDD. Singla et al. [11] found that transfer learning can help when there isn’t enough training data to learn new attacks. Their approach isn’t very different from what [10] had done. They used binary classification to look for a specific attack in the UNSW-NB15 dataset. They used the remainder of the attacks using the same dataset as the original domain. With the normal DNN, they found that it did better when there was less training data for their transfer learning-based approach.

In 2020, Masum et al. [12] used the same computer vision algorithms on IDS that they used on PCs. As image processing needs two-dimensional data, they turned the intrusions dataset into two-dimensional data, and then used VGG-16 [13] to look for intrusions. However, they were able to get about 95% accuracy, which some people might think isn’t enough for the real world because other algorithms have been able to get more than 95% accuracy in the past. The reason could be that the features that were made with VGG-16 might be better for image processing than intrusion detection. Second, Dhillon et al. [14] have shown how TL can be used in real-time. A CNN-LSTM algorithm was made. Then, they moved their model to a new place where they could use data that wasn’t there to figure out whether it was an attack or not. Research done by them shows that there is no difference between the space for features and the space for labels in both fields. They were able to speed up their research by using TL.

Vijayakumar, Alazab et al. [15] evaluated the performance of CNN for intrusion detection. CNN was investigated in their work both directly and in combination with RNN, LSTM, and GRU algorithms. CNN and recurrent algorithms were found to perform nearly identically when used together. Even though they could get good outcomes, the computational power required by their approach was excessive.

III. METHODOLOGIES

A. Hidden Markov Models: A Statistical Modeling based Approach

HMM is a Markov Model in which the process being modelled is a Markov Process. Markov models are used to model processes that change stochastically. This study considers that the phenomenon being modelled is a first-order Markov Process where the next state of the phenomenon is only dependent on the current state of the phenomenon. This is called the Markovian Property. If \( q_t \) represents the state of a model at time instant \( t \), then the conditional probability of its next state will be as given:

\[
P(q_{t+1} = S_i | q_t = S_j, q_t - 2 = S_k, ..., q_t - 1 = S_j) = P(q_{t+1} = S_i | q_t = S_j)
\]

Where \( S_i \) is the current state and \( S_j, S_k \) represents the past states.

In HMM problem, the observations are the features corresponding to a data sample. Forty flow features from the dataset are taken for each data sample and are fed to each HMM as 40 observations. Each CSV file in the dataset is treated as a separate dataset. The performance of HMMs has been evaluated on 2 class and 4 class classification problems. In those cases, 2 HMMs and 4 HMMs have been used, respectively, for training. Observations are taken at a time from the testing dataset, and each model is tested with the T observations. The model that generates the highest probability \( P(O|\lambda) \) implies that the set of observations \( O \) belongs to that HMM and class.

Computation of \( P(O|\lambda) \) using hidden states. Let \( Q \) be the set of hidden states:

\[
O = O_1, O_2, O_3, ..., O_T \text{ where } T = 40.
\]

\[
P(O|\lambda) = P(O_1, O_2, O_3, ..., O_T | \lambda) = \sum_Q P(O|Q, \lambda)P(Q|\lambda)
\]

Where \( Q \) is the set of all possible hidden states, the complexity is of the order \( O(2TN^2) \) for a general system with \( N \) states and \( T \) observations to compute \( P(O|\lambda) \) using the method above. A forward-backwards algorithm is used to compute the same quantity. This reduces the computational complexity to \( O(N^2T) \).

B. Neural Network Architectures

Deep Neural Network architectures have been tested to identify the best architecture to use to study attack correlations. The first two architectures that have been tested are a CLDNN and CNN with BGRU layers. These neural networks are part of a class of neural networks called CRNN, which is CNN with Recurrent Layers. Convolutional layers have been used because they capture spatial features from the dataset and gated RNNs to capture temporal patterns in the sequence. Bidirectional GRUs capture temporal patterns not only forward in time but also backward in time. In addition, the performance of a One Class Neural Network (OCNN) has been evaluated as it is good for minority data detection in large datasets. Finally, the performance of a Convolutional Neural Network without Recurrent layers has been evaluated.

1) CLDNN: The Convolutional LSTM Deep Neural Network architecture has 6 hidden layers. The first layer is a convolutional layer with 256 feature maps, a (1, 3) convolution kernel and 20% dropout. The second hidden layer is a convolutional layer with 256 feature maps and a (2, 3) convolution kernel. The third layer is a convolutional layer with 80 feature maps and a (1, 3) kernel with 20% dropout. The fourth hidden layer is a convolutional layer with 80 feature maps and a (1, 3) kernel. The fifth hidden layer is an LSTM layer with 50 cells. The sixth hidden layer is a fully-connected layer with 128 neurons. The CLDNN architecture has shown in Table II. All hidden layers have Rectified Linear Unit (ReLU) activation as:

\[
g(x) = \max(0, x)
\]

This architecture was used to evaluate its performance on i) The multi-class classification problem when it is trained with all classes of data and tested on all classes of data and ii) The two-class classification problem when it is trained with...
one class of data and tested on all other classes of data. For the multi-class classification problem, the output layer has 15 output classes. For the two-class classification problem, the output layer has two output classes. In both implementations, the output layer has softmax activation. Softmax activation can be given as:

\[ \sigma_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{J} e^{z_j}}, \text{ where } J \text{ is the total number of classes} \]

All 78 features are used for training. For the two-class classification problem, the model has been trained on one kind of attack and evaluates its performance when tested on all the other attacks. In this case, all benign data is labelled zero, and all attack data is labelled one. For the multi-class classification problem, the model is trained with all 15 classes of data. The total dataset is divided into a fifty-fifty split; 50% of the data is used for training, and 50% of the data is used for testing. The loss function used is Categorical CrossEntropy, and the optimizer is Adam. The batch size is 1024. This model is trained for 50 epochs. The categorical CrossEntropy loss function is given by: \( L_{CE} = -\frac{1}{N} \sum_{i=1}^{N} t_i \log(p_i) \)

where \( t_i \) is the true label of the \( i^{th} \) data point, \( p_i \) is the predicted probability of the data point belonging to class \( t_i \) and \( N \) is the batch size.

2) **CNN with BGRU layers and CNN with stacked BGRU layers**: Gated Recurrent Units are different from traditional RNNs without gates as they are able to control the amount of previous data to carry over to the next iteration using hidden states. GRUs usually have a reset gate and update gate as opposed to a forget gate, input gate and output gate as in LSTMs. This makes training the model with GRUs faster than using LSTMs. BGRUs have connections that go backwards in time, enabling them to capture temporal patterns backwards and forward in time. The CNN with Bidirectional Gated Recurrent Units and Stacked Bidirectional Gated Recurrent Units architecture is the same as the architecture of the CLDNN used, except in these architectures, there are BGRU and stacked-BGRU layers in place of the LSTM layer. Each BGRU layer has 50 cells. For a stacked-BGRU approach, there are two BGRU layers consecutively with 50 cells in each layer. Each model is trained for 150 epochs, with a batch size of 1024. The loss function used is mean squared error, and the optimizer used is RMSProp. Table III and Table IV are the architectures of these two DNNs.

3) **One Class Neural Networks**: One Class Neural Networks (OCNN) are used for anomaly detection in complex datasets that require highly non-linear decision boundaries. The loss function of OCNNs is derived from the loss function of OC-SVMs. An OC-SVM, for input data \( X \) with \( N \) input samples \( \{X_i\}_{i=1}^{N} \). \( \phi(X_n) \) is a reproducing kernel Hilbert space that is a mapping from the input space to feature space. All the input data points are labelled one, and the only negative point is the origin. A hyperplane separates the origin from the mapped \( \phi(X_n) \)’s. The hyperplane in the feature space is given by \( f(X_n) = w^T \phi(X_n) - r \). \( v \) is the parameter to control the trade-off between maximizing the distance and the number of data points falsely classified as positive. \( r \) is the hyperplane bias. An OCNN has a feedforward neural network with one hidden layer and one output node. Here, \( w \) is the scalar output from the hidden to the output layer, and \( V \) is the weight matrix from the input to the hidden layer. The hidden layer has a linear or sigmoidal activation given by \( g(\cdot) \).

The optimization problem for OCNNs is:

\[
\min_{w, V} \frac{1}{2} || w ||^2 + \frac{1}{2} || V ||^2 + \frac{1}{N} \sum_{i=1}^{N} \max(0, -\langle w, g(V(X_i)) \rangle - r)
\]

(1)

\( (w, V) \) are updated using normal backpropagation. The model is trained using important features extracted from an autoencoder instead of using the raw data. The OCNN algorithm is implemented by Chalapathy et al. [16]. The performance metrics have been calculated by True Positive Rate (TPR), False Positive Rate (FPR) ROC curve.

4) **CNN**: The Convolutional Deep Neural Network architecture consists 5 hidden layers. The first hidden layer is a Convolutional layer with 256 feature maps and a (1, 3) convolution kernel. The second hidden layer is a convolutional layer with 256 feature maps and a (2, 3) convolution kernel. The third hidden layer is a convolutional layer with 256 feature maps and a (1, 3) convolution kernel and a dropout of 20%. The fourth hidden layer is a convolutional layer with 80 feature maps and a (1, 3) convolution kernel. The fifth hidden layer is a fully-connected layer with 128 neurons. The output layer has 2 output nodes and sigmoid activation. Layers 1, 4 and 5 have ReLU activation and layers 2 and 3 have Leaky ReLU activation with \( \alpha = 0.3 \). All 78 features from the dataset were used for training. This model’s performance is evaluated on the multi-class classification problem. The architecture of CNN has shown in Table V. The model is trained on 70% of data and tested on 30% of the data. 10% of the training data is used as validation data. The model is trained for 100 epochs. The

| TABLE II. CLDNN ARCHITECTURE |
|-----------------------------|
| **CLDNN Architecture**      |
| Conv2D 1x3x256, ReLU, Dropout(rate=0.2) |
| Conv2D 1x3x80, ReLU, Dropout(rate=0.2) |
| Conv2D 2x3x256, ReLU |
| Conv2D 1x3x64, ReLU |
| LSTM 50 |
| Dense, 128, ReLU |
| Dense, 2, Softmax |
| **TABLE III. CNN WITH BGRU ARCHITECTURE** |
| **CNN with BGRU Architecture** |
| Conv2D 1x3x256, ReLU, Dropout(rate=0.2) |
| Conv2D 1x3x256, ReLU |
| Conv2D 1x3x80, ReLU, Dropout(rate=0.2) |
| Conv2D 1x3x64, ReLU |
| BGRU 50 |
| Dense, 128, ReLU |
| Dense, 2, Softmax |
| **TABLE IV. CNN WITH STACKED BGRU ARCHITECTURE** |
| **CNN with Stacked BGRU Architecture** |
| Conv2D 1x3x256, ReLU, Dropout(rate=0.2) |
| Conv2D 1x3x80, ReLU, Dropout(rate=0.2) |
| Conv2D 1x3x64, ReLU |
| BGRU 50 |
| Dense, 128, ReLU |
| Dense, 2, Softmax |
The activation function used is Leaky ReLU and Sigmoid. The loss function used is Binary Cross-Entropy (BCE).

**TABLE V. CNN ARCHITECTURE**

| CNN Architecture |
|-------------------|
| Conv2D 1x3x256, ReLU |
| Conv2D 2x3x256, LeakyReLU(0.3) |
| Conv2D 1x3x80, ReLU |
| Dense, 128, ReLU |
| Dense, 2, Sigmoid |

**C. Attack Correlations**

**Base Architecture Trained on Original Dataset:** This section studies the transferability of learning using the proposed base architecture, training the CNN architecture with one attack, and testing on all the other attacks. A two-class classification problem has been considered to implement this. All attack data has been labelled as one class and all benign data as another class. The model has been trained on benign data and attack data corresponding to the attack whose transferability has to study. As seen in Table I, attacks like Attack 8 (HeartBleed), Attack 9 (Infiltration) and Attack 13 (Web Attack: SQL Injection) are severely underrepresented in the dataset. A synthetic and bootstrapped dataset has been discussed to address this problem. In this section, the models are trained with only the original dataset. In confusion matrices in the results section, Attacks 8, 9, and 13 are not included as there are not enough original samples in the dataset for transferability in learning. The model is trained for 20 epochs for each attack, the loss function used is Binary CrossEntropy, and the optimizer used is Adam.

**Base Architecture Trained on Synthetic and Bootstrapped Dataset:** The CICIDS 2017 dataset is unbalanced, with 80% of the data samples being benign data packets and 20% of the data samples belonging to attack data. The total attack data is further divided into fourteen classes of attacks; thus, each class is severely underrepresented in the dataset, which is not ideal for learning. The main goal of this section is to test the model for strong and consistent attack correlations when it is trained with a greater number of samples from the minority attack classes.

**Synthetic Minority Oversampling Technique (SMOTE):** More attack data has been generated per class using Synthetic Minority Oversampling Technique (SMOTE). SMOTE selects examples that are close to each other in the feature space. This technique selects one point from the minority attack class and uses K nearest neighbours algorithm to find the K nearest data points around the randomly chosen point. The number K is chosen based on how much minority classes have to oversample. In this implementation, 5 nearest neighbours have been chosen. It then randomly selects one neighbour point out of the K neighbours and creates a synthetic data point on the line joining the two selected points. Using this technique, specific regions have been identified in the feature space that can generate more samples belonging to a certain class of data. The number of attack packets generated using this technique is used to match the number of benign data packets in the dataset. This process is repeated for each attack. The model trained with this dataset is tested on: 1) SMOTE generated benign data and real attack data for evaluation of accuracy when tested on a synthetically generated benign dataset and 2) Real benign data and real attack data as this scenario best emulates a real-life scenario.

**Bootstrapped Dataset:** Another technique used to create a balanced training dataset is for every attack that the model is trained with, the attack data is resampled to match the number of benign data packets. This training dataset is bootstrapped dataset. This process is used to replicate attack data corresponding to all attacks. The base architecture is trained with this enhanced dataset and tested on real attack data. During training, it is validated on 20% of the attack data. During training with the SMOTE dataset as well as a bootstrapped dataset, the model is trained for 20 epochs per training attack, the loss function used is Binary CrossEntropy, and the optimizer is Adam.

**Attack Boosting Algorithm:** The previous sections explore transferability in learning when the model is trained with a single attack and tested on other attacks. This section aims to identify a subset of training attacks that can be used to train the model to cause it to scale for other attacks. The possibility of the model scaling for a certain test attack has been explored when trained with a combination of training attacks (that do not include the testing attack). The aim was to observe consistent and strong off-diagonal correlations and identify training attacks. To do this, the attack boosting algorithm has been used as outlined in Algorithm 1. 50% of the set is used only as training data for selecting the training attacks, and the model performance is evaluated on the remaining 50% of the set that serves as the validation data. The testing data consists of the attack for which we want to select a subset of training attacks along with benign data. The loss function used is Binary CrossEntropy, and the optimizer used is Adam.

**Algorithm 1 Attack Boosting Algorithm**

**Require:** Datasets

1: **Result:** Selected Training Attack List
2: **while** Accuracy increase=1 and remaining attack list is not empty **do**
3: **for** single attack in remaining attack list do append single attack data to the current training set (without updating current training set), train model and evaluate on validation set
4: **if** accuracy > appending accuracy **then**
5: appending attack=single attack
6: **end**
7: **end**
8: **if** appending accuracy > current accuracy **then**
9: current accuracy=appending accuracy
10: update selected attacks list
11: remove appending attack from remaining attack list
12: update Accuracy increase
13: **end**
14: **end**
D. Hypothesis for Attack Correlations

In this section, we propose hypotheses for observing the correlations as in Fig. 5.2. In doing so, we use the Recursive Feature Elimination Algorithm [17] to identify the important and dominant features for model learning. We propose hypotheses based on the number of important features selected by the algorithm as well as the properties of the attacks and features selected.

Recursive Feature Elimination is a feature selection algorithm used to identify the important features. A different ML algorithm is used to identify important features in a given feature set and wrapped by the RFE algorithm to do so. This ML algorithm at the core of the wrapper does not have to be the same as the classifier used for the classification task at hand, and it can be other algorithms like Decision Trees, Random Forests, to name a few. RFE provides us with a subset of the entire feature set that is important for learning. This is especially important for datasets with a large number of features to identify the important features for learning and to reduce training times significantly. RFE works by recursively training the model with reduced features and removing features without which the model performs with higher accuracy. RFE also ranks the features in order of importance. The top n features can be selected based on this algorithm.

The CICIDS 2017 dataset has 78 features, a relatively large number of features. The main goal is to identify a smaller subset of features that are important for the learning and scalability of the proposed model and speed up training times. Two implementations of the RFE algorithm have been done in the proposed model. A single attack has been selected from all the attacks for the first implementation. The RFE algorithm is used to identify features that the classifier uses to separate from the benign data. This process has been repeated for all attacks in the dataset in order to identify the common features between attacks to validate the proposed hypotheses. The second implementation involves taking two correlated attacks (as observed in Figure 5) and using the RFE algorithm to identify the number of features that the classifier uses to differentiate both attacks from benign data. The Decision Tree Classifier has been used as the core model for RFE. The training data for both these implementations contain the attack and benign data.

IV. RESULT AND DISCUSSION

A. Comparison between HMM Performance and CLDNN Performance

To observe if the statistical modelling-based approach scales better than the deep learning approach, the performance of the HMM compared with a candidate CLDNN model (Table VI). It has been observed that the CLDNN model is able to achieve an overall accuracy of 99.69% for 2 classes of data and 99.57% for 4 classes of data. In contrast, the HMM model is only 78.7% accurate for 2 classes of data and 50.8% accurate for 4 classes of data. This could be because not all the attacks in the dataset are multi-stage attacks that correspond to the hidden states of the HMMs. The performance of the HMM model deteriorates as the number of classes in the dataset is increases.

Table VI concluded that the deep learning approach shows more promise for the multi-class classification problem. Therefore, more deep learning architectures have been explored in the following sections to identify the best architecture to study attack transferability patterns. The next section gives the architecture details of the used candidate CLDNN model.

B. Result of the Neural Network Architectures

1) Result of the CLDNN Architecture on the Multi-class Classification Problem: An attack accuracy metric is used to evaluate the performance of a model other than the overall accuracy metric [18]. This metric was used because the number of benign packets is much larger than the attack data. A gross misclassification of the attack data packets does not diminish the overall accuracy considerably. A more accurate metric of performance evaluation would be the percentage of correctly classified attack data. Therefore, the attack accuracy is defined as the ratio of the number of correctly classified attack packets in a class to the total number of attack packets in that class. This is also called true positive rate more generally.

\[
\text{Attack Accuracy} = \frac{\text{Number of correctly classified attack packets in class } i}{\text{Total number of attack packets in class } i}
\]

Fig. 1 is the confusion matrix of the performance of this CLDNN architecture. The labels on the Y-axis of the matrix indicate the true labels of the testing data. The labels on the X-axis of the matrix indicate the predicted labels of the testing data. A model with good performance observed high accuracy along the diagonal of the confusion matrix, which indicates that the model correctly classifies the attack with which it is trained with high accuracy. It has been observed that the proposed CLDNN architecture achieves an overall accuracy.
of 98.53%. For most classes, the model is able to classify the attacks correctly. However, it can be observed that although the model is being trained on data from Attacks 8 (HeartBleed), 9 (Infiltration), and 13 (Web Attacks: SQL Injection), it is not able to classify them with a good attack accuracy correctly. This is mainly because Attacks 8, 9 and 13 are significantly underrepresented in the dataset. It could also be that there are not a lot of dominant features that influence the learning of the model. In addition, it is also a possibility that the important features do not have a large variance in their distribution. This makes it hard for the model to learn patterns to differentiate between the data classes.

2) Result of the CLDNN Architecture on the Two-class Classification Problem: Fig. 2 is the confusion matrix for the two-class classification problem. In this confusion matrix, the labels on the Y-axis indicate the attack with which the model has been trained. The labels on the X-axis represent the class of attack that the model is tested on. A model trained on one attack is tested on all other attacks for transferability. More off, the diagonal correlations are preferred for a model that exhibits attack transferability. High classification accuracies on the diagonal represent the model’s performance when it is trained and tested on the same attack. However, to observe transferability in learning, it will be best to observe high accuracies when the model is tested on an attack class that it has not been trained on. It can be observed that training this CLDNN model with one attack does not correlate to any other attack. Most of the correlations observed are on the diagonal of the confusion matrix, which means the model only performs well when tested on the training attack. It exhibits low transferability; for example, when the model is trained with Attack 12, it can identify attacks in class 14 with 91.87% accuracy. It does not exhibit other strong off the diagonal correlations. Therefore, it has been concluded from these results that the transferability of learning cannot be tested using this architecture.

3) Results on CNN with BGRU Layers and CNN with Stacked BGRU Layers: These results present a comparative study of the performance of the CLDNN, CNN with BGRU layer and CNN with stacked BGRU layers for select attacks for the two-class classification problem. The performance of a CNN has also been evaluated with a BGRU layer with 10 cells. The attack accuracy was evaluated when tested on each attack. All the entries are attack accuracies in terms of percentages.

### Table VII. Comparison of Performance of LSTM, BGRU and Stacked BGRU Architectures on All Testing Attacks When Trained with Attack 4 (DoS: HULK)

| Testing Attack | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| LSTM           | 0.45| 3.3 | 98.8| 0.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.5 |
| BGRU (10)      | 0.46| 0.0 | 98.3| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.6 |
| BGRU (50)      | 0.43| 0.7 | 98.7| 9.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 3   |
| Stacked BGRUs  | 0.47| 31.3| 99  | 3.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 3   |

### Table VIII. Comparison of Performance of LSTM, BGRU and Stacked BGRU Architectures on All Testing Attacks When Trained with Attack 2 (DDoS)

| Testing Attack | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| LSTM           | 0.99| 86.3| 99.9| 2.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| BGRU (10)      | 0.99| 84.5| 99.4| 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| BGRU (50)      | 0.99| 99.9| 99.9| 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Stacked BGRUs  | 0.99| 99.9| 99.9| 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

### Table IX. Comparison of Performance of LSTM, BGRU and Stacked BGRU Architectures on All Testing Attacks When Trained with Attack 3 (DoS: GoldenEye)

| Testing Attack | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| LSTM           | 0.27| 96.9| 96.9| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| BGRU (10)      | 0.0 | 96.6| 99.6| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| BGRU (50)      | 0.11| 97  | 97  | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Stacked BGRUs  | 0.30| 96.5| 96.5| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

### Table X. Comparison of Performance of LSTM, BGRU and Stacked BGRU Architectures on All Testing Attacks When Trained with Attack 5 (DoS: Slowhttptest)

| Testing Attack | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| LSTM           | 0.0 | 0.0 | 0.0 | 77.8| 34.3| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| BGRU (10)      | 0.32| 0.0 | 98.8| 35.9| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| BGRU (50)      | 0.0 | 0.0 | 76  | 34.1| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Stacked BGRUs  | 0.0 | 0.0 | 90.5| 34.7| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

From Tables VII, VIII, IX, X, and XI, the models yield high accuracies only when tested on the training attack. Small correlations can be studied, for example, when the models are trained with Attack 5 (DoS: Slowhttptest) and tested on Attack 6 (DoS: Slowloris) and vice versa. However, the prediction accuracies are still low (about 28%-35%), which is insufficient to
TABLE XI. COMPARISON OF PERFORMANCE OF LSTM, BGRU AND STACKED BGRU ARCHITECTURES ON ALL TESTING ATTACKS WHEN TRAINED WITH ATTACK 6 (DoS: Slowloris)

| Testing Attack | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|----------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|
| LSTM           | 0 | 0 | 0 | 0 | 31| 99.44| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BGRU (10)      | 0 | 0 | 0 | 0 | 33| 99.44| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BGRU (50)      | 0 | 0 | 0 | 0 | 31.13| 98.24| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Stacked BGRU   | 0 | 0 | 0 | 0 | 28.91| 98.24| 31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

conclude that those attacks have a strong correlation. It can also tell from Table 4.4 that training the model with Attack 4 (DoS: HULK) has the potential to scale for Attack 2(DDoS) and Attack 3 (DoS: Hulk), but this model only detects those attacks with 31%-47% accuracy. This comparative study concludes that the architectures explored so far do not exhibit very strong attack transferability. The next section will explore another deep learning architecture, a One Class Neural Network and evaluate its performance for anomaly detection.

C. Results of One Class

The implementations are the implementation by Chalapathy et al. [16][19]. The raw data is passed through an autoencoder to find important features. It is then passed through a one-layer NN, and the function it optimizes is given by equation 1. For the CICIDS dataset, the model AUROCs were computed for different seed values.

From Table XII, it can observe that the model performs as good as random guessing in 30% of the runs, marginally better than random guessing in 60% of the runs and worse than random guessing in 10% of the runs. On average, it achieves an AUROC of 0.55. Therefore, It concluded from the AUROC values that this model cannot detect anomalies with high accuracy and would not be suitable for the two-class classification task at hand. The performance of a Convolutional Deep Neural Network has been evaluated in the next section.

TABLE XII. AUROC VALUES FOR DIFFERENT SEED VALUES

| AUROC values for different seed values |
|---------------------------------------|
| 0.60419023864745687 | 0.6040465887906412 | 0.5329012521557807 | 0.4884116825449265 | 0.549943844886464 | 0.3985043859749384 | 0.5154925402064912 | 0.561410034947294 | 0.6227080493949397 | 0.3054279774080824 |

D. Result of CNN Architecture on Multi-class Classification Problem

The overall accuracy of the CNN is 99.65%, whereas the CLDNN achieved an overall accuracy of 98.53%. In the confusion matrix, the diagonal elements represent the percentage of data in each class correctly classified. It has been observed from the confusion matrix that the model is able to classify Attack 8 (HeartBleed) correctly and Attack 9 (Infiltration), that have 11 and 36 data packets in the dataset up to 80% and 81.82% accuracy. Thus, this model can learn the important features from a high dimensional feature set. It concludes that this architecture has performed the best and is the best architecture for studying attack transferability. This architecture does not have recurrent layers, speeding up training time while increasing classification accuracy by 1.12%. This is our proposed base architecture.

E. Results of Attack Correlations

1) Results of Base Architecture Trained on Original Dataset: Fig. 4 represents the overall accuracy of the base architecture. Figure 5 is the confusion matrix for when the base architecture is tested for the two-class classification task. The diagonal elements are placeholders as they indicate the performance of the model when tested on the training attack. From Fig. 3, it has been observed that the model performs with high accuracies when trained and tested on the same attack. Observed Correlations have been shown in Table XIII. It observes transferability in the off-diagonal elements as:

1) Training with DoS:Golden Eye(Attack 3) scales for DDoS attacks(Attack 2), DoS:HULK(Attack 4), DoS:Slowhttpstest(Attack 5) and DoS:Slowloris(Attack 6).
2) Training with DDoS Attack (Attack 2) scales for DoS:HULK(Attack 4) and vice versa.
3) Training with DoS:Slowhttpstest(Attack 5) scales for DoS:GoldenEye(Attack 3) and DoS:Slowloris(Attack 6).
4) Training with DoS:Slowloris(Attack 6) scales for DoS:GoldenEye(Attack 3), DoS:HULK(Attack 4) and DoS:Slowhttpstest(Attack 5).

TABLE XIII. OBSERVED CORRELATIONS

| Training Attack | Correlated Attacks |
|-----------------|-------------------|
| DDoS(2)         | DoS:HULK(4)       |
| DoS:GoldenEye(3) | DDoS(2),DoS:HULK(4),DoS:Slowhttpstest(5),DoS:Slowloris(6) |
| DoS:HULK(4)     | DoS:Slowhttpstest(5) |
| DoS:Slowloris(6) | DoS:GoldenEye(3),DoS:Slowloris(6) |
| DoS:Slowhttpstest(5) | DoS:GoldenEye(3),DoS:HULK(4),DoS:Slowhttpstest(5) |
2) Results of Base Architecture Trained on Synthetic and Bootstrapped Dataset: Fig. 6, Fig. 7 and Fig. 8 are the confusion matrices of the base architecture when trained on the SMOTE dataset and bootstrapped dataset and tested on SMOTE and real attack data. From Fig. 5, it can observe that most of the attack pairs that showed a correlation when the model was trained with the original training data also showed correlations when the model was being trained with synthetic and bootstrapped data. Like the previous section, the diagonal elements in the confusion matrices are placeholders as they represent the accuracies when training and testing on the same attack that already observed scales well with accuracies above 90% from Fig. 3.

3) Results of Attack Boosting Algorithm: As observed from Table XIV, the algorithm chooses Attack 11 (Brute Force: SSH Patator) as the selected training attack when tested on Attack 6 (Slowloris). This produces a very low PR. In Fig. 5, it can observe that training the model with Attack 3 (DoS: GoldenEye) alone yields a true positive accuracy of 95.26% and Attack 5 (DoS: Slowhttptest) alone yields a true positive accuracy of 91.16% when tested on Attack 6 (DoS: Slowloris). Table XV can observe a Positive Rate of 0.877 when the model was trained on Attack 11 data and tested on Attack 6. However, this does not outperform the model’s performance, which was trained with a single attack. From Table XIV can also be observed that the selected attacks for Attack 1 cannot correctly classify Attack 1. The selected attack for testing attack Attack 2 also achieves a low PR. From Figure 5, it can observe that training the model with attack data corresponding to only Attack 3 or Attack 4 yields true positive accuracies of 80.53% and 93.87% when tested on attack 3 (as seen in Fig. 5).

Comparing the positive rates from Tables XIV and XV with Fig. 5, it concludes that training with one kind of attack scales better than training with a subset of attacks. Therefore, this algorithm has not been employed to further study attack
perform well on Slow HTTP Attacks like Attack 5 and Attack 6 that, are Slow HTTP Attacks performed with Slowloris and Slowhtptest attack tools and vice versa. We hypothesize that this is because Attacks 3, 5, and 6 are very similar in the way they are performed.

Training with DoS: slowHTTPattack simulated with Slowhtptest scales well for Slowloris and vice versa as they are essentially the same attack but simulated using different tools. In Slow HTTP attacks, the attacker opens multiple HTTP connections with the target machine and keeps them open by sending packet headers or bodies slowly, thereby keeping all available connections occupied, making the target unavailable to legitimate requests. Since Attack 5 and Attack 6 are both the same kind of attack, it is possible for the model to learn attacking trends.

DoS: A goldenEye is an attack tool that explores vulnerabilities in the target; it tests how susceptible a target is to a DoS attack. DoS: GoldenEye opens multiple HTTP connections with the target server and keeps those connections open with keep-alive headers and caching control options. The connections cannot be closed, making the server unavailable to other HTTP requests. Slow HTTP attacks also open multiple HTTP connections with the target and keep open connections by sending large volumes of data at a slow rate, thereby making the target unavailable to other users. Whether it’s for the keep-alive header or data being received slowly, the target is unable to close these connections. The same applies to Slowloris/Slowhtptest testing tools. This implies that although DoS: GoldenEye is not a Slow HTTP attack, it operates similar to Slow HTTP attacks. This could explain why training with DoS: GoldenEye Scales for both the Slow HTTP attacks.

**Hypothesis 2:** It has been observed from Figure 5 that training the model with attack data from DoS: HULK (Attack 4) causes it to scale for DDOS attacks (Attack 2) and vice versa. Training with Distributed Denial of Service attacks scales for DoS: HULK attack because the DDOS training attack was generated with Low Orbit Ion Cannon that simulated UDP, TCP or HTTP flooding attacks by sending enormous traffic

---

**TABLE XVII. FEATURES SELECTED BY RFE FOR SELECT TRAINING-TESTING ATTACK PAIRS**

| Select Attack Pairs | Selected Features |
|---------------------|------------------|
| (3,2)               | Destination Port, Total Length of Fwd Packets, Subflow Fwd Packets, Subflow Bwd Packets, Init Win bytes forward |
| (3,4)               | Destination Port, Bwd Packet Length Std, Flow IAT Mean, Packet Length Mean, Init Win bytes forward, Idle Min |
| (6,4)               | Destination Port, Total Length of Bwd Packets, Bwd Packet Length Std, Flow IAT Mean, Bwd IAT Std, Bwd IAT Mean, Bwd IAT Std, Packet Length Mean, Bwd Avg Bulk Rate, Subflow Fwd Packets, Init Win bytes forward, Init Win bytes backward, Active Mean, Active Std |
volume to the target machine. DoS: HTTP Unbearable Load King is a flooding attack where the target is flooded with HTTP data packets, sending GET requests and hitting its resource pool. Both these attacks are flooding attacks that may cause them to scale for each other.

**Hypothesis 3:** Fig. 5 shows that training the model with DoS: GoldenEye (Attack 3) causes the model to scale for DDoS (Attack 2) and DoS: HULK (Attack 4); however, this observed correlation does not apply the other way around. Similarly, training the model with DoS: Slowloris (Attack 6) causes it to scale for DoS: HULK (Attack4) but not vice versa. The focus was on these asymmetric comparisons in this hypothesis. To explain this, it has been hypothesized that correlated attacks will have few dominant features selected by the algorithm, which may explain the correlations between the attacks. In contrast, the uncorrelated attacks will have many features selected by the RFE algorithm, indicating that there are not many dominant features that are important for learning which could explain the low correlations. This can be strengthened by the results in Table XVI for Attack pairs (3,2), (3,4), and (3,6) that have 6, 6 and 9 features selected by the algorithm. Each of these attack pairs is correlated. However, for uncorrelated attacks like (3,7) and (3,1), the RFE algorithm selects 24 and 21 features which are considerably more.

In addition to these hypotheses, some features observed from the selected feature list from Table XVII provide validation for our hypotheses. For example, in addition to DoS: GoldenEye and DDoS attacks having only a few features selected in their reduced feature set, the features selected as indicated in Table XVII are Subflow Fwd packets, Subflow Fwd bytes, Subflow Bwd packets, Subflow Fwd packets are the average number of packets in a sub-flow in the forward direction. Subflow Fwd bytes is the average number of bytes in a subflow in the forward direction, Subflow Bwd packet is the average number of bytes in a subflow in the backward direction. Subflows in the forward and backward direction are usually associated with distributed systems which point to these two attacks being distributed attacks. DoS: The GoldenEye attack tool can be used as a distributed attack as well by specifying the number of workers while simulating the attack. This provides further explanation for generalization.

Similarly, to study the correlation between DoS: GoldenEye, DoS: Slowhtptest, DoS: Slowloris and DoS: HULK, the RFE algorithm chooses features Inter-Arrival Time, Bytes Sent in the Forward Direction, Bytes sent in the Backward Directions in the Initial Window which is coherent with the hypotheses for symmetric comparisons.

V. CONCLUSION

The deep learning approach shows that it generalizes well for this dataset in comparison to explicit statistical modelling-based methods. After evaluating the performance of several candidate DNN architectures like CLDNN, CNN with BGRU, OCNN and CNN, a CNN architecture has been proposed with the highest accuracy in the multi-class classification problem (Fig. 3). Further, this architecture has been used to study attack transferability and observe its performance in Fig. 5. Training attacks have been identified, for which the model generalizes well on certain testing attacks. The problem of underrepresented attacks in the dataset has been addressed using synthetic and bootstrapped dataset based training methods. Fig. 6, Fig. 7 and Fig. 8 show that it can observe the same attack correlations as in Fig. 5 for the performance of the model trained on the SMOTE dataset and bootstrapped datasets. This confirms our identification of training-testing pairs of attacks that exhibit attack transferability.

Furthermore, this study proposes hypotheses for the observed correlations. A Reduced Feature Elimination Algorithm has been used to strengthen the hypotheses. The relationship between the number of features chosen and the training-testing attack pairs can be observed in Table XVI. Table XVII lists feature selected by the algorithm for selected attacks coherent with the proposed hypotheses.

This work validates the hypotheses with features from the RFE algorithm. Future work could validate these hypotheses using transferability in learning for different datasets, for example, DARPA 2009 IDD, KDDCup99, NSL-KDD, UNSW-NB15, and WSN-DS. Future work could also analyze specific or selected features trends in studying attack correlations. It is interesting to observe from Fig. 5 that it only observed correlations in attacks that are different types of DoS attacks. It did not observe correlations from other kinds of attacks, for example, Attack 1 (Botnets), Attack 7 (Brute Force: FTP-Patator), Attack 8 (Heartbleed), Attack 9 (Infiltration) and Attack 13 (Web Attack: SQL Injection). Future research can focus on identifying attributes that could help the model scale for these attacks and testing for transferability of learning for these attacks. Future work could also focus on attack detection when there is more than one attack being performed simultaneously.

ACKNOWLEDGMENT

This work was supported by the Research Center of Computer Science and Engineering, PES Institute of Technology & Management, Shivamogga. The research centre is recognized by Visvesvaraya Technological University (VTU). The authors are grateful to all of those with whom we have had the pleasure to work during this work. Each of my Research Committee members has provided me with extensive personal and professional guidance and taught me a great deal about scientific research and life in general.

REFERENCES

[1] A. Lopez, A. Mohan, and S. Nair, “Network traffic behavioral analytics for detection of ddoS attacks”, SMU Data Science Review, vol. 2, no. 1, 2019.

[2] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp, 1:108-116, 2018.

[3] M. Alkasassbeh, G. Al-Naymat, A. Hassanat, and M. Almseidin, “Detecting distributed denial of service attacks using data mining techniques”, International Journal of Advanced Computer Science and Applications, vol. 7, Jan. 2016. DOI: 10.14569/IJACSA. 2016.070159.

[4] R. Vinayakumar, M. Alazab, K. P. Somam, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, “Deep learning approach for intelligent intrusion detection system”, IEEE Access, vol. 7, pp. 41,525–41,550, 2019. DOI: 10.1109/ACCESS.2019.2895334.

[5] I. Sharafaldin, A. H. Lashkari, Ghorbani, and A. A. “Toward generating a new intrusion detection dataset and intrusion traffic characterization.”, in ICISSp, 2018, pp. 108–116.
[6] M. A. Ferrag, L. Maglaras, S. Moscholios, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study”, Journal of Information Security and Applications, vol. 50, Dec. 2019. DOI: 10.1016/j.isa.2019.102419.

[7] J. Zhao, S. Shetty, and J. W. Pan. Feature-based transfer learning for network security. In MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM), pages 17-22. IEEE, 2017.

[8] Zhao, J., Shetty, S., Pan, J. et al. Transfer learning for detecting unknown network attacks. EURASIP J. on Info. Security 2019, 1 (2019). https://doi.org/10.1186/s13635-019-0084-4

[9] Z. Taghiyarrenani, A. Fanian, E. Mahdavi, A. Mirzaei, and H. Farsi. Transfer learning based intrusion detection. In 2018 8th International Conference on Computer and Knowledge Engineering (ICCKE), pages 92-97, 2018.

[10] P. Wu, H. Guo, and R. Buckland. A transfer learning approach for network intrusion detection. In 2019 IEEE 4th International Conference on Big Data Analytics (ICBDA), pages 281-285, 2019.

[11] A. Singla, E. Bertino, and D. Verma. Overcoming the lack of labeled data: Training intrusion detection models using transfer learning. In 2019 IEEE International Conference on Smart Computing (SMARTCOMP), pages 69-74. IEEE, 2019.

[12] M. Masum and H. Shahriar. Ti-nid: Deep neural network with transfer learning for network intrusion detection. In 2020 15th International Conference for Internet Technology and Secured Transactions (ICITST), pages 1-7. IEEE, 2020.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for largescale image ecognition. arXiv preprint arXiv:1409.1556, page arXiv:1409.1556, Sept. 2014.

[14] H. Dhillon and A. Haque. Towards network traffic monitoring using deep transfer learning. In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pages 1089-1096.IEEE, 2020.

[15] R. Vinayakumar, K. Soman, and P. Poornachandran. Applying convolutional neural network for network intrusion detection. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pages 1222-1228. IEEE, 2017.

[16] R. Chalapathy, A. K. Menon, and S. Chawla, Anomaly detection using one-class neural networks, 2019. arXiv: 1802.06360 [cs.LG].

[17] Abhijit Das, Pramod and Sunitha B S, “Anomaly-based Network Intrusion Detection using Ensemble Machine Learning Approach” International Journal of Advanced Computer Science and Applications(IJACSA), 13(2), 2022. http://dx.doi.org/10.14569/IJACSA.2022.0130275

[18] Abhijit Das, Pramod and Sunitha B S, “An Efficient Feature Selection Approach for Intrusion Detection System using Decision Tree” International Journal of Advanced Computer Science and Applications(IJACSA), 13(2), 2022. http://dx.doi.org/10.14569/IJACSA.2022.0130276

[19] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp, 1:108-116, 2018.