TC-CNN: A Tree Classified model using AI for identifying malware intrusions in open Networks

P. Ramachandran [1], Dr. R. Balasubramanian [2]

[1]Research Scholar ,PG and Research ,Department of Computer Science, J.J.College of Arts and Science (Autonomous),Pudukkottai

[2]Ph.D,Research Guide and Professor ,PG and Research ,Department of Computer Science,J.J.College of Arts and Science (Autonomous),Pudukkottai

Abstract: Proliferation of the internet by multiple devices has led to dramatic increases in network traffic. The Internet medium has also been growing with this usage, but this fast growth has also resulted in new threats making networks vulnerable to intruders and attackers or malicious users. This has made network security an important factor due to excessive usage of ICT (Information and Communications Technology) as threats to IVTs has also grown manifold. Securing data is a major issue, especially when they are transmitted across open networks. IDSs (Intrusion Detection Systems) are methods or techniques or algorithm which cater to detection of intrusions while on transit. IDSs are useful in identifying harmful operations. Secure automated threat detection and prevention is a more effective procedure to reduce workloads of monitors by scanning the network, server functions and inform monitors on suspicious activity. IDSs monitor systems continually in the angle of threat. This paper's proposed technique detects suspicious activities using AI (Artificial Intelligence) and analyzes networks concurrently for defense from harmful activities. The proposed algorithm's experimental results conducted on the UNSW_NB15_training-set shows good performances in terms of accuracy clocking above 96%.

Index Terms—IDSs, ANNs, Machine learning, Artificial Intelligence, Network Security

Introduction: Fast-paced technological growth and their advancements have encouraged organizations to adopt ICTs. Organizations have expanded and users have increased, hand held devices and mobiles are found in billions. These factors have contributed to excessive growth of internet usage. Millions of packets are being transmitted using the medium of internet where traffic through these networks is heterogeneous and consists of flows from multiple applications and utilities. Traffic from hand held devices mainly smart phones have also increased with an expected four old increase globally [1]. Thus, current digital age has created an environment where every action is routed through the internet making it vulnerable in terms of security and ICTs may be compromised. Internet is full of dangers including malware and DDOS attacks. Figure 1 displayed details on malware attacks.

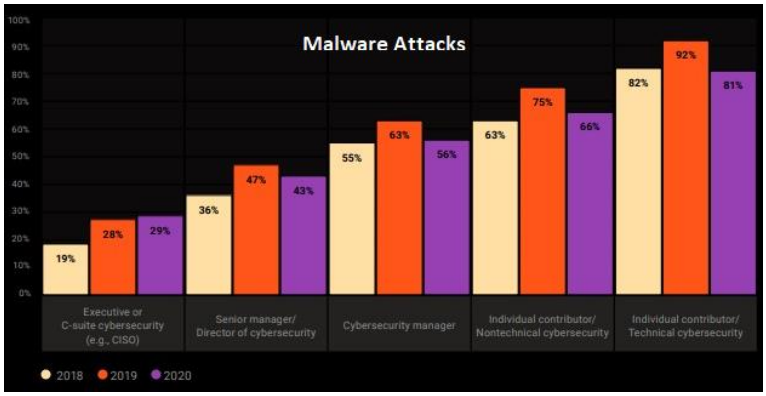


Fig. 1 – Malware Attacks (Source: https://www.comparitech.com/antivirus/malware-statistics-facts/)

Moreover, sensitive data is also increasingly being stored digitally where security flaws could leak this private data including passwords. It is not just the leak of data that creates an issue, but protecting computers or networks

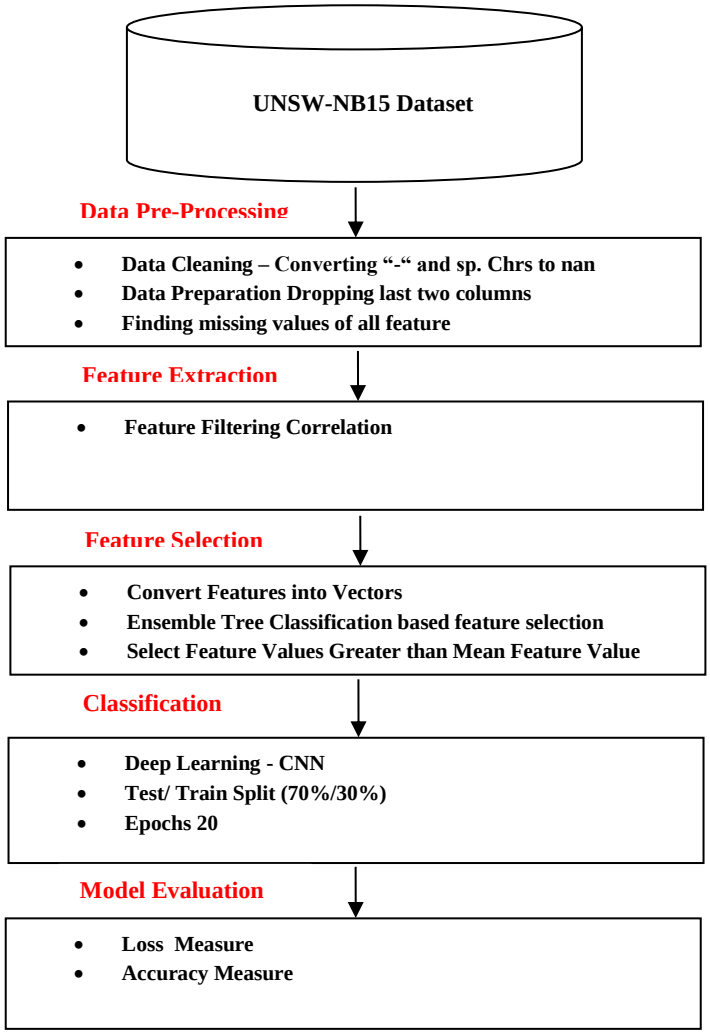
against malware is also important. Network services like monitoring, accounting, control, and optimization can improve their security standards. Policies like bring-your-own-device have also been enabled in the past by corporate to manage access to their resources [2]. Taking these factors into consideration, the ability to detect and prevent attacks on network systems becomes significant. Networks can be protected against such attacks making IDS an essential layer of ICTs considered in the angle of global cyber safety. IDSs can also help in detecting intrusions or identifying attack types when used in a proper sense. They have played a crucial role in warning about severe attacks like Probe, DDOS, U2R and R2L [3][4]. This issue increases specially for large datacenters. Voluminous amounts of data are transmitted through networks by these data centers. IDSs detect intrusions and generate alerts on intrusions. This is done in IDSs, by analyzing network traffic. IDSs can alert administrators on malicious behavior. Most past IDSs needed manual operations, but with the introduction of MLTs (Machine Learning Techniques) this need changed. MLTs are a promising area for automatic IDSs. IDSs which are multilayered in detections can handle novel attacks on networks by autonomously adapting to this new data. Though ICTs can be shielded from vulnerabilities by anomaly detections, these detections are highly complex in nature as they are based on rules which need domain experts to identify anomalies. Protocols defined for these detections have to be implemented and even if implemented harmful activities following regular patterns go unnoticed. In the field of cyber security, MLTs have been playing an essential part with the potential of DLTs (Deep Learning Techniques) which have been tested for solving various kinds of problems. DLTs like ANNs (Artificial Neural Networks) are a part of AI (Artificial Intelligence) that mimic the functions of the human brain. These techniques create complex hierarchical representations through simple building blocks for solving of high-level problems. DLTs have been applied in various cyber security cases [5][6]. Thus, this implies DLTs and IDSs, when combined together, can work be very useful in guarding networks against attacks. Traditional approaches like firewalls, encryption, authentication and VPN have been used in securing network infrastructure from intruders [7]. IDSs can be viewed as an upgraded version of these technologies where identifying network attacks is the bottom line. This paper proposes an IDS system to detect malware in networks using ANNs called TC-CNN (Tree Classified - Convolution Neural Networks). The next section of this paper is an exhaustive review of literature on network attacks and IDS followed by the proposed methodology in section three. The results are displayed in section four while the paper is concluded in the final fifth section.

Review of Related Literature: Malwares develop rapidly and keep appearing increasing their diversity. Traditional static-based malware analysis finds it difficult to trace these new origins of malware. Several methods have attempted to overcome this issue by the usage of signatures on payloads. However, all these methods target only specific aspects of malware and changes to small portions of data by malwares go undetected. Anomaly-based intrusion detection methodology was first introduced in [8] to detect abnormal activity. Since then MLTs have been used with improvements to detect system intrusions. SVM (Support Vector Machine) was combined with GA (Genetic Algorithm) in [9]. RF (Random Forest) was used in [10] for cyber intrusion detections. Malware s were analyzed in [11] for dynamic malware analysis. Malware keep increasing in terms of diversity making their detection through patterns longer in terms of time. The study examined behaviors of malwares using a self created Malheur dataset which was compiled from reports of anti-malware vendors. The\y analyzed by monitoring simulations on CW Sandbox environment. The

scheme used four main steps. They executed malware binaries in their simulated environment and obtained outputs as system calls. Their next step produced sequential reports which then converted their behavior into vectors in a high-dimensional. These vectors were then analyzed geometrically for clustering those using MLTs and further classifications of malware. Their alternations between clustering and classifications showed reduced execution times and memory usage for processing obtained reports. Their incremental technique consumed ¼ th of clustering time in minutes making them conclude that incremental analysis of malware behavior could identify their behavior efficiently. Malware system call sequences were classified using DLTs in [12]. The study used a sample malware sample dataset included private and Virus Share collections. Their scheme examined system call sequences and classified malware for accuracy using CNNs (Convolution Neural Networks) and RNNs (Recurrent Neural Networks). The malware collection inputs from the dataset were input into a Cuckoo Sandbox for getting numerical feature vector outputs which was then trained on NNs (Neural Networks).The CNN layer captured correlations between input vectors in a neighbourhood and generated new feature vectors which were then processed by the RNN layer. LSTM (Long Short Term Memory) cells modeled resulting sequence and mean pooling was used to sort outputs. The study achieved 89.399 % in CNN-LSTM while feed forward networks clocked 79.989 % while CCs produced 89.1890%). Studying data traffic for detecting malware consumes a lot of time, but the study in [13] used DNNs (Deep Neural Networks) for the same and analyzed process behavior. They study used RNNs for extracting features while CNNs were used to classify. RNN training was done with LSTM as RNN are known to create errors while processing their layers. LSTM used in the study helps mitigating errors created by RNN learning and thus extracts only required features. While evaluating malware and process log files in for training and validation, they generated a dataset using Cuckoo Sandbox for running malware emulations. The study used this knowledge to trace malware process for injections. Their system showed 91.89% accuracy in detecting malware, but was not tested on large-scale data. Two important characteristics were studied in [14].The study's dataset was created from malware samples of an anti-virus and compared with a different sha1 generated from previously collected malware samples. The study extracted features, construction using NNs which were then classified by RNN. In NN constructions, network communications were captured while training and classification root node vectors were exposed and predictions were done using this phase. The study reduced analysis time by 66% while 96.9% of URL's got compared. The study in [15] used Microsoft PE file formats to identify malwares. They parsed the PE file for selecting a set of features 100/645. Their created dataset had 5k dirty files and 3k clean files. RF then selected 13/100 features. Other algorithms namely J48Graft and PART algorithms were used to choose top seven features for identifying malware features in systems. Malware behaviors were examined in [16]. The study was an automated malware behavior detection method with the use of MLTs. Their dataset was created from windows System 32 instances for training and subsequent monitoring of systems. Five classifiers namely kNN (k Nearest Neighbour), NB (Naive Bayes), DTs, SVM (Support Vector Machines) and MLPs (Multi-Layer Perceptron) were evaluated on their automatic detection of malware. They selected important malware features using Anubis Sandbox API which monitored system calls and reported as xml files from which the features were chosen. Their experimentation results showed their feature selection reduced features from 5,191 to 116 attributes for MLTs to train thus reducing execution times. The classifiers achieved 94.199 % in true positive rates. Malwares detected by software is vulnerable to infections

or can be disabled by malware. The study in [17] proposed hardware assisted malware detection method to avoid this issue. The proposal was an epoch-based monitoring technique which executed in epochs. These epochs in memory were then used to identify most malicious behavior based on its location and frequency of memory accesses instead of patterns or sequences. On detecting malware in the system, their authenticated handler managed the situation automatically. Their proposal when tested by classifiers yielded a maximum of 99% with RF in terms of TPRs (True Positive Rates). Thus, past and recent studies indicate the use of MLTs and DLTs in intrusion detections and in identifying malwares.

Proposed Methodology: The TC-CNN technique is a tool to detect malware in networks. It follows the main steps of data pre-processing, followed by feature extraction from the UNSW-NB15 dataset. Important features are then selected in TC-CNN which is then input to DNN for classifications. The model is then evaluated for its accuracy. The architecture of TC-CNN is depicted in Figure 2.



Data Pre-processing: TC-CNN initially prepared the data in the dataset by its processing procedures. Data cleaning is the first step of data preparation in TC-CNN. It is the removal of incorrect/incomplete or wrongly formatted or corrupt data found in the samples. This process of cleaning differs for datasets or features of a dataset. Data cleaning in this work is performed by

removing incomplete or irrelevant or duplicate samples. Incorrect or "dirty" data can result in poor decision making. Unwanted feature columns are dropped and all exiting features are checked for null or zero values as shown in Figure 3.

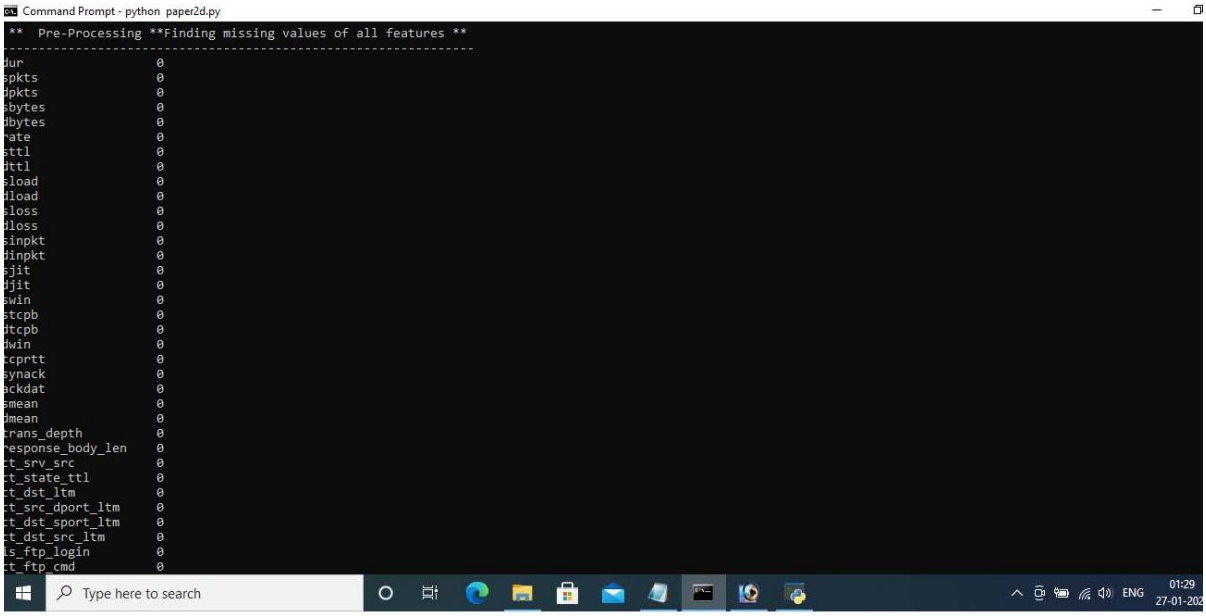


Fig. 3 – TC-CNN Null Value Checks

Thus the output of TC-CNN's data pre-processing produces quality data required for further processes in terms of Completeness, Consistency and Uniformity.

Feature Extraction: Feature extraction implies identifying features that are important to classifications or further selections. Feature extractions can also be executed by filtering unwanted features using some measure. These extractions are executed using various measures like Information Theory, Consistency based measures [18], Chi-Squared based measures, Information Gain measures, Symmetric Uncertainty measures and Correlation-based measures are used for removing the irrelevant and redundant features. TC-CNN uses finds correlations between features to filter unwanted features. The study in [19] proposed Correlation based implementations for selecting suitable features. They discretized the dataset which was then used to compute feature classes feature correlations using symmetric uncertainty. Figure 4 depicts a Graphical Representation of Feature Correlations

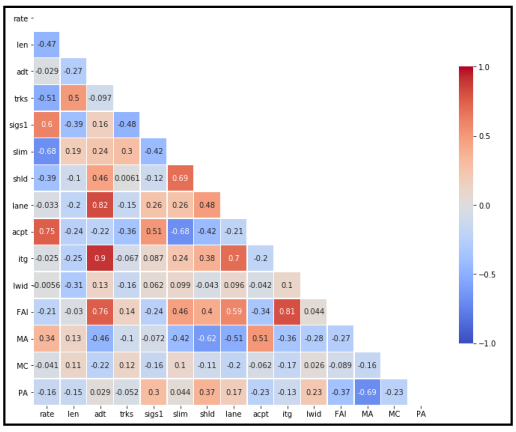


Fig. 4 – Graphical Representation of Feature Correlations

Correlation is a well-known similarity measures between two features. If two features are linearly dependent, then their correlation coefficient is ±1. If the features are uncorrelated, the correlation coefficient is 0. The association between the features is found out by using the correlation method. There are two broad categories that can be used to measure the correlation between two random variables. One is based on classical linear correlation and the other is based on information theory where linear correlation coefficient is a familiar measure and is used by TC-CNN to assess correlations between features. Assuming 'r' is the linear correlation coefficient between a pair of variables (X, Y), r can be computed using equation (1):

$$r = \frac{\sum (X_i - \overline{X_i})(Y_i - \overline{Y_i})}{\sqrt{\sum (X_i - \overline{X_i})^2} \sqrt{\sum (Y_i - \overline{Y_i})^2}} \quad \ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Feature Selection: Feature Selection is a prominent steps before MLTs learn from samples. It is a process of reducing the feature set by choosing most relevant features from the original feature set according to an evaluation criterion and also removing the redundant features from the entire feature set. Assuming Fs is the feature set with larger number of features {Fs_1, Fs_2, Fs_3… …Fs_n} where n is the number of features in the data set. Feature selection F_{sel} can be defined as the process of selecting Fs or most discriminatory features count is greater than or equal to 1. Feature selection methods involve generation of subsets. Feature selection is also a kind of dimensionality reduction. Many feature selection methods like filters [20] wrappers [21] and hybrid methods [22] have been used in studies for a long time. The wrapper model uses the predictive accuracy of a predetermined learning algorithm to determine the goodness of the selected subsets. These methods are computationally expensive for data with a large number of features. The filter model separates feature selection from classifier learning and selects feature subsets that are independent of any learning algorithm. It relies on various measures of the general characteristics of the training data such as distance, information, dependency, and consistency. TC-CNN uses the model , ERTC (Extremely Randomized Trees Classifier), an ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output it's classification result. It parallels RF (Random Forest) Classifier in operations but differs in its manner of construction. The ETF (Extra Trees Forest) is constructed from the original sample. Each tree has random k sample features from the feature-set from which each decision tree selects the best feature to split the data based on the Gini Index. This random sample of features leads to the creation of multiple de-correlated decision trees. To perform feature selection using the above forest structure, during the construction of the forest, for each feature, the normalized total reduction in the mathematical criteria used in the decision of feature of split based on the Gini Importance of the feature. Thus, ETF is used in this study to select the best features and filtered or selected based on Feature Values Greater that are greater than Mean Features Value.

Classification: TC-CNN uses CNN for classifying intrusions from the dataset. CNN or ConvNet is a DLT and a part of AI. CNN learning are preferred as they filer characteristics for classifying data Moreover, CNNs require very limited pre-processing when compared to other classification algorithms. CNNs are analogous to a human brain's neuron connectivity patterns and capture Spatial and Temporal dependencies in the input data. Figure 5 depicts a CNN.

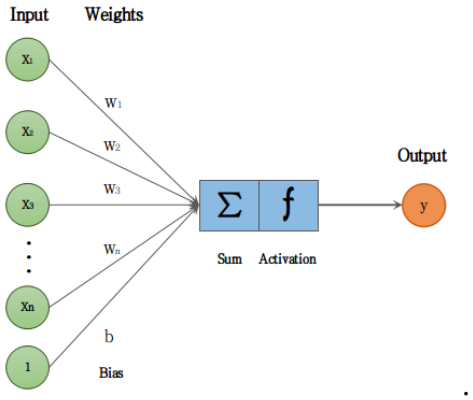


Fig. 5 – CNN

CNNs learn appropriate representations of features from inputs and differ from MLPs in their sharing of weights and pooling. CNN layers have convolution kernels which generate varied feature maps. Neighboring neuron regions are connected to a neuron's feature map of the next layer. While generating the feature map, all input spatial locations are shared by the kernel. Convolutions and pooling layers yield in fully connected layers which are then used for classifying data [19 -21]. CNNs sharing of weights help the model learn same patterns occurring at different input position of inputs which happen without learning separate detectors for each input position. This makes CNN models robust to input translations [22]. Pooling layers reduce computational burdens by reducing count of connections between convolution layers [23]. Mathematically speaking each CNN layer has a set of kernels *which* convolve input data. This convolution between data and kernels produces a new feature map *xk* and the transformation can be defined as equation (2)

$$xk^{l}=\sigma(wk^{l-1} * x^{l-1} + bk^{l-1}) \quad (2)$$

Where, l is the convolution layer, $W=\{w1,w2,\ldots,wn\}$ are n kernels and $B=\{b1,b2,\ldots, bn\}$ are *n* biases. In learning, CNNs use a window with which the values of bias and weights from various features of the input data are optimized irrespective of their position within the input data.

Results and Discussions

The proposed model was evaluated using python 3 on the UNSW-NB15 Dataset compiled by the University of South Wales [24] was downloaded. The proposed system was implemented on AMD Radeon processor with 16gb running 64-bit Windows 10. Keras and Tensor-Flow were used the software framework. The exponentially increasing CNN architecture performed on a GPU enabled tensorflow in a single Nvidia-GK110BGL-Tesla-k40. This data set was chosen as it reflects a more modern and complex threat environment. The full dataset contains a total of 25,400,443 records. The partition of the full dataset are divided into a training set and a test set according to the hierarchical sampling method, namely, UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv. The training dataset consists of 175,341 records whereas the testing dataset contains 82,332 records. The number of features in the partitioned dataset is different from the number of features in the full dataset and has 43 features with the class labels. The partitioned dataset contains ten categories, one normal and nine attacks, namely, generic, exploits, fuzzers, DoS, reconnaissance, analysis, backdoor, shellcode and worms. Figure 6 depicts a snapshot of the dataset

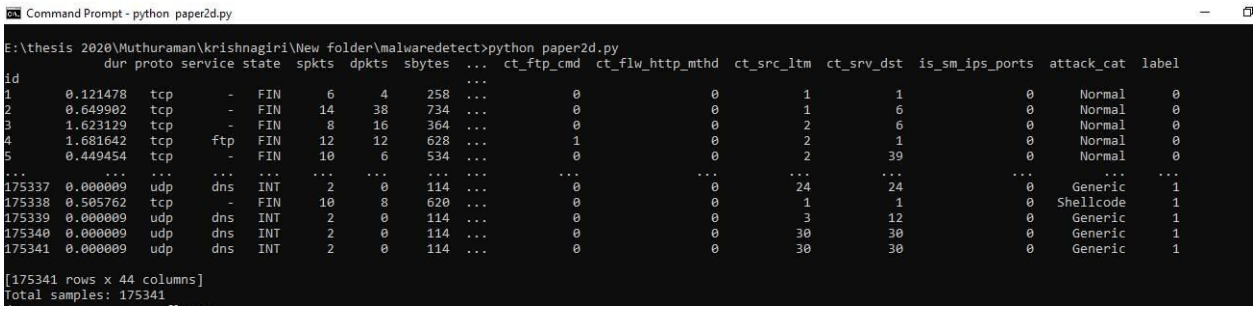


Fig. 6 - UNSW-NB15 Dataset Snapshot

TC-CNN Data Preprocessing: The dataset was cleaned as a preparatory step in this work as it ultimately increases overall productivity and allows quality classifications. Missing values in features are found by using a finite Difference measure and then filled up with the correct value. Figure 7 depicts the output of pre-processing.

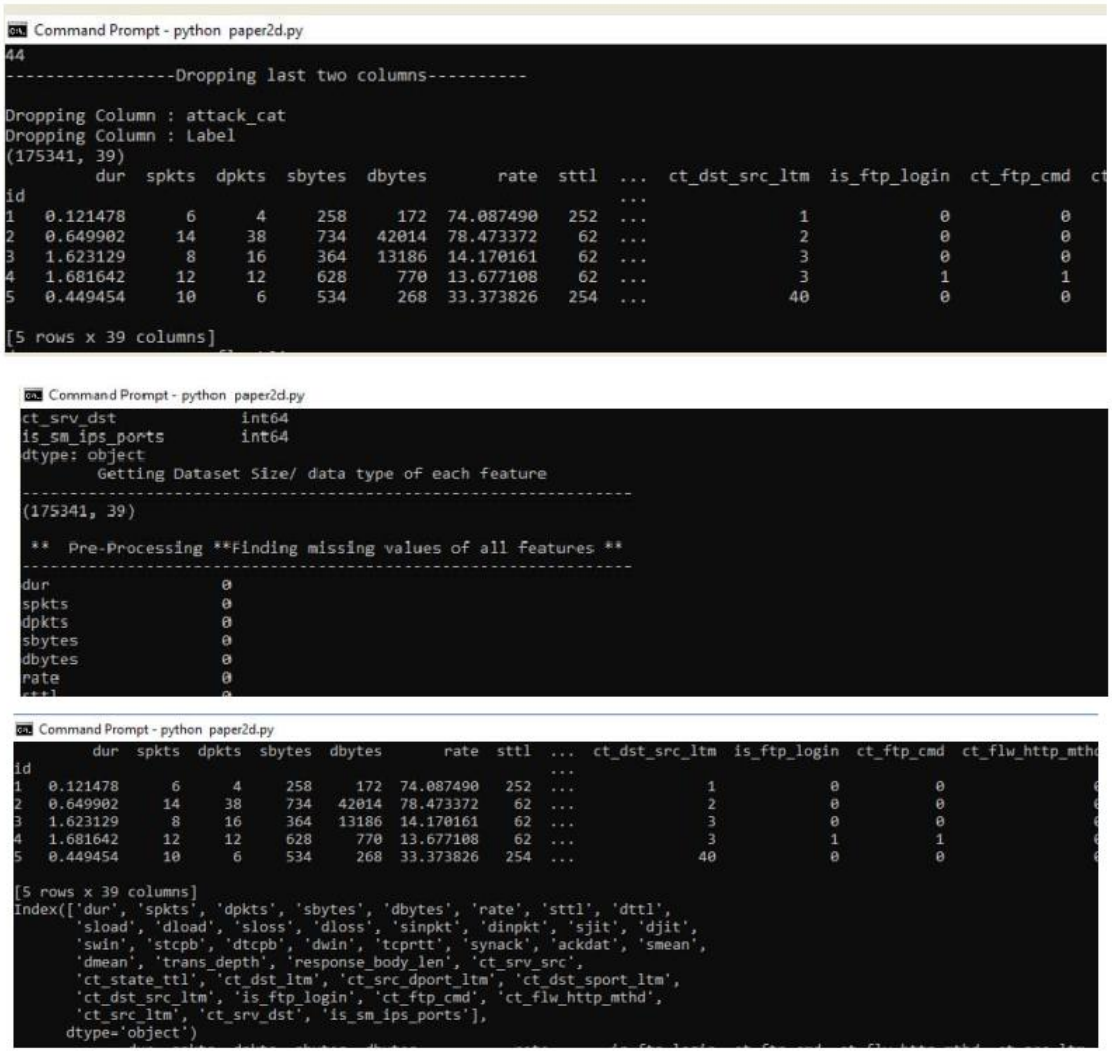


Fig. 7 – TC-CNN Preprocessing outputs

TC-CNN Feature Extraction: TC-CNN finds correlations between features to eliminate unwanted features. The correlation coefficient is used to find out significant features. Thus, feature extraction in this work is a dependency measure. Pearson's Correlation can find association between continuous features. Figure 8 depicts the output of feature extraction along with correlation values of samples.

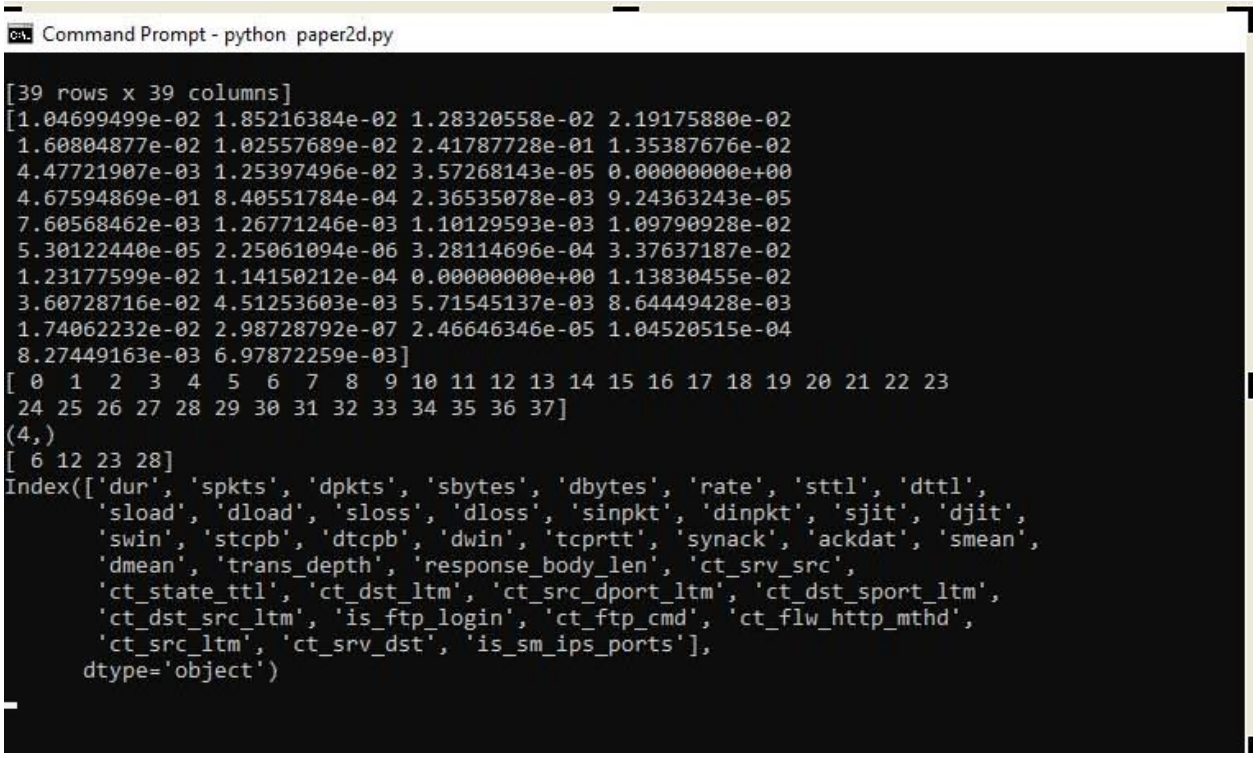


Fig. 8 – TC-CNN Extracted Features

TC-CNN Feature Selection: TC-CNN uses an ensemble Classifier which selects features from the feature-set based on Gini Index. Feature selection in this work is also a dimensionality reduction phase to improve the accuracy of the classifier. Figure 9 depicts the output of feature selections.

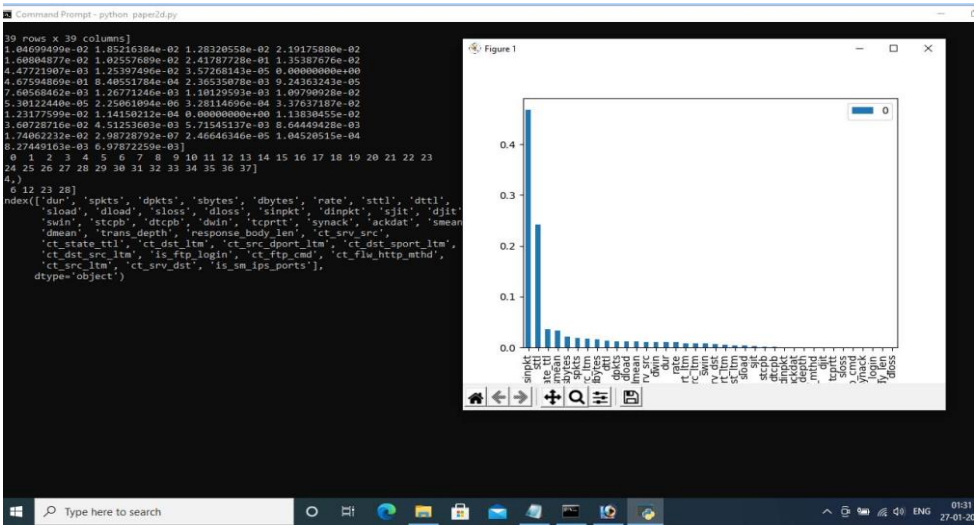


Fig. 9 – TC-CNN Feature Selection Output

TC-CNN Classification: CNN classification in this work uses two convolution, two pooling, and three fully connect layer is used. The kernel size of the convolution layers is [4 *4] and [3 *3] respectively, and the pooling size for both pooling layers are [2 * 2]. Furthermore,

three fully connected layers include 50, 20 and 2 neurons are used. To prevent overflow, a dropout by 0.2 is considered. Moreover, the data was split into a 70/30 combination where 70% of data was used for testing. Figure 10 depicts the data split into 70% in training and CNN learning on the test samples.

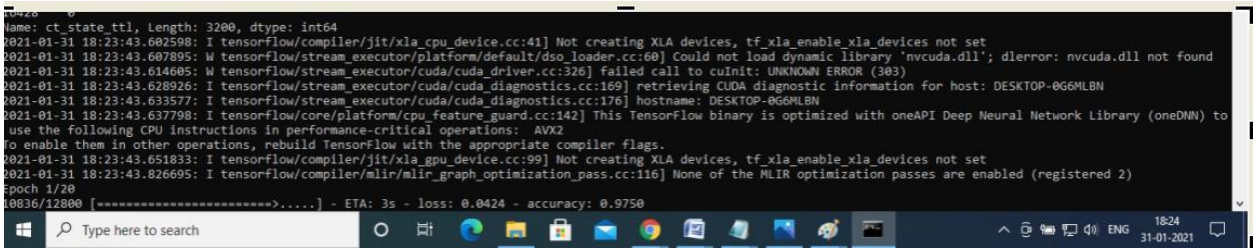


Fig. 10 - TC-CNN data split and CNN Learning

The Rectified Linear Unit (ReLU) activation function is used in all layer except the last layer, which uses the 'Softmax' activation function. For optimization, Adaptive moment estimation (Adam) method is used, and the number of epochs is set to 20. Figure 11 depicts the output of CNN Learning which achieved 97.39 % in terms of accuracy.

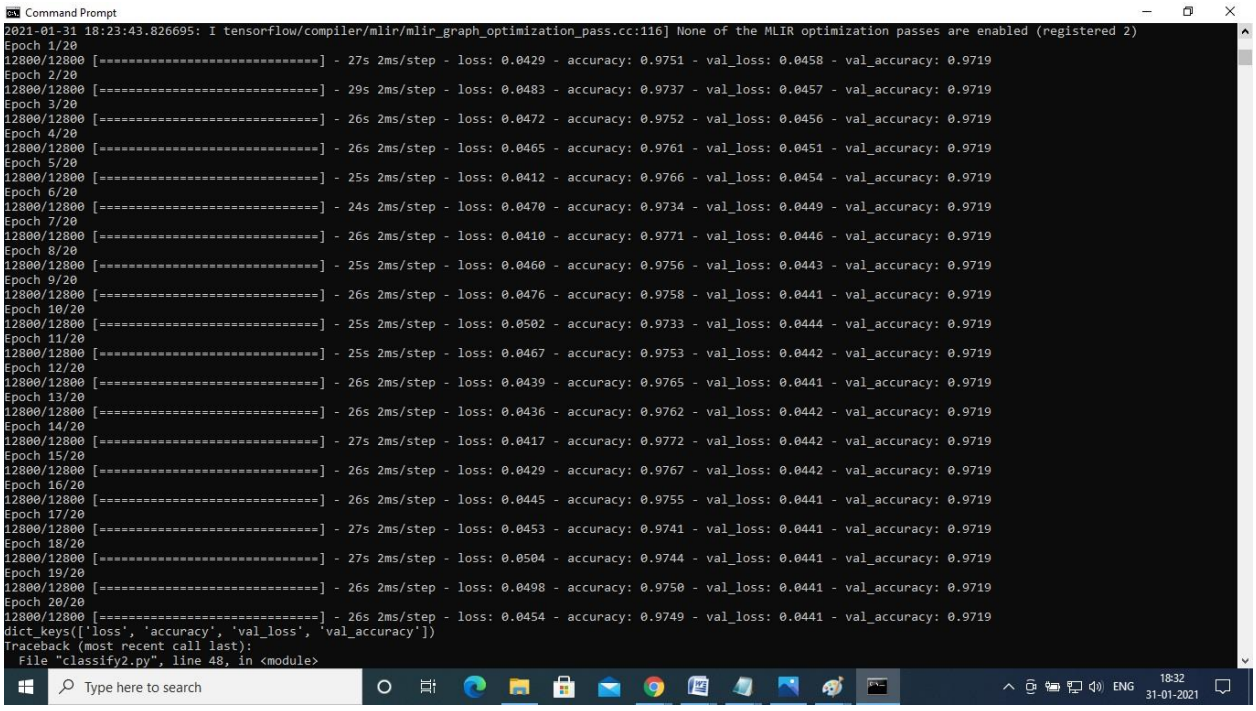


Fig. 11 – CNN Learning Epochs

Model Evaluation: A learning curve is a plot of model learning performance over experience or time. Learning curves are a widely used diagnostic tool in machine learning for algorithms that learn from a training dataset incrementally. The model can be evaluated on the training dataset and on a hold out validation dataset after each update during training and plots of the measured performance created show learning curves.Reviewing learning curves of models during training can be used to diagnose problems with learning, such as an underfit or overfit model, as well as whether the training and validation datasets are suitably representative. A loss function is used to optimize a machine learning algorithm. The loss is calculated on training and validation and its interpretation is based on how well the model is doing in these two sets. It is the sum of errors made for each example in training or validation sets. Loss value implies

how poorly or well a model behaves after each iteration of optimization. An accuracy metric is used to measure the algorithm's performance in an interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage. It is the measure of how accurate your model's prediction is compared to the true data. Figure 12 depicts the Training and validation accuracy while Figure 13 depicts the training value loss of TC-CNN.

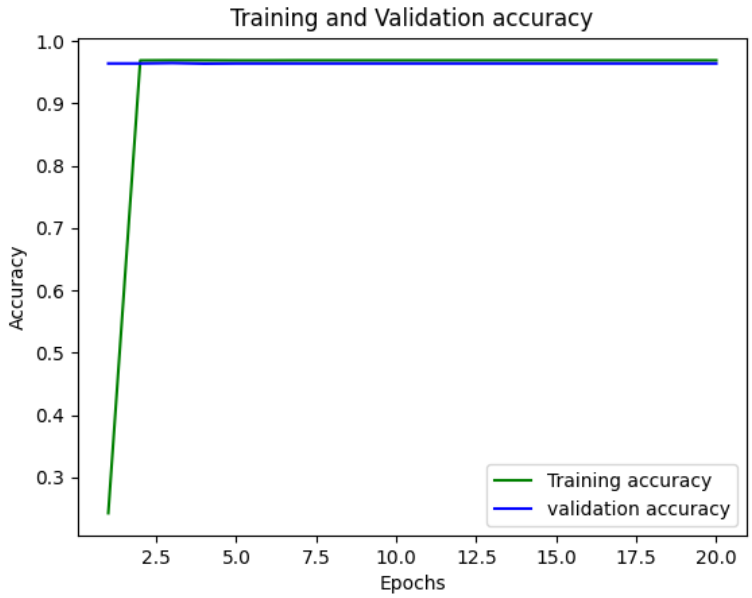


Fig. 12 - Training and validation accuracy of TC-CNN

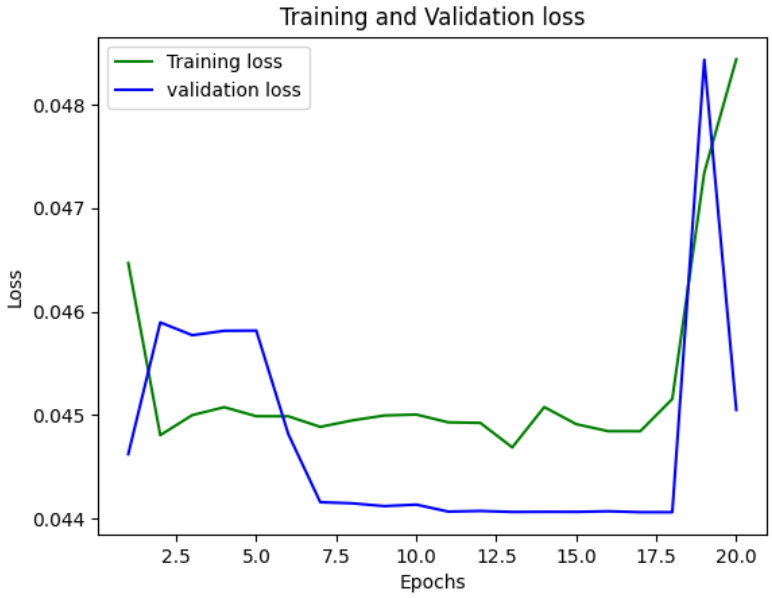


Fig. 13 – Training and Validation Loss of TC-CNN

Conclusion

This paper has implemented the problem of classifying network attacks and identifying malware. The dataset used provided samples for classifying and identifying

corresponding elements. The first part identified important features in the dataset using correlation and clustering based tree classifications for minimizing dimensionality and removing unwanted features. The second part used deep learning to identify and classify attacks on networks in the form of malware. The approach as whole produced an accuracy of 97% in training implying this proposed approach can be implemented in network systems to identify malware. Moreover, the depicted results in form of figures and also imply that TC-CNN is a promising approach and can classify malware samples much faster than all those solutions that rely on the manually extraction of features and thus, are more scalable. It can be concluded the TC-CNN is a viable, implementable technique for identifying malware in online networks. .

Future scope: Even that both approaches have been successfully applied, there is still a huge margin of improvement. Another possible modification is to expand the vocabulary of malware to virus by studing their patters and implementing them as samples for CNN learning. The study also aims to improve byintroducing and evaluating other AI methods for generic identification of attacks on networks and specifically mobiles where the internet is an open and unprotected areas of network access.

References:

[1] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," in Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), 2016, pp. 21–26.

[2] S. Vieira, W. H. L. Pinaya, and A. Mechelli, "Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications," Neurosci. Biobehav. Rev., 2017.

[3] G. LeCun, Yann and Bengio, Yoshua and Hinton, Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp.436–444, 2015.

[4] Y. Yoo et al., "Deep Learning of Joint Myelin-T1w MRI Features on Normal-Appearing Brain Tissues Distinguishes Multiple Sclerosis from Healthy Controls," Mult. Scler. J., vol. 23, no. 2, p. 315, 2017.

[5] M. E. Aminanto and K. Kim, "Deep Learning in Intrusion Detection System: An Overview," Proc. Int. Res. Conf. Eng. Technol., pp. 1–12, 2016.

[6] D. Silver et al., "2016 - Mastering the game of Go with deep neural networks and tree search - DeepMind nature16961," Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[7] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey," arXiv Prepr. arXiv1701.02145, pp. 1–43, 2017.

[8] M. Tavallaee and E. B. and W. a. G. A. A. Lu, "'A detailed analysis of the KDD CUP 99 data set,' in Computational Intelligence for Security and Defense Applications," in Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on, 2009, pp. 1–6.

[9] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: A review," Comput. Secur., vol. 30, no. 6–7, pp. 353–375, 2011.

[10] L. Mohammadpour, M. Hussain, A. Aryanfar, V. M. Raee, and F. Sattar, "Evaluating performance of intrusion detection system using support vector machines: Review," Int. J. Secur. its Appl., vol. 9, no. 9, 2015.

[11] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," Cluster Comput., pp. 1–13, 2017.

[12] G. Litjens et al., "A Survey on Deep Learning in Medical Image Analysis," CoRR, vol. 1702.05747, 2017.

[13] R. C. O'Reilly, D. Wyatte, S. Herd, B. Mingus, and D. J. Jilk, "Recurrent processing during object recognition," Front. Psychol., vol. 4, no. APR, pp. 1–38, 2013.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv Prepr. arXiv1207.0580, 2012.

[15] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks arXiv:1311.2901v3 [cs.CV] 28 Nov 2013," in Computer Vision–ECCV 2014, 2014, vol. 8689, pp. 818–833.

[16] Y. Shi et al., "Early endothelial progenitor cells as a source of myeloid cells to improve the pre-vascularisation of bone constructs," Eur. Cells Mater., vol. 27, pp. 64–80, 2014.

[17] R. Singh, H. Kumar, and R. K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," Expert Syst. Appl., vol. 42, no. 22, pp. 8609–8624, 2015.

[18] H. S. and B. S. D., "KDD Cup '99 Dataset," 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html .

[19] T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, "Efficient dynamic malware anal-ysis based on network behavior using deep learning," in Global Communications Conference (GLOBECOM), 2016 IEEE. IEEE, 2016, pp. 1–7.

[20] K. Raman et al., "Selecting features to classify malware," InfoSec Southwest, vol. 2012, 2012. A Survey on Malware Detection from Deep Learning 79

[21] I. Firdausi, A. Erwin, A. S. Nugroho, et al., "Analysis of machine learning techniques used in behavior-based malware detection," in Advances in Computing, Control and Telecommunica-tion Technologies (ACT), 2010 Second International Conference on. IEEE, 2010, pp. 201–203.

[22] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in Australasian Joint Conference on Artificial Intelligence. Springer, 2016, pp. 137–149.

[23] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual, vol. 2. IEEE, 2016, pp. 577–582.

[24] Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications And Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6