Network Flow-based Intrusion Detection Approach using One Dimensional Convolution Neural Network with Encoded and Normalised Statistical Features

***Abstract*—Cyber Attacks are happening at a rapid rate and targeting more number of organizations. Accurately detecting them in real-time is the need of the hour. Intrusion Detection Systems (IDSes) are widely used to detect such attacks. Recently, deep learning based IDS are gaining popularity. In this work, we propose an IDS in the form of Network Flow Classifier (NFC) based on 1-Dimensional Convolutional Neural Networks (1D-CNNs). The proposed NFC works on Statistical Network Flow Summary (SNFS) extracted from the network flow. The NFC segregates the features present in the input SNFS into Categorical features and Numerical features, and, applies preprocessing operations separately onto them. NFC uses Autoencoders to reduce the dimensionality of preprocessed SNFS. A 1D-CNN classifier is then trained on dimensionality reduced SNFSs to detect the class of network flow, thus identifying the type of the attack. To evaluate the performance of the proposed NFC, experiments were conducted by considering the CSE-CIC-IDS2018 dataset. Experimental results show that the proposed deep learning approach for NFC using 1D-CNN performs as good as machine learning based Random Forest Classifier.**

I. Introduction

In today's cyber-world, accurate detection and mitigation of cyberattacks in real-time are indispensable requirements. Although the countermeasures for cyberattacks are continuously coming-up to, attackers use new evasion techniques to launch cyberattacks, which are increasing year-by-year [1]. Along with compromising the privacy and functionality of networking devices and user devices, cyberattacks also target financial related digital communication. Intrusion Detection Systems (IDS) are being used as one of the solutions to detect such attacks. The IDS are broadly classified as Host-based IDS (HIDS) and Network-based IDS (NIDS) based on the data source they audit. IDS can use either signature-based [2] or anomaly-based [3] or a combination of both the techniques to detect the intrusion. While Signature-based IDSs are proficient at detecting known attacks by maintaining a signature database, they fail to detect new attacks when the corresponding signature is unavailable in the signature database. On the other hand, Anomaly-based IDS use a baseline profile to identify unseen attacks. While are efficient in identifying unseen attacks, they suffer from a high false alarm rate [2].

Apart from Signature based and Anomaly based approaches, IDS can also be formulated as a classification problem when the data about various possible attacks are collected beforehand and labelled. The NIDSes generally use raw data packets and flow statistics as the source of the information to detect intrusions. For structured data such as network flow statistics, machine learning-based classifiers such as Random Forest (RF) [4] and Naive Bayes were extensively used to perform the classification task [5]–[7]. . Recent research works show that Deep Learning-based classifiers are also capable of detecting the intrusions even when the input data is unstructured. The significant advantage of the Deep Learning-based classifiers is that they can learn the features themselves, which are required to perform the classification task. Thus, they can also be used as feature extractors. In this work, the Network Flow-based Intrusion Detection Approach is proposed using One Dimensional Convolution Neural Network (1D-CNN). It is able to analyze network flow-related statistical parameters to detect intrusions.

The main contributions of this work are:

1) Network flow pre-processor is proposed. It consists of Feature Segregator, Binary Encoder, and Normaliser. Binary Encoder is applied to Categorical Features. Normaliser is applied to the features that take numerical values.
2) Set of experiments was conducted to evaluate the performance of the proposed Network Flow-based Intrusion Detection Approach. Obtained experimental results demonstrate that the detection ability of the proposed approach with the Random Forest classifier and 1D-CNN are nearly the same.

The rest of the paper is organized as follows. Section II discusses the related work. In Section III, proposed method is discussed in detail. The experimental results are discussed in IV. Section V concludes the paper.

II. Related Work

The majority of the intrusion detection models are either based on anomaly detection or classification. In classification, the task of the model is to map its input to one of the pre-defined classes. On the other hand, the anomaly detection problem aims to find how far the input is from normal data.

Autoencoder has a higher reconstruction error when it sees a sample which is far from the domain of the data with which it was trained. This fact was used in [8], which trained

autoencoders using benign flow data. This work also employed Variational Autoencoders to define reconstruction probability. These two metrics were used separately to detect anomalous input data. To compare the performance of their proposed approach with classical machine learning models, One-Class SVM was used. In all these models, it is essential to note that the class of the anomalous sample can not be detected.

On the other hand, (Multi-class) classification can be used to detect the type of attack, along with detecting that the flow is malicious. In classification, feature selection plays a vital role by eliminating the features that are not important to the classification and improving classifier metrics. Out of may feature selection methods, Genetic Algorithms were used in [9]. This work used Kernel Extreme Learning Machine as a classification algorithm whose parameters were also tuned using Genetic Algorithms.

It is also possible not to use feature selection at all if all features are essential during the classification. As an example of this, authors of [10] proposed a granular classifier based on if-then rules. While the premise of the rules is realized using entropy clustering, the conclusion part is realized using Support Vector Machines. An opposition-based Genetic Algorithm was used for tuning the parameters of the classifier.

Modern rule generation algorithms such as apriori [11] can be used to mine association rules from large datasets. Authors of [12] applied a hashing based apriori algorithm on the Hadoop MapReduce framework to extract the rules from KDD dataset [13].

The adoption of deep learning-based classifiers is increasing in IDS. [14] uses Deep Neural Network (DNN) trained on NSL-KDD [15] dataset for detecting intrusions. Before feeding the dataset into the DNN classifier, features were normalized, and XGBoost feature selection was applied to select 21 best features out of 41 features. SVM, Linear Regression, and Naive Bayes models were used for comparison purposes.

Labeled datasets used for training intrusion detection models are often imbalanced, with majority class consisting of benign flow data, and, minority classes consisting of malicious flow data. If not handled properly, the classifier might be biased towards the majority class. [16] proposes a two-stage intrusion detection approach consisting of Threat Detection using CTC classifier and Threat Ranking using ForestPA decision tree classifier, the output of which were combined to get the final classification results. These algorithms are best-suited to handle imbalanced data. As the feature selection step, the Infinite Latent Feature Section was used. Experiments were conducted using CICIDS2017 [17] dataset.

Recurrent Neural Networks (RNN) have been used for language modeling tasks extensively [18]. RNNs are used when there is a long-term temporal relationship between the data. As packets of a network flow have temporal interrelationship, it makes RNNs a better candidate in analyzing packet flow data. Authors of [19] have used RNNs for IDS, both with binary and multi-class classification on the NSL-KDD dataset. Long Short Term Memory (LSTM), an extension of RNN, is used in [20] to build an intrusion classification model based on the KDD dataset.

Convolution Neural Networks are popular in extracting structural information from the data. 1D-CNN is one of the popular models to analyze short term temporal data. In [21], experiments were conducted with various parameters of 1D-CNN, and also with several 1D-CNN–RNN hybrid architectures.

A hybrid architecture consisting of CNNs and Bi-direction LSTMs (Bi-LSTM) can efficiently capture long-term and short-term temporal relationships. The authors of [22] convert the NSL-KDD dataset into a flow dataset by grouping the packets of each flow. This data is then encoded and normalized and passed onto 1D-CNN for local feature detection. These extracted features then passed to an array Bi-LSTMs, followed by attention layer and softmax layers.

CNNs can process multi-channel data like RGB images. This enables them to infer inter-channel relationships along with short term temporal/spatial relationships. However, flow datasets often consist of only one channel. In order to augment these datasets, [23] uses two auto-encoders trained separately on benign flow and malicious flow to reconstruct the input vector. These reconstructed vectors are added as a channel to the input vector, making it 3-channel data. This data is passed onto 1D-CNN, followed by a softmax layer for classification.

2D-CNNs are used in spatial data analysis, such as image classifications [24]. HAST-IDS [25] uses a hybrid model of 2D-CNN and LSTM, where 2D-CNN is used for extracting the features from the images that are constructed using raw data packets, and LSTMs were used for analyzing temporal relationships between the extracted features.

This work proposes a 1D-CNN based Network Flow-based Intrusion Detection Approach to identify network related intrusions. A set of experiments were conducted to measure the detection proficiency of the proposed approach by considering the CSE-CIC-IDS dataset 2018 [17], [26].

III. Proposed Method

The proposed approach consists of training and testing phases, as shown in Figure 1. Each phase consists of 3 stages: 1) Network flow preprocessing, 2) Autoencoder-based dimensionality reduction, and 3) Classifier that classifies the given test instance as benign or attack.

A. Data preprocessing

Raw Feature Vector consists of the features of mixed type (Categorical and Numerical). The Feature Segregator segregates the features of the Raw Feature Vector into Categorical Features and Numerical Features. Then, the Categorical Features are encoded into the binary format as per the steps shown in the Algorithm 1 using Binary Encoder.

Numerical features are also converted into uniform range due to the reason that range of the feature values is broad. In order to reduce their range variation, Log transform is applied to each of the Numerical Features independently $\mathcal{F}$ using Equation 1 [27]. It is assumed that $\mathcal{F}.values$ contains the list of the values obtained the feature $\mathcal{F}$ in the feature vector.

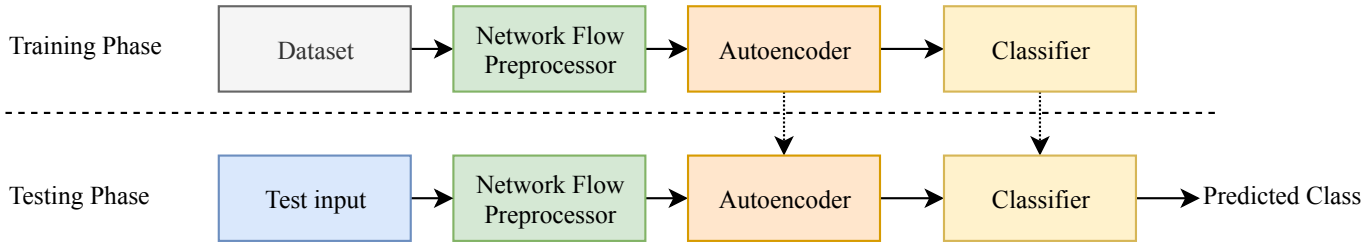

Fig. 1. Components of the proposed approach

$$\mathcal{F}_t.values = \begin{cases} \ln(\mathcal{F}.values), & \text{if } m > 0 \\ \ln(\mathcal{F}.values + |m| + \epsilon) & \text{if } m \leq 0 \end{cases} \quad (1)$$

Where $m = \min(\mathcal{F}.values)$ and ϵ is a small positive number. The addition and function application operations are done element-wise. Further, in order to make the Numerical Features comparable, features are then normalised using Z-score normalisation (Equation 2) [28].

$$\mathcal{F}_n.values = \frac{\mathcal{F}_t.values - \mu(\mathcal{F}_t.values)}{\sigma(\mathcal{F}_t.values)} \quad (2)$$

Where $\mu(.)$ and $\sigma(.)$ are the mean and standard deviation of entire feature values, respectively. Also, the subtraction and division operations are applied element-wise. Before this stage, the features having $\sigma(\mathcal{F}_t.values) = 0$ are eliminated. Z-score normalization effectively brings the mean of the features to 0 and their standard deviation to 1. At last, Binary encoded Categorical Features, and Normalised Numerical Features are concatenated to get the final, Preprocessed Feature Vector, as shown in Figure 2. The parameters of binary Encoder and normalizer are re-used in the testing phase.

B. Autoencoder

Transforming higher dimensional data into a lower dimension, while preserving its characteristics is called dimensionality reduction. Autoencoder has been used widely to reduce dimensionality [29]. In this work, a deep autoencoder is employed to perform the dimensionality reduction of the Preprocessed Feature Vector. It consists of encoder and decoder parts, where the Encoder transforms the higher dimension vector to a lower dimension, the decoder reconstructs the higher dimension vector from the encoded vector. Mean Squared Error between the feature vector and the reconstructed feature vector is used to measure the performance of the Autoencoder. L^2 norm penalty is applied to the encoded feature vector, which forces its activations to be closer to 0. While training phase, both the Encoder and decoder part of the Autoencoder are required, whereas, during testing, only its encoder part is used, as shown in Figure 3. The decoder part is the mirror image of the encoder part.

In this work, autoencoder architecture consisting of three dense layers consisting 100, 75 and 50 neurons, respectively with the ReLu activation function is used.

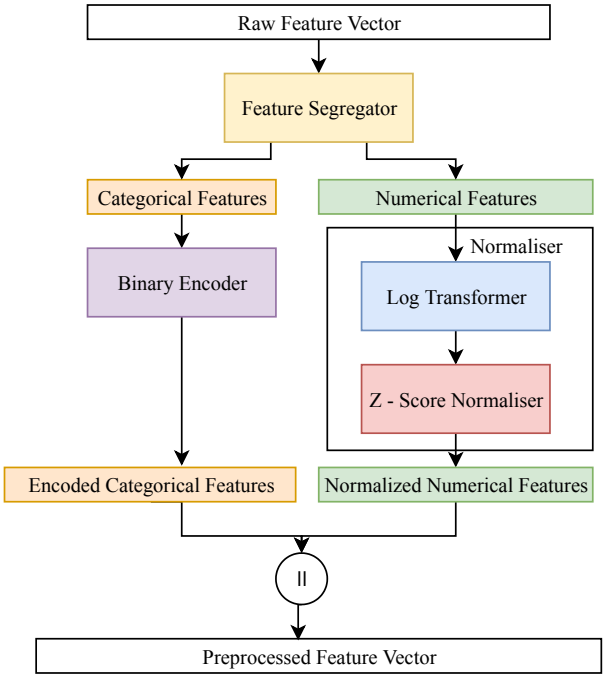

Fig. 2. Flow diagram of Network Flow Preprocessor. The operator || is the concatenation operator.

Algorithm 1: Binary Encoding of a Categorical Feature

Input: A Categorical Feature $\mathcal{F}$
Number of bits to be used encode them n
Output: Set of binary features $\{f_0, f_1, \ldots f_{n-1}\}$
Create n binary encoded empty features $\{f_0, f_1, \ldots f_{n-1}\}$
for $i = 0$ *to len*$(\mathcal{F}.values)$ **do**
  $v \longleftarrow$ bin$(\mathcal{F}.values[i])$
  for $j = 0$ *to* n **do**
    $f_j.values[i] \longleftarrow j^{th}$ bit of v
  end
end

C. Classifier

Random Forest classifier [4] is an ensemble Decision Tree-based classifier. It has been widely used for classification [30], [31] as well as regression [32], [33] tasks. In this work,

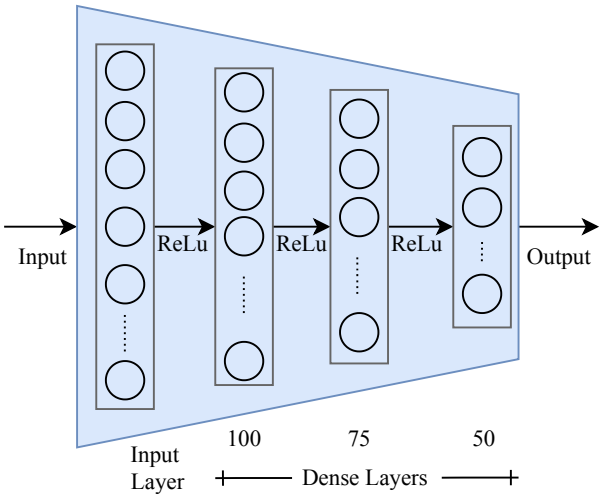

Fig. 3. Architecture of the Encoder Part of the Autoencoder.

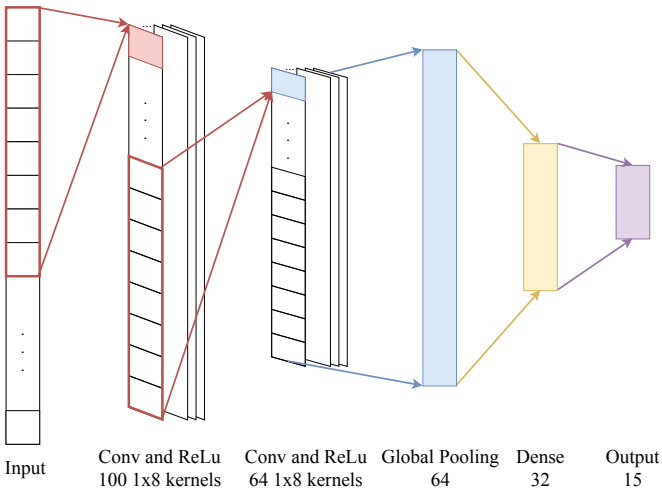

Fig. 4. One Dimensional Convolution Neural Network (1D-CNN) Architecture

Random Forest classifier and 1D-CNN classifier both are used separately to measure the effectiveness of the proposed network flow-based intrusion detection approach. The Random Forest classifier set with 10 Decision Trees and it was trained on the Gini split criterion. Architecture of the 1D-CNN classifier used in this work is as shown in 4. It consists of a number of convolution layers and pooling layers are acting as feature extractors, followed by dense layers to predict the class of input samples. ReLu activation function is used at all hidden layers. Softmax activation function as shown in Equation 3 is used at output units to output the probability distribution over classes of flow.

$$\text{softmax}(\mathbf{x})_i = \frac{e^{\mathbf{x}_i}}{\sum_j e^{\mathbf{x}_j}} \tag{3}$$

Where $\mathbf{x}$ is the activations of the last layer of the neural network. Softmax activation function normalises the input vector so that the sum of its elements is 1, and, thus can be interpreted as probability of the each output class that an input sample belongs to.

IV. Experimental Results and Analysis

A. Experimental Setup

The experiments were conducted on NVIDIA DGX station with 4 Tesla V100 Graphics Processing Units (GPUs). Total GPU memory was 128 GB. The station had Intel Xeon E5-2698 v4 20 Core CPU running at 2.2 GHz. System memory was 256 GB.

The preprocessing step was implemented using pandas [34]. Tensorflow [35] was used to implement 1D-CNN and Autoencoder, whereas scikit-learn [36] was used to build Random Forest classifier.

B. Dataset Description

In this work, CSE-CIC-IDS2018 [17], [26] dataset was used to evaluate the proposed approach. Every instance of the dataset consists of the destination port, protocol numbers, and other statistical parameters. Each of the instances is labeled with the class to which the network flow belongs to. The total number of attack classes (intrusion) in the dataset is fourteen and one benign class. The total number of instances in the dataset is 16232928. The distribution of instances among various classes are described in Table I. The distribution clearly highlights that the dataset is imbalanced.

The dataset was split into training and testing parts such that 70% of the data was used to train the chosen classifier. This resulted in a training split consisting of 12986342 instances and a testing split consisting of 3246586 instances. In order to ensure the same proportion of the classes in both the splits, the stratified sampling technique was used.

TABLE I
Dataset Description

Class Name	Instance count
Benign	13484693
DDOS attack-HOIC	686012
DDoS attacks-LOIC-HTTP	576191
DoS attacks-Hulk	461912
Bot	286191
FTP-BruteForce	193360
SSH-Bruteforce	187589
Infilteration	161934
DoS attacks-SlowHTTPTest	139890
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
DDOS attack-LOIC-UDP	1730
Brute Force -Web	611
Brute Force -XSS	230
SQL Injection	87

TABLE II
EXPERIMENTAL RESULTS

		Accuracy			Precision		Recall		F1 score	
		1D-CNN	RF	Difference	1D-CNN	RF	1D-CNN	RF	1D-CNN	RF
Without Preprocessing	With AE	83.07%	98.20%	15.13%	0.69	0.98	0.83	0.98	0.75	0.98
	Without AE	83.07%	98.24%	15.17%	0.69	0.98	0.83	0.98	0.75	0.98
With Preprocessing	With AE	98.08%	98.21%	0.13%	0.97	0.98	0.98	0.98	0.97	0.98
	Without AE	98.29%	98.25%	0.04%	0.98	0.98	0.98	0.98	0.98	0.98

C. Experimental Description

To compare the effectiveness of the proposed approach, a set of experiments were conducted without preprocessing and autoencoder steps (blocks shown in Figure 1). Elimination of two blocks results in 4 combinations.

1) With preprocessing, with autoencoder
2) With preprocessing, without autoencoder
3) Without preprocessing, with autoencoder
4) Without preprocessing, without autoencodr

Effectiveness of the experiments were measured using the metrics such as Accuracy, Precision, Recall and F1 score [37]. All the metics except Accuracy were calculated using weighted average.

The obtained results of the experiments are shown in Table II.

1) Without preprocessing and without Autoencoder: This set of experiments was used as a baseline for all other experiments. In this experiment, the dataset was provided directly to the classifier without applying network preprocessing and autoencoder-based dimensionality reduction operations. Thus, no encoding, normalization, and no feature elimination were applied. The number of input features was 78.

The Random Forest classifier achieved 98.24% of accuracy. However, 1D-CNN was only able to achieve 83.07% of accuracy, that too, by predicting every testing instance as benign, since the testing dataset contained 83.07% benign instances.

2) Without preprocessing and with Autoencoder: In this set of experiments, Autoencoder was used to reduce the dimension of Raw Feature Vector from 78 features to 50 features. Classification models were trained with the feature vectors produced by Autoencoder.

The accuracy of the 1D-CNN classifier remained at 83.07%, by predicting every input instance as benign. However, the Random Forest classifier was able to achieve 98.20% of accuracy, which was slightly less than the previous experiment. By this experiment, it can be understood that autoencoders also work well on un-normalized data.

3) With preprocessing and without Autoencoder: Data preprocessing was done in this set of experiments. In the dataset, `dst_port` and `protocol` features were of categorical types, which were encoded using 16 bits and 8 bits, respectively using Algorithm 1. Other features were numerical, and for each of the numerical features, Log transformation and Z-score normalization were applied by using Equation 1 and 2, respectively. Before this process, the features `Bwd PSH Flags` ,`Bwd URG Flags` ,`Fwd Byts/b Avg` ,`Fwd Pkts/b Avg` ,`Fwd Blk Rate Avg` `Bwd Byts/b Avg` , `Bwd Pkts/b Avg` and `Bwd Blk Rate Avg` were found to have $\sigma(.) = 0$ and were eliminated. The dimension of the feature vector was 92 after the preprocessing operation.

Both 1D-CNN and Random Forest classifiers worked well and achieved almost similar accuracy of 98.29% and 98.25%, respectively, in this experiment. This proves that the preprocessing step is essential to the 1D-CNN classifier, as it brings all features to a comparable range, thus enabling it to learn the relationship between the features.

4) With preprocessing and with Autoencoder: This was the fourth and final set of experiments conducted. Preprocessing steps were the same as the previous experiments, resulting in Preprocessed Feature Vectors of containing 92 features. Autoencoder was used to reduce the dimension of the Preprocessed Feature Vector to 50 features.

There were no improvements in the accuracy of the classifiers as compared to the previous experiment.

D. Results Discussion

Data preprocessing was proved to be a crucial stage without which the accuracy of the proposed approach (both with and without Autoencoder) remained at 83.07%. Random Forest classifier achieved well as compared to the 1D-CNN-based classifier, surpassing it in Accuracy difference of 15%, even with the Autoencoder produced Feature Vector.

The difference between the accuracy of the proposed approach with the Random Forest classifier has become less once preprocessing of the raw Feature Vector is done. This is due to the reason that all the features were within a similar range, and 1D-CNN was able to learn the relationships between them. The Random Forest classifier also slightly benefited from Preprocessed Feature Vector. However, Autoencoder produced Feature Vector did not to further improve the accuracy of the classifiers.

V. CONCLUSIONS

In this work, the network flow-based intrusion detection approach using 1D-CNN is proposed. Eight sets of experiments were conducted to measure and compare the effectiveness of the proposed approach. Obtained experimental results demonstrate that the proposed network flow preprocessing step is essential to achieve a high detection rate. However, autoencoders produced

feature vector was unable to influence the classifier to achieve a high detection rate. Although the proposed approach with the Random Forest classifiers achieved almost similar performance with the preprocessed feature vector.

REFERENCES

[1] Symantec, "ISTR – internet security threat report," 2019, https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf, last accessed on 2020-01-13.

[2] N. Hubballi and V. Suryanarayanan, "False alarm minimization techniques in signature-based intrusion detection systems: A survey," *Computer Communications*, vol. 49, pp. 1 – 17, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366414001480

[3] A. R. Jakhale, "Design of anomaly packet detection framework by data mining algorithm for network flow," in *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*, June 2017, pp. 1–6.

[4] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[5] P. A. Flach and N. Lachiche, "Naive bayesian classification of structured data," *Machine Learning*, vol. 57, no. 3, pp. 233–269, Dec 2004. [Online]. Available: https://doi.org/10.1023/B:MACH.0000039778.69032.ab

[6] J. K. Jaiswal and R. Samikannu, "Application of random forest algorithm on feature subset selection and classification and regression," in *2017 World Congress on Computing and Communication Technologies (WCCCT)*, 2017, pp. 65–68.

[7] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Comput. Surv.*, vol. 51, no. 3, May 2018. [Online]. Available: https://doi.org/10.1145/3178582

[8] S. Zavrak and M. İskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108 346–108 358, 2020.

[9] J. Ghasemi, J. Esmaily, and R. Moradinezhad, "Intrusion detection system using an optimized kernel extreme learning machine and efficient features," *Sādhanā*, vol. 45, no. 1, p. 2, Dec 2019. [Online]. Available: https://doi.org/10.1007/s12046-019-1230-x

[10] H. Liu, G. Hao, and B. Xing, "Entropy clustering-based granular classifiers for network intrusion detection," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, p. 4, 2020. [Online]. Available: https://doi.org/10.1186/s13638-019-1567-1

[11] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, p. 487–499.

[12] Azeez, Ayemobola, Misra, Maskeliūnas, and Damaševičius, "Network intrusion detection with a hashing based apriori algorithm using hadoop mapreduce," *Computers*, vol. 8, no. 4, p. 86, Dec 2019. [Online]. Available: http://dx.doi.org/10.3390/computers8040086

[13] The UCI KDD Archive, "KDD Cup 1999 data," 2018, https://registry.opendata.aws/cse-cic-ids2018/, last accessed on 2020-01-13.

[14] P. Devan and N. Khare, "An efficient xgboost–dnn-based classification model for network intrusion detection system," *Neural Computing and Applications*, Jan 2020. [Online]. Available: https://doi.org/10.1007/s00521-020-04708-x

[15] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, July 2009, pp. 1–6.

[16] R. Panigrahi and S. Borah, "Dual-stage intrusion detection for class imbalance scenarios," *Computer Fraud & Security*, vol. 2019, no. 12, pp. 12 – 19, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361372319301289

[17] I. Sharafaldin., A. H. Lashkari., and A. A. Ghorbani., "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, INSTICC. SciTePress, 2018, pp. 108–116.

[18] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *2012 IEEE Spoken Language Technology Workshop (SLT)*, Dec 2012, pp. 234–239.

[19] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[20] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *2016 International Conference on Platform Technology and Service (PlatCon)*, Feb 2016, pp. 1–5.

[21] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2017, pp. 1222–1228.

[22] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset," *IEEE Access*, vol. 8, pp. 29 575–29 585, 2020.

[23] G. Andresini, A. Appice, N. D. Mauro, C. Loglisci, and D. Malerba, "Multi-channel deep feature learning for intrusion detection," *IEEE Access*, vol. 8, pp. 53 346–53 359, 2020.

[24] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, Dec 2014, pp. 844–848.

[25] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[26] Registry of Open Data on AWS, "A realistic cyber defense dataset (CSE-CIC-IDS2018)," 2018, https://registry.opendata.aws/cse-cic-ids2018/, last accessed on 2020-01-13.

[27] L. Metcalf and W. Casey, "Chapter 4 - introduction to data analysis," in *Cybersecurity and Applied Mathematics*, L. Metcalf and W. Casey, Eds. Boston: Syngress, 2016, pp. 57 – 58. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B978012804452000004X

[28] J. Han, M. Kamber, and J. Pei, "3 - data preprocessing," in *Data Mining (Third Edition)*, third edition ed., ser. The Morgan Kaufmann Series in Data Management Systems, J. Han, M. Kamber, and J. Pei, Eds. Boston: Morgan Kaufmann, 2012, p. 114–115. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780123814791000034

[29] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232 – 242, 2016, roLoD: Robust Local Descriptors for Computer Vision 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231215017671

[30] V. Rodriguez-Galiano, M. Chica-Olmo, F. Abarca-Hernandez, P. Atkinson, and C. Jeganathan, "Random forest classification of mediterranean land cover using multi-seasonal imagery and multi-seasonal texture," *Remote Sensing of Environment*, vol. 121, pp. 93 – 107, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0034425711004408

[31] M. S. Alam and S. T. Vuong, "Random forest classification for detecting android malware," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, Aug 2013, pp. 663–669.

[32] O. Mutanga, E. Adam, and M. A. Cho, "High density biomass estimation for wetland vegetation using worldview-2 imagery and random forest regression algorithm," *International Journal of Applied Earth Observation and Geoinformation*, vol. 18, pp. 399 – 406, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0303243412000566

[33] C. Lindner, P. A. Bromiley, M. C. Ionita, and T. F. Cootes, "Robust and accurate shape model matching using random forest regression-voting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1862–1874, Sep. 2015.

[34] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python," *arXiv e-prints*, p. arXiv:1907.10121, Jul 2019.

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[37] scikit-learn, "Precision-recall," 2019, https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html, last accessed on 2020-01-13.