Oliveira et al. "A Stacked Ensemble Classifier for an Intrusion Detection System in the Edge of IoT and IIoT Networks" in IEEE Latin-American Conference on Communications (LATINCOM), Rio de Janeiro, 2022, pp. 1-6. https://doi.org/10.1109/LATINCOM56090.2022.10000559

A Stacked Ensemble Classifier for an Intrusion Detection System in the Edge of IoT and IIoT Networks

Giovanni Aparecido da Silva Oliveira[1], Priscila Serra Silva Lima[1], Fabio Kon[1], Routo Terada[1], Daniel Macêdo Batista[1], Roberto Hirata Junior[1], Mosab Hamdam[1]

[1]Computer Science Department, Institute of Mathematics and Statistics, University of São Paulo

Abstract

Over the last three decades, cyberattacks have become a threat to national security. These attacks can compromise Internet of Things (IoT) and Industrial Internet of Things (IIoT) networks and affect society. In this paper, we explore Artificial Intelligence (AI) techniques with Machine and Deep Learning models to improve the performance of an anomalybased Intrusion Detection System (IDS). We use the ensemble classifier method to find the best combination between multiple models of prediction algorithms and to stack the output of these individual models to obtain the final prediction of a new and unique model with better precision. Although, there are many ensemble approaches, finding a suitable ensemble configuration for a given dataset is still challenging. We designed an Artificial Neural Network (ANN) with the Adam optimizer to update all model weights based on training data and achieve the best performance. The result shows that it is possible to use a stacked ensemble classifier to achieve good evaluation metrics. For instance, the average accuracy achieved by one of the proposed models was 99.7%. This result was better than the results obtained by any other individual classifier. All the developed code is publicly available to ensure reproducibility.

Introduction

The Internet of Things (IoT) and Industrial Internet of Things (IIoT) denote a network of connected physical devices to exchange information.

While IoT focuses on devices and services for consumers, smart appliances for the home, virtual assistants, security systems, or wearables that track health, IIoT focuses on devices and services for industry, especially in the manufacturing, energy, water supply, and transport sectors. IoT Analytics predicts that the global number of connected IoT devices will grow by 9% and reach 27 billion IoT connections by 2025 [1].

With the considerable increase in the number of connected devices, security concerns have become a critical point for IoT and IIoT investment; in the case of IIoT devices, the matter is more significant as they affect critical industry structures.

The first widely-publicized attack in IIoT became known as Stuxnet in 2010, a malicious computer worm designed to attack the SCADA operating system developed by Siemens and used to control uranium enrichment centrifuge in Iran. The attack had two goals: force centrifuges to start spinning 40% faster for fifteen minutes, causing aluminum production to crack, and record telemetry data from regular operation, while the centrifuges were tearing themselves apart under Stuxnet attack without the workers knowing. This attack and other cyberattacks in the digital environment could be considered the start of a cyberwar, which could become a worldwide concern [2].

A few years later, other malware attacks, Black Energy in 2015 and Industroyer in 2016, compromised power grids in Ukraine [3]. Currently, the conflict between Russia and Ukraine may be the first war that simultaneously causes tensions in the physical and digital environments, in addition to the armed war. Among the multiple cyberattacks that have taken place against Ukraine, Industroyer 2 stands out for being a variant of the Industroyer malware attack, which compromised power grids again. Also, Distributed Denial-of-Service (DDoS) attacks took down the websites of several government agencies and the two largest state-owned banks; Privatbank and Oschadbank (the State Savings Bank) [4]. Usually, attackers can use vulnerable and infected devices or contract Attack-as-a-Service to initiate an attack [5] and threaten networks' confidentiality, integrity, and availability.

According to Gartner, more than 25% of identified attacks in organizations involve IoT [6]. In addition, the Global Risks Report 2022 from World Economic Forum recognizes the need to adopt mitigation measures against cyberattacks [7]. All these attacks reignite the old discussion about detecting an attack event against IoT and IIoT networks that are crucial for the correct functioning of a city, a state, or even the whole country.

Intrusion Detection Systems (IDS) can be used to monitor a network for malicious activity or policy violations. The IDS differs based on the detection

method, which can be signature-based for detecting known attacks, anomaly-based for detecting a change in the network traffic pattern, or hybrid.

Several works in the literature propose utilizing machine and deep learning models to improve the performance of IDSs. Most of these works present a supervised-learning solution.

We use a ML-based classifier to find an optimal combination between multiple detection algorithms models and stack the output of these individual models to obtain the final detection result of a new and unique model with better classification metrics. Although many ensemble approaches exist, finding a suitable ensemble configuration for multiple attack types is still challenging.

This work proposes a stacked ensemble classifier to be integrated on an anomaly-based IDS for attack detection, with high precision and efficiency, with the selection of a few features from the TON_IoT dataset, one of the most recent IoT attack datasets [8]. The TON_IoT is a dataset containing data collected from a synthetic testbed with seven different sensors and nine types of attacks related to IoT and IIoT networks. The Cloud, Fog, and Edge layers compose the testbed with Software-Defined Network (SDN) and Network Function Virtualization (NFV). This dataset includes data from telemetry, operating system logs, and regular and anomaly network traffic. Our results show that it is possible to use a stacked ensemble classifier to achieve very good evaluation metrics. For instance, the average accuracy achieved by one of the proposed models was 99.7%. This result was better than the results obtained by any other individual classifier. All the developed code is publicly available to ensure reproducibility.

We structured the rest of this paper as follows: Section **??** reviews the related work. Section **??** describes the stacked ensemble classifier. Section **??** describes the pre-processing of TON_IoT dataset and presents the experiment design. Section **??** presents the results of our proposed methodology and compares them to the existing state of the art. Finally, Section **??** discusses the results obtained in the previous section and potential future works to improve the ensemble classifier.

Related Work

Machine and Deep learning are effectively used for network traffic classification and attack detection. They can be applied through different methods such as Naive Bayes (NB), Decision Tree (DT), Extra-Tree (ET), Extreme Gradient Boosting (XGB), Artificial Neural Network (ANN), and Deep Neural Network (DNN) [9]. The qualities of some of these models can be composed by ensemble techniques such as mean combiner, median combiner, max combiner, majority voting, weighed majority voting (WMV), Random Forest (RF) [10], and stacking. However, no standardized approach performs well on all datasets.

Among the works recently developed, Mustafa *et al*. [11] proposed a model with the Adaboost ensemble technique composed of DT, NB, and ANN for detection, where DNS, HTTP, and MQTT protocols were considered to generate new statistical flow features. The results showed that their ensemble technique obtained a more outstanding Detection Rate (DR) and a lower False Positive Rate (FPR) than other models with the UNSW-NB15 [12] and NIMS botnet [13] datasets.

Zhou *et al*. [14] used three different algorithms to compose an ensemble classifier: C4.5, RF, and Forest by Penalizing Attributes (Forest PA) integrated into the Average of Probabilities (AOP) combination rule. They applied the Correlation-Based Feature Selection Bat-Algorithm (CFS-BA) to reduce the data's dimensionality and select the most relevant features. Cross-validation was also used to validate the model's performance, classify the traffic as normal or anomaly, and the types of attacks. The tests were performed on three different datasets: NSL-KDD [15], AWID [16], and CIC-IDS2017 [17], and the best evaluation was achieved with the last one, although the identification of some types of attacks did not reach even 90% due to the low volume of data related to the attacks.

Priya *et al*. [18] used a stacked ensemble to train a learning model in two phases. The first one integrates the Support-Vector Machine (SVM), NB, and DT as base classifiers and sends the result to the second one, which integrates the classifiers in RF and ANN for detection using the softmax activation function. The K-fold cross-validation is used to try to obtain optimized training and test sets. Finally, the proposed model is tested on standard IoT attack datasets such as WUSTL_IIOT-2018 [19], N_BaIoT [20], and Bot_IoT [21], getting the highest accuracy of 99% on attack activity detection. NB presented the worst result in the three datasets tested. In this case, no specific technique was used to reduce dimensionality and select the features.

M. Rashid *et al*. [22] used the ensemble method with three tree-based algorithms: DT, RF, and XGB with the datasets NSL-KDD and UNSW-NB15. The features were selected using the sklearn's `selectbest` method. They also used a cross-validation approach to validate the model's performance and classify traffic as normal or anomalous. Performance metrics got better results with the first dataset, reaching 99% of accuracy.

In this paper, we propose an anomaly detection model based on a stacked ensemble classifier selecting only a few features without degrading the performance of traffic classification in terms of accuracy, precision, recall, and F1-score. The few selected features have the potential properties to achieve results above 99% in all the evaluated metrics, reducing the complexity of training the model considering the time and operational cost, as it significantly minimizes the redundant and irrelevant data that compose a high-dimensional dataset and that can interfere in the classification process. Furthermore, the application layer data, which is sensitive to privacy, was also discarded, as well as statistical flow capabilities. In this case, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) outperformed the other algorithms tested, and both approaches are cited in several papers and considered more accurate. However, both have not been considered in the works related to the ensemble classifiers mentioned above.

Proposed Approach

We propose a stacked ensemble classifier to detect malicious network packets at the Edge of an IoT/IIoT environment. Each packet flowing in the network access point (Fig. 1.b) is analyzed by distinct machine learning models (Fig. 1.c), producing partial results. Another model (Fig. 1.d) uses these partial results to produce a final detection result. If the packet is detected as an attack, we discard it and do not forward it to its destination (It is also possible to log these discarded packets and/or warn the system administrator).

If it is possible to run all base classifiers in parallel, then the total classification time (T_{total}) will be the sum of the maximum time among all base classifiers (T_{model}) and the classification time of the ensemble model (T_{ensemble}):

$$T_{\text{total}} = \max_{\text{model} \in \text{base}} \{T_{\text{model}}\} + T_{\text{ensemble}} \qquad (1)$$

We have chosen ten machine learning models as base classifiers because of their success in classifying attacks, being known by the community, and their availability on open source platforms. The main idea behind the classifier we propose in this work is to use heterogeneous machine-learning models to classify the same network packet and an ensemble model to combine the results produced by the base classifiers. Though the performance variates considerably among the base models, the idea is to have an optimal machine-learning model to ensemble the results according to their performance during the training phase. Therefore, we tested and compared

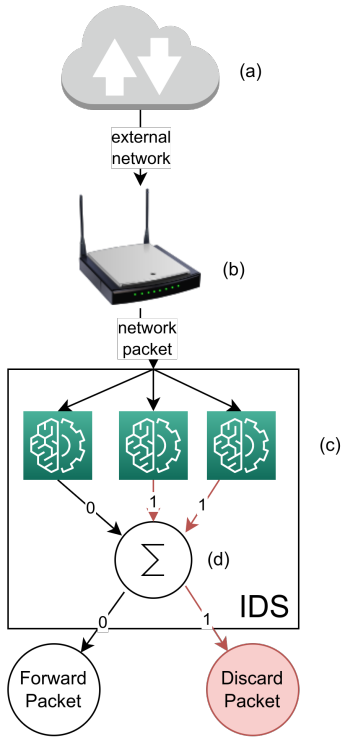

Figure 1: *Overview of the proposed solution for an IDS based on a stacked ensemble classifier.*

two models for the ensemble classifier: a 4-layered deep neural network and the Random Forests. We present their characteristics in the following items.

- **Bidirectional Gated Recurrent Unit Recurrent Neural Network (B-GRU-RNN):** sequence information in both directions, forward and backward. GRU has two gates, reset and update, that modulate the data flow inside the unit. It uses less memory and is faster than LSTM; however, the LSTM is more accurate when using datasets with longer sequences [23].
- **Bidirectional Long Short-Term Memory Recurrent Neural Network (B-LSTM-RNN):** sequence information in both directions, forward or backward. LSTM has three gates, input, output, and forget, that modulate the data flow inside the unit [24].
- **Random Forest (RF):** ensemble algorithm for classification and regression that uses bagging to build trees and combine the parallel outputs to reduce overfitting. For classification, the output is the class selected by most trees, and for regression, the mean or average prediction of the individual trees [25]
- **Gradient Boost Tree (GBT):** similar to RF. It differs in the build of the individual trees and how the outputs are combined. It uses boosting and combines decision trees with only one split se-

quentially to each new tree correcting the errors of the previous one [26].

- **K-Nearest Neighbours (KNN):** algorithm for classification and regression that does not require training. All data obtained from the dataset is used in the test phase, resulting in high-speed training and slow testing and validation. In addition, it has a similarity of resources to predict values of any new data points [27].
- **Deep Neural Network (DNN):** feedforward network in which data flow from the input to the output layer without looping back. It has neurons and assigns random numerical values, or "weights" to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If it cannot accurately recognize a particular pattern, an algorithm will adjust the weights. That way, the algorithm can make specific parameters more influential until it determines the correct mathematical manipulation to process the data fully [28].
- **Extreme Gradient Boosting (XGB):** similar to GBT but uses advanced regularization (L1 and L2), which improves model generalization capabilities. High-performance training in parallel or distributed across clusters [29].
- **Multi-Layer Perceptron (MLP):** consists of at least three layers of nodes; an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. It uses a technique called backpropagation for training [30].
- **Support Vector Machine (SVM):** a supervised machine learning method based on nonlinear optimization to find a group of special points called support vectors that define the decision classification frontiers [31].
- **Naive Bayes (NB):** a probabilistic algorithm for classification based on probability models that incorporate strong independence assumptions [31].

We used the RF and DNN base classifiers to combine the submodels' results and produce a final detection outcome. We expect the ensemble classifier to present better metrics (accuracy, precision, recall, and F1-score) than the best submodels' metrics.

Experimental Design

Dataset preprocessing

We used the TON_IoT dataset to train and test our classifier against nine attack types present in the dataset: DoS, DDoS, Ransomware, Backdoor, Brute Force, Cross-site Scripting (XSS), Injection, Man-In-The-Middle (MITM), and Scanning.

The attackers performed their attacks at the Fog and the Edge of the testbed used to create the dataset. However, the solution presented in this paper claims to be equally valid in preventing system invasions directly in the Edge layer (e.g., Wifi Cracking [32]) as shown in the example of Fig. 2.

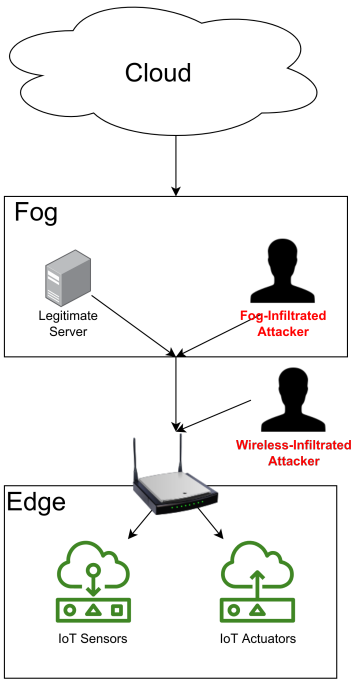

Figure 2: *Use case for the IDS in a Cloud-Fog-Edge environment.*

We started with the features from the Train-Test data subsets provided by the dataset's authors. There are six feature categories: connection, statistical, DNS, SSL, HTTP, and violation. The selected subset is balanced among the nine attack types containing 20,000 records each, except for MITM containing 1,043 records, totaling 161,043 attack records. The selected subset also has 300,000 non-attack records extracted from the regular operation of the testbed. We produce a binary classification problem of the network packets in the *attack* and *non-attack* classes. For this reason, we label the records with a Boolean field.

We adopted a minimalist approach for the feature selection to minimize the processing time for a given network packet. Therefore, we didn't used the derivated and application-layer features, remaining only the connection features. Removing the derivated features from the connection category left us with: timestamp, source/destination IP/port, and transport-layer protocol. To provide a timestamp and IP-independent solution, we removed the re-

lated features and ended up with our three features: source port, destination port, and transport-layer protocol. These features have a well-defined finite domain which is: $\mathbb{Z} \in [0, 65535]$ for the port numbers and [tcp, udp, none] for the transport protocol.

The last step of data preprocessing is to apply the standard scaler to the data frame. The feature values are linearly scaled to present a mean equal to 0 and a standard deviation of 1.

Implementation

We implemented all classifiers presented in Section **??** using 3rd-party libraries in Python 3. Table 1 describes the libraries and the non-default parameters used for each model. The source code for all classifiers is available in our public repository[1].

Train-test splitting

We generated random dataset partitions with 40% (Train I), 30% (Train II), and 30% (Test) of the total mass for training the base classifiers, training the ensemble layer, and testing, respectively.

We generated ten distinct random partitions to obtain the confidence interval for the results presented in Section **??**. Each experiment was executed in a different new runtime environment in Google Colab. The only change between the executions is the seed for the pseudo-random engine that partitions the dataset.

Execution

We divided the execution process into three phases. In the first phase, we trained the base classifiers with the Train I subset. Then, we used the trained models to detect the packets in the Train II and Test subsets. These detection results are used as input for the second and third phases. We identify the most well-performing model based on the detection results of the Train II and Test subsets. They are used as a reference for performance comparison with the ensemble classifiers.

In the second phase, we used the initial detection results as input for the ensemble classifiers. The reference model is also trained with the Train II subset.

After having the ensemble and reference models trained, we used them to obtain the records from the Test subset. We used these results to do the overall analysis.

Table 1: *Implementation details of the models used in the experiment. Abbreviations: K.r.: Kernel regularization; Bidir.: Bidirectional layer; n.: neurons in layer; a.: layers' activation function; BGR: B-GRU-RNN; BLR: B-LSTM-RNN; TensorFlow: TF; DF: Decision Forests; SKL: SKLearn; XGB: XGBoost.*

Model	Library	Custom Attributes
BGR	TF 2.8.2 Keras 2.8.0	K.r. L2 in all layers; Adam optimization; Binary cross-entropy loss; Bidir. GRU: 64 n., a. tanh; Dense: 128 n., a. relu; Dense: 1 n., a. sigmoid; Window size = 100.
BLR	TF 2.8.2 Keras 2.8.0	K.r. L2 in all layers; Adam optimization; Binary cross-entropy loss; Bidir. LSTM: 64 n., a. tanh; Dense: 128 n., a. relu; Dense: 1 n., a. sigmoid; Window size = 100.
RF	TF 2.8.2 DF 0.2.6	Using default arguments.
GBT	TF 2.8.2 DF 0.2.6	Using default arguments.
KNN	SKL 1.0.2	Using default arguments.
DNN	TF 2.8.2 Keras 2.8.0	K.r. L2 in all layers; Adam optimization; Binary cross-entropy loss; 4x Dense: 128 n., a. relu; Dense: 1 n., a. sigmoid.
XGB	XGB 0.90	Using default arguments.
MLP	TF 2.8.2 Keras 2.8.0	K.r. L2 in all layers; Adam optimization; Binary cross-entropy loss; 2x Dense: 128 n., a. relu; Dense: 1 n., a. sigmoid.
SVM	SKL 1.0.2	Using default arguments.
NB	SKL 1.0.2	Using default arguments of Gaussian variant.

[1] `https://github.com/giovannioliveira/ids`

Results and Discussion

We evaluated two ensemble classifiers: SE-DNN combines the base classifiers using a Deep Neural Network and SE-RF combines the base classifiers using Random Forest.

We trained and tested each base classifier using the Train I subset and we evaluated the models using the standard classification metrics (accuracy, precision, recall and F1-score). Table 2 presents the results.

Table 2: *Classification metrics for the base-classifiers in percentage (Best results are highlighted in bold).*

Model	Accuracy	Precision	Recall	F1
BGR	**98.62**	**99.68**	98.20	**98.93**
BLR	98.29	98.72	**98.65**	98.69
RF	96.30	96.36	98.01	97.18
GBT	96.29	96.34	98.02	97.17
KNN	95.79	96.19	97.38	96.78
DNN	93.71	95.20	95.12	95.16
XGB	93.59	95.37	94.74	95.06
MLP	92.89	95.46	93.52	94.48
SVM	72.34	82.91	72.40	77.30
NB	73.04	92.17	64.00	75.54

It is possible to observe that B-GRU-RNN is the best base-classifier. It obtained the highest metrics, except for the Recall where B-LSTM-RNN presented a slight advantage of 0.45%. Therefore, we chose B-GRU-RNN as the baseline model for the comparison with the ensemble classifiers.

Fig. 3 presents the overall performance metrics obtained by each final model (SE-DNN, SE-RF, and B-GRU-RNN). The results were taken as the arithmetic mean of the ten runs' results. All the confidence intervals of 95% are smaller than 10^{-4}. For this reason, we omitted them in the representation.

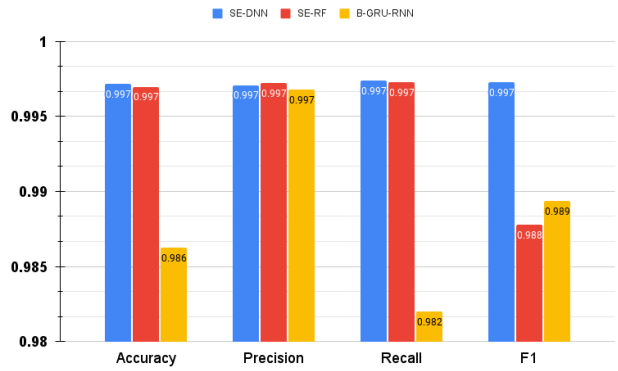

Figure 3: *Classification metrics for the ensemble classifiers (SE-DNN and SE-RF) and the baseline for comparison (B-GRU-RNN).*

The accuracy of the ensemble models is approximately 1.1% greater than the baseline. The precision presents similar values. The recall of the ensemble models is 1.5% greater than the baseline. We take the F1-score as a tiebreaker. For this metric, the highest value of 99.7% was obtained for the SE-DNN model, 0.8% greater than the baseline. The SE-RF, though had a F1-score slightly lower than the baseline.

Conclusion

We conclude our overall analysis by identifying the SE-DNN as the best model among the considered ones. This model has gains varying from 0.8% to 1.5% when compared to the Accuracy, Recall, and F1-score presented by the reference classifier.

We believe that the analysis of the raw network records (pcaps) from the integral dataset could result in the inclusion of new features to increase the classifiers' performance, respecting the same feature-selection restrictions used in this paper. Using the whole dataset would also increase the statistical reliability of the results.

Because the dataset also presents records of IoT/IIoT devices' telemetry and servers' operating system logs, we can to explore federated learning techniques to use this data to increase the classification metrics.

We conjecture that the ensemble model proposed as a solution in this paper could be used to classify other datasets, such as the one used in [33]. A future work will classify records from different datasets using both proposed models and compare their performance using a set of high-performance reference models and models from other authors.

Acknowledgement

This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior–Brasil (CAPES)–Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9. It is also part of the FAPESP proc. 2021/10234-5.

References

[1] S. Sinha. *State of IoT 2021: Number of connected IoT devices growing 9% to 12.3 billion globally, cellular IoT now surpassing 2 billion*. [Online; accessed 2022-05-05]. 2021. URL: `https://iot-analytics.com/number-connected-iot-devices/`.

[2] David Kushner. "The real story of stuxnet". In: *ieee Spectrum* 50.3 (2013), pp. 48–53.

[3] Defense Use Case. "Analysis of the cyber attack on the Ukrainian power grid". In: *Electricity Information Sharing and Analysis Center (E-ISAC)* 388 (2016), pp. 1–29.

[4] CERT-UA. *Information on cyberattacks on February 15, 2022*. [Online; accessed 2022-05-21]. 2022. URL: https://cert.gov.ua/article/37139.

[5] Max Goncharov. "Russian underground 101". In: *Trend Micro incorporated research paper* (2012), p. 51.

[6] M. Hung. *Leading the Iot: Gartner Insights on How to Lead in a Connected World*. [Online; accessed 2022-05-05]. 2017. URL: https://www.gartner.com/imagesrv/books/iot/iotEbook%5C_digital.pdf.

[7] World Economic Forum. *Global Risks Report 2022*. [Online; accessed 2022-05-05]. 2022. URL: https://www.weforum.org/reports/global-risks-report-2022/digest.

[8] Abdullah Alsaedi et al. "TONIoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems". In: *IEEE Access* 8 (2020), pp. 165130–165150. DOI: 10.1109/ACCESS.2020.3022862.

[9] Bo Dong and Xue Wang. "Comparison deep learning method to traditional methods using for network intrusion detection". In: *2016 8th IEEE international conference on communication software and networks (ICCSN)*. IEEE. 2016, pp. 581–585.

[10] Kazi Abu Taher, Billal Mohammed Yasin Jisan, and Md Mahbubur Rahman. "Network intrusion detection using supervised machine learning technique with feature selection". In: *2019 International conference on robotics, electrical and signal processing techniques (ICREST)*. IEEE. 2019, pp. 643–646.

[11] Nour Moustafa, Benjamin Turnbull, and Kim-Kwang Raymond Choo. "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things". In: *IEEE Internet of Things Journal* 6.3 (2018), pp. 4815–4830.

[12] Nour Moustafa and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". In: *2015 military communications and information systems conference (MilCIS)*. IEEE. 2015, pp. 1–6.

[13] *The-NIMS-Dataset*. [Online; accessed 2022-05-05]. 2022. URL: https://projects.%20cs.dal.ca/projectx/Download.html.

[14] Yuyang Zhou et al. "Building an efficient intrusion detection system based on feature selection and ensemble classifier". In: *Computer networks* 174 (2020), p. 107247.

[15] Mahbod Tavallaee et al. "A detailed analysis of the KDD CUP 99 data set". In: *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee. 2009, pp. 1–6.

[16] Constantinos Kolias et al. "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset". In: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 184–208.

[17] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. "Toward generating a new intrusion detection dataset and intrusion traffic characterization." In: *ICISSp* 1 (2018), pp. 108–116.

[18] V Priya et al. "Robust attack detection approach for IIoT using ensemble classifier". In: *arXiv preprint arXiv:2102.01515* (2021).

[19] Sahar Aldhaheri et al. "Deepdca: novel network-based detection of iot attacks using artificial immune system". In: *Applied Sciences* 10.6 (2020), p. 1909.

[20] AM Chandrasekhar and K Raghuveer. "Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers". In: *2013 International Conference on Computer Communication and Informatics*. IEEE. 2013, pp. 1–7.

[21] Nickolaos Koroniotis et al. "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset". In: *Future Generation Computer Systems* 100 (2019), pp. 779–796.

[22] Mamunur Rashid et al. "A tree-based stacking ensemble technique with feature selection for network intrusion detection". In: *Applied Intelligence* (2022), pp. 1–14.

[23] Mirco Ravanelli et al. "Light gated recurrent units for speech recognition". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 2.2 (2018), pp. 92–102.

[24] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[25] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[26] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[27] Padraig Cunningham and Sarah Jane Delany. "K-nearest neighbour classifiers-a tutorial". In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–25.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[29] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[30] Fabrice Rossi and Brieuc Conan-Guez. "Functional multi-layer perceptron: a non-linear tool for functional data analysis". In: *Neural networks* 18.1 (2005), pp. 45–60.

[31] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[32] Stefan Viehböck. "Brute forcing wi-fi protected setup". In: *Wi-Fi Protected Setup* 9 (2011).

[33] Guilherme Werneck de Oliveira et al. "Predicting Response Time in SDN-Fog Environments for IIoT Applications". In: *2021 IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE. 2021, pp. 1–6.