Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 1143–1148
DOI: 10.15439/2023F6021
ISSN 2300-5963 ACSIS, Vol. 35

Incident Detection with Pruned Residual Multilayer Perceptron Networks

Mohamad Soubra
0000-0002-5195-9540
AGH University
Faculty of Computer Science
Electronics and Telecommunications
Krakow, Poland
Email: soubra@agh.edu.pl

Marek-Kisiel Dorohinicki
0000-0002-8459-1877
AGH University
Faculty of Computer Science
Electronics and Telecommunications
Krakow, Poland
Email: doroh@agh.edu.pl

Marcin Kurdziel
0000-0003-2022-7424
AGH University
Faculty of Computer Science
Electronics and Telecommunications
Krakow, Poland
Email: kurdziel@agh.edu.pl

Marek Zachara
0000-0002-8560-9696
AGH University
Faculty of Electrical Engineering,
Automatics, Computer Science
and Biomedical Engineering
Krakow, Poland
Email: mzachara@agh.edu.pl

***Abstract*—Internet of things (IoT) has opened new horizons in connecting all sorts of devices to the internet. However, continuous demand for connectivity increases the cybersecurity risks, rendering IoT devices more prone to cyberattacks. At the same time, rapid advances in Deep Learning (DL)-based algorithms provide state-of-the-art results in many classification tasks, including classification of network traffic or system logs. That said, deep learning algorithms are considered computationally expensive as they require substantial processing and storage capacity. Sadly, IoT devices have limited resources, making renowned DL models hard to implement in this environment. In this paper we present a Residual Neural Network inspired DL-based Intrusion Detection System (IDS) that incorporates weight pruning to make the model more compact in size and resource consumption. Additionally, the proposed system leverages feature selection algorithms to reduce the feature-space size. The model was trained on the NSL-KDD dataset benchmark. Experimental results show that the proposed system is effective, being able to classify network traffic with an F_1 score of up to 98.9% before the pruning and an F_1 score of up to 97.5% after pruning 90% of network weights.**

I. Introduction

INTERNET of Things (IoT) is booming in markets, driving efforts for increasing device inter-connectivity. However, this strive for increased connectivity poses requirements related to provision of security protocols and measures that would secure communication between devices and build trust in users that their data is communicated privately [1]. In order to meet these requirements current security solutions typically endorse defense in depth approach [2] in which the security layers span across network perimeter, intranet and endpoint systems. Such security mechanisms involve many attack detection and prevention technologies. One of the most important class of these technologies, namely Intrusion Detection Systems (IDS) [3], come in various flavors. Host-IDS examines the actions of the users and compares them to decide which actions can be considered as malevolent and which are likely benign. On the other hand, Network-based IDS, examines the traffic traversing through the network and compares it with already known signatures to distinguish between normal and malevolent flow. Though popular, these systems still face various challenges, such as detection accuracy, high false-alarm rates or the inability to detect zero-day attacks [4].

Machine Learning (ML) and Deep Learning (DL)-based technologies recently enjoy numerous practical deployments, e.g., in speech recognition, object detection, natural language processing, etc. It is also increasingly used in the cybersecurity domain [5][6]. Consequently, ML- and DL-based IDS gained popularity in the recent years. In particular, they have proven to be more robust than their predecessors, having lower false-positive rates and higher accuracy [7]. However, this line of research often adopted renowned image classification algorithms [8] to the traffic classification tasks [9][10][11]. Consequently, the proposed systems tend to be computationally cumbersome. Accordingly, for IoT devices, which have limited storage and processing resources, research increasingly focuses on replacing such burdensome algorithms with much lighter solutions.

In this paper, we introduce a new DL-based IDS designed around lightweight residual network [12] architectures. Our solution is coupled with the Extra Tree classification algorithm, which allows us to extract the most important features from the dataset. This makes the proposed system compact, while retaining high accuracy and detection rates. The small

computational footprint of the proposed system is suitable for inference on a CPU, instead of resource-hungry GPU accelerators. Thus, our results show that attaining high accuracy while substantially reducing the size of the model is achievable in IDS tasks.

The following sections begin with review of the state-of-the-art results in ML-based intrusion detection systems. Next, we present the proposed attack detection architecture. Subsequently we describe the experimental setup and report obtained results. Finally, we give conclusions from experiments and outline future work.

II. Related Work

Deep Learning-based intrusion detection systems enjoyed rapid advances in recent years. Some researchers utilized DL capabilities for categorical data classification, where the task is to recognize specific attack instances. Haddad Pajouh et al. [13] proposed a Long-Short Term Memory (LSTM)-based IDS. First, they extracted OpCodes from the traffic and assigned them to input vectors. Next, they leveraged Principal Component Analysis (PCA) to extract the most significant features from the vectors. The model was trained using Adam optimizer [14]. Dropout layers were used to avoid overfitting. The performance of the model was evaluated with 10-fold Cross Validation (CV). Swarna Priya et al. [15] proposed a DNN-based IDS that, similarly to Haddad Pajouh et al. approach, also used PCA as a feature extractor. The system also utilized feature scaling to normalize the input data before feeding it to the classifier. Furthermore, they used Grey Wolf optimization algorithm (GMO) [16] to construct a feature hierarchy. This hierarchy provided features' fitness values. McDermott et al. [17] proposed a Bidirectional LSTM-based IDS. Word embeddings were used to embed the captured packets' content in a vector space suitable for the model. Subsequently, they used word embeddings to establish a dictionary of tokenized words. Sigmoid function, Mean Absolute Error (MSE) and Adam were selected as the activation function, loss function and optimizer, respectively. Zhang et. al [18] proposed a Deep-Belief Network (DBN)-based IDS that employed an improved genetic algorithm. The algorithm incorporated improved crossover and elite retention strategies to prevent the loss of the best individuals. The proposed system was trained and evaluated on the NSL-KDD dataset. Another DBN-based IDS was proposed by Tama et al. [19]. The system incorporated a grid search strategy to select the most significant input features. Evaluation was carried out on three datasets, namely, UNSW-NB15 [20], CIDDS-001 [21], and GPRS [22] using 10-folds cross validation, Repeated Cross-Validation (RepCV) [23] and data sub-sampling. Their model was able to maintain the same detection rate after sub-sampling. Overfitting was prevented with L1 and L2 regulations and an adaptive learning rate. Muna et al. [24] proposed a Deep Autoencoder to reduce the features dimensionality. Their system also encompassed a deep feed-forward Neural Network to detect and classify traffic. It was trained and evaluated on the NSL-KDD dataset. Latif et al. [25] emphasized the importance of providing lightweight DL-based IDS solutions. To this end, they proposed an intrusion detection algorithm employing random neural networks, in which the Poisson distribution was used to estimate the probability of the signals that made the neurons either active or inhibited. The proposed system was evaluated on the DS2OS dataset [26]. Shone et al. [27] proposed a Non-Symmetric Deep Auto-Encoder for unsupervised feature learning. The system employed Random Forest [28] to classify the traffic between benign and malevolent. Both NSL-KDD and KDD Cup '99 datasets were used in training and evaluation. Min et al. [29] proposed a system which uses an ensemble of byte-level word embeddings and text convolutional neural networks. Skip-Gram algorithms was used to create the byte-level word embeddings. Text convolutional neural networks were constructed from one-dimensional convolutions that extracted word-based features. Similarly to Shone et al., Random Forest was chosen as a classifier. The system was trained and evaluated on the ISCX2012 dataset [30]. Zhou et al. [31] proposed Deep Feature Embedding Learning method that reduces input features' dimensionality, thereby decreasing the time needed to train the model. They trained and evaluated their model on the NSL-KDD and UNSW-NB15 datasets. Leaky ReLU was chosen as the activation function for the hidden layers, while Sigmoid function was used as an activation function in the classification layer. Additionally, Dropout was used to avoid overfitting.

Other researchers choose to use deep learning for binary classification, where the goal is to distinguish attack signatures from normal traffic, irrespective of specific attack classes. Diro et al. [32] proposed a DL-based IDS trained in a distributed optimization scheme which involved fog nodes, i.e. mini-clouds implemented as edge devices in the cloud [33]. To avoid overfitting, the parameters were collected in the fog coordinator, which was responsible for their updating and distribution for subsequent epochs. Diro et al. [32] evaluated their system on the NSL-KDD dataset [34]. Similarly, Abeshu et al. [35] proposed a novel DL-based IDS that takes its parameters from the master fog node, while performing system fine-tuning on the worker nodes. Again, NSL-KDD was chosen as training and evaluation dataset. Almiani et al. [36] proposed an RNN-based IDS. Their system employed data oversampling to balance the minority classes, a modified back-propagation algorithm, and the min-max normalization. Kasongo et al. [37] proposed a feed-forward Neural Network-based IDS that was coupled with a wrapper-based feature extraction unit. The wrapper used the Extra Tree algorithm to classify and specify which features are most significant. The proposed system was trained and evaluated on the NSL-KDD dataset. Devan et al. [38] proposed an XGboost DL-based IDS composed of three main steps, namely, input feature normalization, feature selection using a classifier based on a collection of decision trees that derive the significant features, and final classification. Their system also leveraged neural networks with ReLU and Softmax activation functions for the hidden and classification layers, respectively. Nagisetty et al. [39] proposed a DL-based IDS that incorporated three

DL architectures: Multi Layer Perceptrons (MLP), CNNs, and an Autoencoders. The proposed system was trained and evaluated on two datasets, namely, UNSW-NB15 and NSL-KDD99. The system employed Root Mean Square Root Error (RMSE) as the cost function. DNN was used mainly to sort the features and create a feature hierarchy. Zhihan et al. [40] proposed a hierarchical Supporting Vector Machine-based IDS. In addition, a stacked autoencoder was used to denoise the data. The system was evaluated on the NSL-KDD dataset.

In this paper we will benchmark our results with the papers that focus on the binary classification task. To this end, we will evaluate our algorithm with respect to the metrics that they have discussed in their papers as our goal is to see if our pruned networks could compete with the state-of-the-art.

III. Proposed Architecture

In order to protect devices from attacks, while preserving processing and storage resources, we propose an Intrusion Detection System based on pruned residual neural networks [12]. The proposed system is trained in several steps. First, input data is pre-processed, including encoding of symbolic features. The data is then fed to an Extra Tree Classifier [41], which selects the most important features from the feature set. In the next step, the data is normalized and used to train the proposed classification model. Finally, the model is pruned in fine-tuning steps, which minimizes its size and the inference cost.

We evaluate the final model with respect to precision, recall and F_1 score both before and after network pruning. Evaluation is carried on the NSL-KDD dataset.

A. NSL-KDD Dataset

The NSL-KDD dataset is the successor of the KDD'99 [42] dataset, which was introduced by DARPA in 1998. The dataset was firstly proposed by Tavallaee et al. [43] and is composed of 4 different attack classes, namely, Denial of Service (DoS), Probe, User-to-root (U2R), and Remote-to-Local (R2L). In DoS attacks the computing or network resources are exhausted, making the attacked system unable to serve the user's requests. Signatures of a DoS attack in the NSL-KDD dataset would be, e.g., the `Src_byte` and the `Wrong_fragment` features. Probe attacks are mostly used for surveillance, in order to to gain information on the potential victim system. The relevant signatures for probe attacks in the NSL-KDD dataset are the `Src_bytes` and the `Duration` features. User-to-root attacks attempt to grant superuser privileges to the attacker. One way of doing this is accessing the user's system via a normal account and then attempting to escalade privileges by exploiting a vulnerability. Relevant signatures for U2R attacks with respect to the NSL-KDD dataset are, e.g., `Num_file_creations` and `Num_shells` features. In Remote-to-Local attacks the attacker attempts to gain access of the user's system via a remote machine. Relevant signatures for R2L attacks in the NSL-KDD dataset are, e.g., `Duration`, `Service` and `Num_failed_logins` features.

TABLE I: NSL-KDD traffic statistics.

	NSL-KDD		
	Attack Type	***KDDtrain+***	***KDDTest***
1	DOS	45926	7458
2	Probe	11655	2421
3	R2L	995	2754
4	U2R	52	200
5	Normal	67345	9711
Total		125973	22544

The NSL-KDD dataset encompasses two subsets, namely, the KDDtrain+ and KDDtest. In standard classification setup, the proposed system should assign the signatures into four major categories, namely, `DOS`, `Probe`, `R2L`, `U2R`, and `Normal` traffic. Table I reports datasets statistics for these categories. Note that `DOS`, `Probe`, `R2L`, `U2R` and normal traffic makes 36.45%, 9.25%, 0.78%, 0.04% and 45.52% of the dataset instances, respectively. In binary classification setup the proposed system should be able to classify the traffic into two classes, namely, attack and non-attack. Note that classes in this case are balanced, with attack and non-attack traffic making 46.5% and 53.5% of the dataset, respectively.

The NSL-KDD consists of a total of 41 features that comes in four main categories: (a) intrinsic features that can be extracted from the packet's headers, (b) content features which reflect the data content of the packets, (c) time-based features which reflect the connection rates with the hosts, and finally (d) the host-based features. It is also worth mentioning that the KDDtrain+ subset has 3 categorical features, namely:

- `Protocol Type` which consists of 3 categories,
- `Services` which consists of 70 categories,
- `Flag` which consists of 11 main categories.

These features require preprocessing into one-hot encoding before they can be used in the subsequent steps.

B. Data Preprocessing

In this work we focus on a binary classification task, i.e., distinguishing normal network traffic from attacks. We therefore convert the provided labels into attack and non-attack classes before selecting the important features. Next, we remove data duplicates and rows that contain null values. The KDDtrain+ subset consists of both numerical and categorical data. We normalize the numerical features via z-scores:

$$Z = \frac{x - \mu}{\sigma} \tag{1}$$

where x represent the current instance of the feature while μ and σ represent the mean and the standard deviation of the feature respectively. The categorical features, on the other hand, are encoded in one-hot vectors.

In the next step we use Extra Tree classifier with 100 estimators (trees) to identify the most significant features. Specifically, we use the Gini coefficients [44] returned by the Extra Tree classifiers to select the most prominent features. Importantly, for one-hot-encoded features, the feature is retained if the Extra Tree classifier selects any of its dimensions

according to the Gini coefficient. After a series of features selection iterations, the features listed in Table II were used to train the neural network for the binary classification task.

TABLE II: Features selected by the Extra Tree classifier.

Index	NSL-KDD Features	Index	NSL-KDD Features
1	count	12	protocol_type
2	same_srv_rate	13	logged_in
3	dst_host_count	14	rerror_rate
4	dst_host_same_srv_rate	15	same_srv_rate
5	dst_host_serror_rate	16	serror_rate
6	dst_host_same_src_port_rate	17	service
7	dst_host_same_srv_rate	18	flag
8	dst_host_rerror_rate	19	src_bytes
9	dst_host_srv_count	20	srv_rerror_rate
10	dst_host_srv_diff_host_rate	21	srv_serror_rate
11	dst_host_srv_serror_rate	22	dst_host_srv_rerror_rate

C. Pruning

We use pruning to reduce the overall size of the trained neural model. There are several pruning strategies that can be used to this effect:

- The classical approach in which the model is firstly trained with all parameters and then subset of the trained parameters is removed during additional fine-tuning epochs.
- Pruning at initialization, where parameters are pruned before the model is trained [45].
- Pruning during the main training run.

Furthermore, pruning can carried out globally, i.e., across the whole model, or locally, i.e., in each network layer [46].

In this work we use global, magnitude-based pruning which employs fine-tuning epochs after the main training run, during which weights with low magnitudes are gradually set to zero.

IV. Experimental Setup

Table III summarizes the hyper-parameters used in the experiments. These training hyperparameters were selected with few trial training runs. We evaluate variants of this architecture with varying widths. In particular, we vary the number of neurons inside the residual blocks while keeping a fixed network width of the skip-connection nodes. Furthermore, batch normalization layers [47] are used to improve the training. This architecture proved to work well, while saving on the number of model parameters. Each constructed model was run five times with different random seeds.

We also carried out evaluation of pruned variants of our models. To this end, a 20 epoch fine-tuning run with magnitude-based pruning was done. For each network instance the sparsity schedule started with 85% initial sparsity and increased with each iteration, until it reached a final sparsity of 90% by the end of the last fine-tuning epoch. For the performance numbers we report mean and variance of training time, test accuracy, precision, recall and F_1 score:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

TABLE III: List of training hyper-parameters.

Models width	1024, 256, 32, 8
Activation function	ReLU [48]
Optimizer	Stochastic Gradient Descend (SGD) with Nesterov accelerated gradient = 0.9 [49]
Loss function	Binary Cross Entropy [50]
Learning rate	0.1
Decay for unpruned models	1e-6
Decay for pruned models	Polynomial Decay
Batch size	120
Number of epochs (for both pruned and unpruned networks)	20

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{4}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

where TP is the true-positive count, FP is the false-positive count, TN is the true-negative count and FN is the false-negative count. Due to the substantial pruning rate, the sparse models of the same width tended to have the same performance across random seeds. Consequently, the variance estimates are not meaningful in this case and we don't report them.

V. Results

The proposed models were trained using the Google Collab environment. Table IV reports the results for unpruned networks. The model with the highest width achieved 98.96% accuracy, 99.39% precision, 98.38% recall and 98.91% F_1 score. Note that this is the most computationally expensive of our models. That said, the model with quarter the width preformed equally well up to the variance across random seeds. The remaining two models performed slightly worse, with F_1 score around 0.6% below that of the larger models. These models were, however, much more computationally efficient, with the training time stabilizing below width equal to 32 units. Our results also shows that the variance across the training runs is low for all models, which shows that the performance is not highly affected by the initial seeds.

Results for the pruned networks are reported in Table V. The F_1 score of the model with 1024-unit width dropped by about 1.5% after pruning, with performance decrease manifesting mostly in model's recall. For the pruned model with quarter the width, the performance metrics were about 0.5% below those of the larger pruned model and up to 2% below the unpruned network. The two smallest models scored the lowest after pruning, with an F_1 score approximately 2% below larger pruned networks. Overall, our results shows that even with aggressive pruning and small initiated models residual fully-connected networks perform well in this task, with precision recall and F_1 score above 95%.

TABLE IV: Performance metrics for unpruned models.

Model	Training Time(sec)	Accuracy	Precision	Recall	F1 Score
1024	360.6±35.7	98.96±0.03%	99.39±0.10%	98.38±0.11%	98.91±0.07%
256	155.4±6.5	98.91±0.05%	99.44±0.02%	98.22±0.08%	98.82±0.05%
32	72.9±7.1	98.38±0.07%	81.07±0.16%	97.45±0.23%	98.25±0.08%
8	73.7±8.5	98.38±0.07%	99.07±0.16%	97.45±0.23%	98.25±0.08%

TABLE V: Performance metrics for pruned models.

Model	Training Time(sec)	Accuracy	Precision	Recall	F1 Score
1024	471.3	97.67%	99.42%	95.59%	97.46%
256	217.9	97.31%	99.22%	95.01%	97.07%
32	118.6	95.82%	95.91%	95.13%	95.52%
8	119.3	95.82%	95.91%	95.13%	95.52%

TABLE VI: Performance metrics reported in related work.

Reference	Accuracy	Precision	Recall	F1 Score
[32]	99.20%	99.02%	99.27%	99.14%
[35]	99.20%	-	99.27%	-
[36]	92.42%	90.20%	-	92.29%
[37]	99.37%	-	92%	-
[38]	97.60%	97%	97%	97%
[39]	98.96%	-	-	92.28%
[40]	97.83%	-	-	-
Our unpruned 1024 model	98.96±0.03%	99.39±0.10%	98.38±0.11%	98.91±0.07%
Our pruned 32 model	95.82%	95.91%	95.13%	95.52%

To benchmark our results against the state of the art, we selected peer-reviewed papers which addressed the binary classification task with respect to the same NSL-KDD dataset. Some of these papers reported all the metrics mentioned earlier, while others took into consideration only a subset of them. Comparison between the benchmarks and our results is summarized in in table VI.

Comparing with the state-of-the-art for this benchmark dataset in binary classification setup, we observe that all of the proposed unpruned networks give competitive or better precision in detecting attacks (Table VI). More precisely, the models with 1024 and 256 widths achieved better accuracy compared to [36] [38] [39] [40], recall compared to [37] [38] and F_1 score compared to [36] [38] [39]. The pruned models achieved slightly lower results, but still maintained strong performance while requiring only 10% of the initial parameters.

VI. Conclusions and Future Work

Proliferation of IoT devices is making a huge impact on the communication sector. The increased interconnectivity comes not only with new business opportunities, but also increases security risks related to prevalence of network vulnerabilities and persistent cyberattack threats. Conventional IDS and firewalls deployed to counter cyber-threats are often inadequate for IoT environments, e.g., due to high false-positive rates or large resource requirements. In this paper we proposed an ML-based IDS that employs residual MLP networks and demonstrated that it provides strong results with respect to the precision and recall of attack detection, even when implemented with relatively small networks. We also demonstrated that it retains most of its accuracy after pruning of as much as 90% of its parameters.

In our future work we intend to extend this line of research with novel and promising neural architectures, e.g., transformer models. These models excel at text embedding and classification. We therefore intend to explore their ability to classify network and system logs. We also intend to explore more pruning strategies, e.g., unit-based pruning which removes entire neurons, rather than individual weights. Such pruning strategies may result in lower computational footprint, while still maintaining strong attack detection performance.

References

[1] P. P. Gaikwad, J. P. Gabhane, and S. S. Golait, "A survey based on smart homes system using internet-of-things," in *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*. IEEE, 2015, pp. 0330–0335.

[2] D. Kuipers and M. Fabro, "Control systems cyber security: Defense in depth strategies," Idaho National Lab.(INL), Idaho Falls, ID (United States), Tech. Rep., 2006.

[3] Y. Lin, C. Wang, C. Ma, Z. Dou, and X. Ma, "A new combination method for multisensor conflict information," *J. Supercomputing*, vol. 72, no. 7, pp. 2874–2890, 2016.

[4] H. Liao, C. R. Lin, Y. Lin, and K. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network Computing Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[5] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[6] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[7] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security," in *10th International Conference on Cyber Conflict, CyCon 2018, Tallinn, Estonia, May 29 - June 1*, T. Minárik, R. Jakschis, and L. Lindström, Eds. IEEE, 2018, pp. 371–390.

[8] S. H. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, *A Guide to Convolutional Neural Networks for Computer Vision*, ser. Synthesis Lectures on Computer Vision. Morgan & Claypool Publishers, 2018.

[9] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42 210–42 219, 2019.

[10] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking, ICOIN 2017, Da Nang, Vietnam, January 11-13*. IEEE, 2017, pp. 712–717.

[11] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50 850–50 859, 2018.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30*. IEEE Computer Society, 2016, pp. 770–778.

[13] H. H. Pajouh, A. Dehghantanha, R. Khayami, and K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computing Systems*, vol. 85, pp. 88–96, 2018.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[15] R. M. S. Priya, P. K. R. Maddikunta, P. M., S. Koppu, T. R. Gadekallu, C. L. Chowdhary, and M. Alazab, "An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in iomt architecture," *Computing and Communication*, vol. 160, pp. 139–149, 2020.

[16] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[17] C. D. McDermott, F. Majdani, and A. Petrovski, "Botnet detection in the internet of things using deep learning approaches," in *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13*. IEEE, 2018, pp. 1–8.

[18] Y. Zhang, P. Li, and X. Wang, "Intrusion detection for iot based on improved genetic algorithm and deep belief network," *IEEE Access*, vol. 7, pp. 31 711–31 722, 2019.

[19] B. A. Tama and K.-H. Rhee, "Attack classification analysis of IoT network via deep learning approach," *Res. Briefs Inf. Commun. Technol. Evol.(ReBICTE)*, vol. 3, pp. 1–9, 2017.

[20] N. Moustafa and J. 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 2015, Canberra, Australia, November 10-12*. IEEE, 2015, pp. 1–6.

[21] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proceedings of the 16th European conference on cyber warfare and security. ACPI*, 2017, pp. 361–369.

[22] D. W. Vilela, T. F. Ed'Wilson, A. A. Shinoda, N. V. de Souza Araújo, R. De Oliveira, and V. E. Nascimento, "A dataset for evaluating intrusion detection systems in [ieee] 802.11 wireless networks," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, 2014, pp. 1–5.

[23] J. Kim, "Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap," *Computational Statistics and Data Analysis*, vol. 53, no. 11, pp. 3735–3745, 2009.

[24] M. Al-Hawawreh, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications*, vol. 41, pp. 1–11, 2018.

[25] S. Latif, Z. Zou, Z. Idrees, and J. Ahmad, "A novel attack detection scheme for the industrial internet of things using a lightweight random neural network," *IEEE Access*, vol. 8, pp. 89 337–89 350, 2020.

[26] M. Pahl and F. Aubet, "Ds2os traffic traces IoT traffic traces gathered in a the ds2os iot environment," 2018.

[27] N. Shone, N. N. Tran, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[28] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, pp. 197–227, 2016.

[29] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "TR-IDS: anomaly-based intrusion detection through text-convolutional neural network and random forest," *Secur. Commun. Networks*, pp. 4 943 509:1–4 943 509:9, 2018.

[30] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

[31] C. Lin, Z. Wang, J. Deng, L. Wang, J. Ren, and G. Wu, "mts: Temporal-and spatial-collaborative charging for wireless rechargeable sensor networks with multiple vehicles," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19*. IEEE, 2018, pp. 99–107.

[32] A. A. Diro and N. K. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computing Systems*, vol. 82, pp. 761–768, 2018.

[33] A. A. Bukhari, F. K. Hussain, and O. K. Hussain, "Fog node discovery and selection: A systematic literature review," *Future Generation Computing Systems*, vol. 135, pp. 114–128, 2022.

[34] L. Dhanabal and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International journal of advanced research in computer and communication engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[35] A. A. Diro and N. K. Chilamkurti, "Deep learning: The frontier for distributed attack detection in fog-to-things computing," *IEEE Communincation Magazine*, vol. 56, no. 2, pp. 169–175, 2018.

[36] M. Almiani, A. A. Ghazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for iot intrusion detection system," *Simulation Modelling and Practice Theory*, vol. 101, p. 102031, 2020.

[37] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Computer Security*, vol. 92, p. 101752, 2020.

[38] P. Devan and N. Khare, "An efficient xgboost-dnn-based classification model for network intrusion detection system," *Neural Computations and Applications*, vol. 32, no. 16, pp. 12 499–12 514, 2020.

[39] A. Nagisetty and G. P. Gupta, "Framework for detection of malicious activities in iot networks using keras deep learning library," in *2019 3rd international conference on computing methodologies and communication (ICCMC)*. IEEE, 2019, pp. 633–637.

[40] Z. Lv, L. Qiao, J. Li, and H. Song, "Deep-learning-enabled security issues in the internet of things," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9531–9538, 2021.

[41] M. Pal, "Random forest classifier for remote sensing classification," *International journal of remote sensing*, vol. 26, no. 1, pp. 217–222, 2005.

[42] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets," in *Proceedings of the third annual conference on privacy, security and trust*, vol. 94. Citeseer, 2005, pp. 1723–1722.

[43] 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, CISDA 2009, Ottawa, Canada, July 8-10*. IEEE, 2009, pp. 1–6.

[44] G. Pyatt, "On the interpretation and disaggregation of gini coefficients," *The Economic Journal*, vol. 86, no. 342, pp. 243–255, 1976.

[45] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Pruning neural networks at initialization: Why are we missing the mark?" in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7*. OpenReview.net, 2021.

[46] D. W. Blalock, J. J. G. Ortiz, J. Frankle, and J. V. Guttag, "What is the state of neural network pruning?" in *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4*, I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, Eds. mlsys.org, 2020.

[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 448–456.

[48] K. Fukushima, "Cognitron: A self-organizing multilayered neural network," *Biological cybernetics*, vol. 20, no. 3-4, pp. 121–136, 1975.

[49] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June*, ser. JMLR Workshop and Conference Proceedings, vol. 28. JMLR.org, 2013, pp. 1139–1147.

[50] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020.