Robust Network Intrusion Detection Through Explainable Artificial Intelligence (XAI)

Pieter Barnard, Nicola Marchetti, *Senior Member, IEEE*, and Luiz A. DaSilva, *Fellow, IEEE*

***Abstract*—In this letter, we present a two-stage pipeline for robust network intrusion detection. First, we implement an extreme gradient boosting (XGBoost) model to perform supervised intrusion detection, and leverage the SHapley Additive exPlanation (SHAP) framework to devise explanations of our model. In the second stage, we use these explanations to train an auto-encoder to distinguish between previously seen and unseen attacks. Experiments conducted on the NSL-KDD dataset show that our solution is able to accurately detect new attacks encountered during testing, while its overall performance is comparable to numerous state-of-the-art works from the cybersecurity literature.**

***Index Terms*—Network intrusion detection system (NIDS), anomaly detection, machine learning, explainable artificial intelligence (XAI).**

I. Introduction

IN RECENT years, there has been an alarming increase in the number of cybersecurity threats witnessed around the globe. As an effective tool to tackle these threats, network intrusion detection systems (NIDSs) are designed to monitor traffic flows within a network and alert cybersecurity personnel whenever potential attacks occur. At the forefront of today's state-of-the-art, machine learning (ML), and in particular deep learning methods for NIDSs, have allowed experts to achieve high detection accuracy on a wide host of simulated and real-world intrusion datasets [1]. However, in the realm of cybersecurity, the emergence of new technologies and ongoing efforts by attackers means that new (zero-day) attacks are constantly appearing across our networks, posing major challenges to the design of NIDSs that can remain robust over long periods of time.

In the cybersecurity literature, two main approaches dominate the design of a NIDS [2]. In the first approach, the NIDS is designed in a supervised manner to detect whenever a traffic flow exhibits signatures which are similar to those appearing alongside *known* attacks. In the second approach, unsupervised ML methods are used to learn statistical or latent representations of normal traffic, and anomaly detection methods are subsequently employed to detect whenever a flow differs significantly from this baseline. Importantly, while NIDSs based on the former approach have shown general success against known attacks, they often fail when presented with zero-day attacks [3]. On the other hand, recent works adopting deep learning solutions for anomaly-based NIDS, such as deep autoencoders [4]–[6], have demonstrated promising results against detecting new attacks.

In addition to high detection accuracy, many cybersecurity experts now also consider interpretability as an essential characteristic of a robust NIDS. This is particularly the case for NIDSs based on deep learning models, as their inherent 'black-box' nature means that even once malicious traffic has been successfully detected, considerable human effort is still required to determine *why* the flow is malicious, and therefore, how to best deal with the attack [7]. Recently, the emergence of Explainable AI (XAI) techniques for *post-hoc* interpretability has led to a new wave of cybersecurity works that now include additional layers of explainability for a human-in-the-loop [8]–[10]. For example, in [10], the authors demonstrate how the SHapley Additive exPlanation (SHAP) framework [11] can be used to gain insights into the features of their model which contribute the most to different types of attacks.

In this letter, we combine the benefits of XAI alongside the strengths of both supervised and unsupervised NIDSs, and propose a two-stage pipeline for robust network intrusion detection. In the first stage, we implement an extreme gradient boosting (XGBoost) model to perform supervised intrusion detection, and leverage the prominent SHAP framework to devise explanations of our model. In the second stage, we pass these explanations as input to a deep autoencoder module, whose primary purpose is to learn a latent representation of our model's 'typical' behaviour during training. Based on the hypothesis that our model will deviate from this typical baseline when trying to classify zero-day attacks, we then perform anomaly detection based on the reconstruction error of the autoencoder to determine whether a traffic flow *potentially* belongs to a new class of attacks during testing.[1] We note that while the former stage of our pipeline bears similarity to existing works for explainable intrusion detection, the use of explanations for anomaly detection in the latter stage appears to be completely unexplored in current research.

To the best of our knowledge, this letter is the first to expand beyond the mere use of XAI as an end-to-end tool to aid a human-in-the-loop understand why decisions are made by

Manuscript received 28 February 2022; revised 18 May 2022; accepted 19 June 2022. Date of publication 27 June 2022; date of current version 26 August 2022. This work was supported in part by the Science Foundation Ireland under Grant 18/CRT/6222 and Grant 13/RC/2077_P2, and in part by the Commonwealth Cyber Initiative (CCI). The associate editor coordinating the review of this article and approving it for publication was L. Foschini. *(Corresponding author: Pieter Barnard.)*

Pieter Barnard and Nicola Marchetti are with the CONNECT Research Centre, Trinity College Dublin, Dublin 2, D02 PN40 Ireland (e-mail: barnardp@tcd.ie; nicola.marchetti@tcd.ie).

Luiz A. DaSilva is with the Commonwealth Cyber Initiative, Virginia Tech, Arlington, VA 22203 USA (e-mail: ldasilva@vt.edu).

Digital Object Identifier 10.1109/LNET.2022.3186589

[1] It is important to note that traffic may still be deemed anomalous even if it does not stem from a new class of attacks. Such cases, which we refer to as new 'normal', may occur if the training data is incomplete or unable to account for all conceivable input-output scenarios.

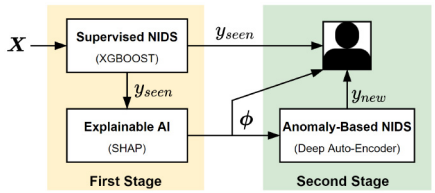


Fig. 1. Proposed pipeline for robust network intrusion detection.

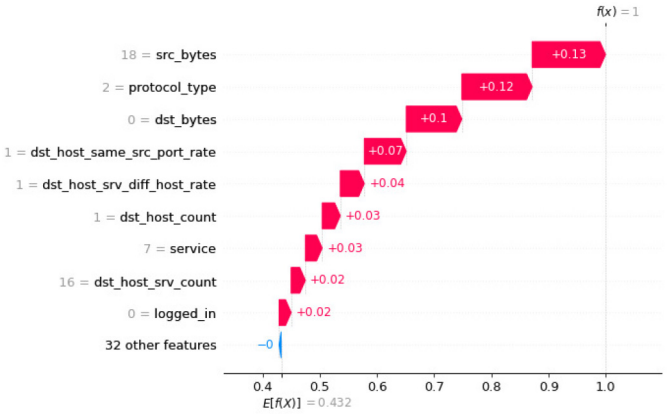


Fig. 2. Example SHAP explanation of a random sample taken from the training set (Probe Attack).

the NIDS, but also to explore the feasibility of employing additional ML mechanisms on these explanations to further enhance the performance of the NIDS. Moreover, numerical evaluations conducted on the NSL-KDD intrusion dataset [12] show that our solution is able to outperform various state-of-the-art works in terms of its overall accuracy, recall and precision.

II. Proposed Solution

This section presents the details of our proposed pipeline. In Sections II-A and II-B, we outline our initial model for supervised intrusion detection and use of SHAP to explain its behaviour, respectively. In Section II-C, we describe our anomaly-based NIDS and provide overall comments about our solution. Sections II-D and II-E discuss the pre-processing and implementation aspects of our solution, respectively. To foster reproducibility, we also provide full access to the source code used in our work.[2]

A. First Stage - Supervised Detection

Fig. 1 summarises the main stages of our proposed pipeline. In the initial stage, relevant data from the network is collected and pre-processed into an N-dimensional vector of features, $\mathbf{x} = [x_1, \ldots, x_N] \in \mathbb{R}^N$. For example, a typical flow-based NIDS may include features related to the general statistics of the flow, such as the amount of time the flow has been active, the average packet payload size, and the number of urgent packets sent across the flow, etc. Following the pre-processing stage, the input features are then passed into an XGBoost model and optimised to perform binary classification, where the output of the model, $y_{seen} \in [0, 1]$, is a scalar value corresponding to the probability of a certain flow being malicious, with $y_{seen} \geq 0.5$ implying malicious traffic and $y_{seen} < 0.5$ implying normal traffic.

B. First Stage - Explainable AI

As XGBoosts models are generally difficult to interpret given their complex ensemble structures, we consider the use of XAI techniques to construct post-hoc explanations of our model in the next sub-block of our pipeline. Specifically, we implement the SHAP method from [11], which has recently been proposed as a state-of-the-art approach for obtaining accurate explanations of tree-based models with polynomial time complexity. Under this framework, an explanation for a particular sample takes the form of a vector of "feature importance scores" which reflect the *magnitude* and *direction* by which each feature impacts the decision of the model for that sample.

For instance, in the example shown in Fig. 2, which corresponds to an explanation of a training sample that our model has previously classified as being malicious (in this case a Probe attack is occurring), features which have a positive impact on the model's decision (i.e., that raise the probability of a particular flow being malicious) are shown next to red bars, while features that reduce the probability are shown next to blue bars. From Fig. 2, we can see that the main features considered by the model include the 'src_bytes' feature (in this case a value of 18 bytes has raised the probability by 0.13%) and the 'protocol_type' (in this case UDP type traffic has raised the probability by 0.12%). In addition, we see that other features, such as the number of bytes received from the destination or port rates also have small but positive impacts on the model's decision, while the remaining 32 features (not shown due to space constraints) have a combined negative but negligible impact towards the model's decision.

Formally, given a specific output from our NIDS, y_{seen}, SHAP produces an explanation in the form of a vector of importance scores or "SHAP values", $\boldsymbol{\phi}(\mathbf{x}) = [\phi_1, \ldots, \phi_i, \ldots, \phi_N]$, where ϕ_i denotes x_i's impact on the observed output. Moreover, we note that SHAP takes its basis from the Shapley value from game theory, and exhibits a number of desirable properties in contrast to most other XAI methods in the literature [11]. For example, one key property of this approach, known as the "local accuracy" property, ensures that the sum of all SHAP values for a specific instance adds up to the *difference* between the model's output for that instance and a constant baseline, ϕ_0. I.e.,

$$\sum_{i=1}^{N} \phi_i = y_{seen} - \phi_0. \tag{1}$$

In SHAP, the baseline value, ϕ_0, corresponds to the average or expected output of the model seen during training, and can be interpreted as the initial output of the model before the impact of any features are taken into account (i.e., in Fig. 2 the baseline value is found to be ≈ 0.432). Moreover, we see

[2]Source code: https://github.com/barnardp/Intrusion_Detection_XAI.

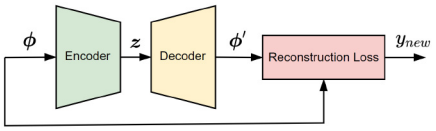


Fig. 3. Our Anomaly-Based NIDS.

from the right-hand side of equation (1), that the explanation essentially *decomposes* the output of the model (ignoring the constant baseline effect for emphasis) amongst each of the input features. In the context of our NIDS, this means that cybersecurity experts can easily understand the relative importance of each feature as it increases or decreases the probability of a flow being classified as malicious. This, in turn, can shed much needed light on nature of the attack, as well as how best to counteract it.

C. Second Stage - Anomaly Detection

In the first stage of our pipeline, the use of a supervised ML model implies that we can generally expect our system to achieve good performance on previously seen attacks, but still face the likely possibility of performing poorly against unseen attacks. In the second stage of our pipeline, we attempt to address this shortcoming by exploring the hypothesis that our model will behave differently when attempting to classify traffic flows emanating from zero-day attacks, compared to how it behaves when classifying traffic similar to that seen during training. To test this hypothesis, we turn to the discriminatory power provided by unsupervised learning techniques, such deep autoencoders. Specifically, we design an anomaly-based NIDS, where the aim in this case is to accurately differentiate between previously seen behaviours (i.e., explanations based on the training data), and new behaviours (i.e., arising from zero-day attacks, or possibly, from new 'normal' traffic).

Fig. 3 illustrates our anomaly-based NIDS. Here, explanations from the first stage are passed to a deep autoencoder, which essentially consists of two deep neural networks (DNNs) in which the output of the first network (the 'encoder'), is fed directly into the input of the second network (the 'decoder'). Moreover, by purposely reducing the dimension of the encoder's output, z, to well below that of its input, the encoder is forced to learn a compressed latent representation of the original input using only the most salient aspects encountered during its training. In the case of the decoder, the network size is gradually increased across each each hidden layer, up to the point where the final output, $\boldsymbol{\phi'}$, has similar dimension to the original explanation, $\boldsymbol{\phi}$. The overall encoder-decoder network can then be jointly optimised to reconstruct explanations based on the training data.

Under the assumption that our original hypothesis holds true, we can expect that the reconstruction error of our autoencoder will increase significantly whenever new attacks or new 'normal' flows are encountered. To test this hypothesis and ensure such flows are reliably flagged at the output of the NIDS, y_{new}, we calculate the absolute reconstruction error (ARE) of each sample during testing and consider any flows resulting in an ARE greater than the 95^{th} percentile of that seen during training as anomalous, where,

$$ARE(\boldsymbol{\phi}, \boldsymbol{\phi'}) = \frac{\sum_{i=1}^{N} |\phi_i - \phi'_i|}{N}. \quad (2)$$

In the final steps of our pipeline, the explanation and classification results from both our supervised and anomaly-based NIDSs are presented to cybersecurity personnel for further analysis. Although specific details around such analysis fall outside the scope of this letter, we briefly highlight two edge cases which may occur in a practical setting. For example, in the case of a flow which has been deemed anomalous but not malicious, an analyst may have to manually investigate the flow and its explanation to determine whether it corresponds to a new attack or a new 'normal' instance. However, if a flow is found to be malicious but not anomalous, it is likely that this flow stems from a known attack, implying that an automated response may be possible. Moreover, we stipulate that the former case may find use in an online-learning scenario, where the NIDS is continuously adapted over time to maintain its accuracy, while the combination of both these cases may have further consequences related to the 'zero-touch' paradigm envisioned for 6G and next-generation networks [13].

D. Data Pre-Processing

As previously outlined, our XGBoost model for supervised intrusion detection requires all inputs to the model to be of numerical format. To ensure this requirement is met during the training and testing of our solution, we apply label encoding to all non-numerical features passed to our model using the 'LabelEncoder' method from the Scikit-Learn [14] library for Python. For the NSL-KDD dataset used in this letter, this includes three features which are initially categorical in format: 'protocol_type', 'service', and 'flag'. In the case of our autoencoder, we convert all input features into the range $(-1, 1)$ using the 'MinMaxScaler' method from Scikit-Learn. Finally, as we only consider binary classification in our current work, we merge all non-normal labels in the NSL-KDD dataset into a single class of attacks.

E. Pipeline Implementation

We implement our XGBoost model using Scikit-Learn's ML API. To train our model, we use a binary logistic loss function and keep all remaining parameters of the model at their default values. In the case of our autoencoder model, we implement both the encoder and decoder networks using the TensorFlow API [15]. Before training each network, we split the original training set into two subsets, and use roughly 80% of the original data as a new training set, and the remaining 20% as a validation set. We note that any samples used during the evaluation of our solution are kept separate from the training and validation sets used at this stage.

For our encoder network, we use 4 neural layers, including an input layer, two hidden layers, and an output layer. For each layer, we use a 'ReLu' activation function with 41, 1456, 724, 14 neurons, respectively. In order to avoid overfitting during training, we also apply dropout with strength 0.2 on the first hidden layer. Similarly, our decoder consists

of 4 layers with 14, 632, 1644, 41 neurons, respectively, as well as a 'ReLu' activation function at each layer, except for the final output layer, which uses a 'tanh' function instead to ensure all outputs fall within the range $(-1, 1)$. We train the encoder-decoder networks jointly for 1000 epochs using Tensorflow's default 'Adam' optimiser with a mean absolute error loss function, and an early-stopping criteria based on the validation loss during training. We also employ mini-batching of size 512 to aid the convergence speed of our autoencoder during training. In addition, we note that the final parameters of our encoder-decoder network have been chosen based on a mixture of empirical experiments, as well as a grid search strategy.

Finally, we use an open-source implementation of SHAP [11] to compute explanations our model. Specifically, we use the 'TreeExplainer' method and set the feature perturbation option to 'interventional' to ensure the explanations remain faithful to the model's true behaviour [16].

III. Evaluation

In this section, we provide a brief overview of the NSL-KDD dataset and our evaluation strategy, followed by a summary and discussion of our results.

A. NSL-KDD Dataset

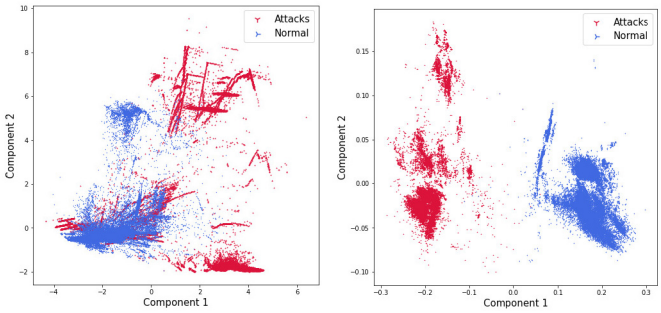
Proposed by [12], the NSL-KDD dataset presents an improved version of the KDD'99 dataset, originally released by DARPA as one of the first publicly-available intrusion datasets to contain a wide host of realistic attacks to a military network. Similar to the KDD'99 dataset, the NSL-KDD dataset contains 41 network-related features derived from TCP/IP dumps, as well as offering examples of 23 different attacks in its training set, with an additional 17 new attacks across its test set. In comparison to KDD'99, the NSL-KDD dataset offers a number of important improvements aimed at promoting greater consistency and fairness when comparing across different NIDSs, including the removal of duplicate flows and the use of a proportional inclusion policy to minimise class-imbalance issues associated with rare attack types. Although the NSL-KDD dataset has some limits in its ability to capture examples of more modern attack types, it presents one of the few publicly available intrusion datasets that can be used to evaluate the performance of a NIDS where there is a shift in the training and testing distributions; in this letter, we use the 'KDDTrain+' and 'KDDTest+' datasets for training and evaluating our solution, respectively.

B. Evaluation Strategy

For our evaluation, we consider separately the performance of: i) our supervised NIDS; ii) our anomaly-based NIDS; as well as iii) the combination of both NIDS in our overall pipeline. For cases (i) and (iii), we calculate performance metrics based on the general ability of our NIDSs to detect attacks, new or old, while for case (ii) we calculate metrics based solely on instances of new attacks. In our analysis, we consider common network intrusion metrics, such as accuracy, recall, and precision. To benchmark our work, we also compare our results against those reported in various state-of-the-art works which also incorporate the NSL-KDD dataset in their evaluations [6], [17], as well as any other works which offer explainability in their solutions [8], [10]. We note that the results reported in [9] are not included in this comparison, as these only consider a portion of the NSL-KDD test set.

TABLE I
Performance of Our Proposed NIDS Pipeline

Method	Accuracy (%)	Recall (%)	Precision (%)	Scope
Supervised NIDS	79.20	65.61	96.82	All
Anomaly-based NIDS	78.53	89.41	43.01	New
Overall Pipeline	93.28	97.81	91.05	All



(a) (b)

Fig. 4. PCA applied to the training data (a), and the explanations (b).

C. Results & Discussion

Table I summarises the performance of our supervised NIDS, our anomaly-based NIDS, as well as our overall pipeline, where for simplicity, we assume any instance flagged as either malicious or anomalous to be an attack.[3] Here we see that, on its own, our supervised NIDS achieves a reasonable accuracy of 79% against general attacks, with an additional high precision of 96% but low recall of 65%. On the other hand, our anomaly-based NIDS achieves a relatively low precision of 43% on new attacks[4] but high recall of 89%, with an overall accuracy of 78%. Remarkably, when we combine both NIDSs together, our overall pipeline performs strongly across all three metrics, achieving an overall accuracy of 93%, precision of 91% and recall of 97%. This significant jump in performance supports our main hypothesis, that the explanations from our supervised NIDS can in fact be used alongside additional ML mechanisms to greatly enhance the performance of our intrusion system.

To try and explain why this is the case, we apply principal component analysis (PCA) to both the raw training data as well as its explanations. As seen in Fig. 4a, plotting the first 2 PCA components of the raw training data reveals a strong overlap between samples of normal and malicious traffic. On the other hand, a plot of the first 2 PCA components based on the training data's explanations shows that the 'explanation

[3]Since it is possible for an analyst to further validate each anomalous instance as being either an attack or new 'normal' instance, the results presented here constitute a lower bound on what may be achieved in practice.

[4]We believe this low precision can be largely attributed to the fact that all anomalies are assumed to be attacks, i.e., the precision is reduced due to the presence of new 'normal' instances being flagged as attacks.

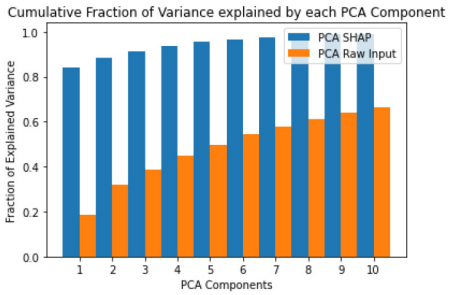


Fig. 5. Cumulative variance explained by each PCA component.

TABLE II
OVERALL PERFORMANCE AGAINST BENCHMARKS

Method	Accuracy (%)	Recall (%)	Precision (%)	XAI
Proposed Solution	93.28	97.81	91.05	✓
Yang *et al.* [6]	89.36	84.86	95.98	✗
Javaid *et al.* [17]	88.39	95.95	85.44	✗
Wang *et al.* [10]	80.6	80.6	82.8	✓
Marino *et al.* [8]	95.5	-	-	✓

domain' is able to distinctively separate clusters of normal flows from malicious ones. We believe one possible reason is that the explanation model itself is formulated in terms of a simplified linear model, i.e., in generating the SHAP values, we inherently disentangle some of the non-linearities existing throughout the decision space. Additionally, when we examine the cumulative percentage of variance (PoV) explained by each PCA component, as shown in Fig. 5, we find that only 32% of the data variance is explained by the first 2 components of the raw training data, compared to 88% in the case of the explanations. As the PoV can be regarded as a measure of how much of the original signal information is contained within each component, this suggests that the explanation domain is also capable of compressing greater amounts of important information (i.e., salient aspects from the decision boundary between normal and abnormal traffic) more efficiently than the raw training data.

Table II compares the overall performance of our pipeline against various state-of-the-art works from the literature. As seen in Table II, our solution is able to outperform all but one of the considered benchmarks in terms of its overall accuracy, falling short by only 2.2% compared to [8]. In addition, our solution is able to outperform all benchmarks in terms of its recall rate, while only slightly under-performing in terms of its precision compared to the work in [6]. While these results suggest that our solution is capable of achieving satisfactory performance compared to the state-of-the-art for binary classification, we hope to extend our analysis to the case of multiclass classification in future work.

IV. CONCLUSION

In this letter, we have presented a two-staged pipeline for robust network intrusion detection. Our proposed pipeline consists of an initial XGBoost model to perform supervised intrusion detection, followed by an autoencoder trained on SHAP explanations representing the behaviour of our initial model during training. By combining the detection capabilities of both these NIDSs, our overall pipeline is able to outperform numerous state-of-the-art works in terms of its accuracy, recall and precision on the NSL-KDD dataset, as well as offering an extra layer of explainability. In future work, we will extend our pipeline in line with the goals and concepts envisioned for future zero-touch networks. In particular, we envision this will require additional stages within our pipeline which can aid towards automating much of the manual tasks conventionally performed within the network management cycle, such as detecting and adapting to new attacks, as one example. Finally, we also plan to further investigate some of the findings of our work, such as the effectiveness of the 'explanation domain' in separating clusters of attacks from normal traffic, as well as to optimise the various stages our pipeline and to extend its ability to support multiclass classification across additional intrusion datasets, such as the more recent CIC-IDS2017 [18].

REFERENCES

[1] Y. Xin *et al.*, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.
[2] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, 2016, pp. 258–263.
[3] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.
[4] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[5] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018, *arXiv:1802.09089*.
[6] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, "Network intrusion detection based on supervised adversarial variational auto-encoder with regularization," *IEEE Access*, vol. 8, pp. 42169–42184, 2020.
[7] J. R. Goodall *et al.*, "Situ: Identifying and explaining suspicious behavior in networks," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 204–214, Jan. 2019.
[8] D. L. Marino, C. S. Wickramasinghe, and M. Manic, "An adversarial approach for explainable ai in intrusion detection systems," in *Proc. 44th Ann. Conf. IEEE Ind. Electr. Soc.*, 2018, pp. 3237–3243.
[9] K. Amarasinghe, K. Kenney, and M. Manic, "Toward explainable deep neural network based anomaly detection," in *Proc. 11th Int. Conf. Human Syst. Inter. (HSI)*, 2018, pp. 311–317.
[10] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73127–73141, 2020.
[11] S. M. Lundberg *et al.*, "From local explanations to global understanding with explainable AI for trees," *Nat. Mach. Intell.*, vol. 2, no. 1, pp. 56–67, 2020.
[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, 2009, pp. 1–6.
[13] C. Benzaid and T. Taleb, "AI-driven zero touch network and service management in 5G and beyond: Challenges and research directions," *IEEE Netw.*, vol. 34, no. 2, pp. 186–194, Mar./Apr. 2020.
[14] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.
[15] M. Abadi *et al.*. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." 2015. [Online]. Available: https://www.tensorflow.org/
[16] H. Chen, J. D. Janizek, S. Lundberg, and S.-I. Lee, "True to the model or true to the data?" 2020, *arXiv:2006.16234*.
[17] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," *EAI Endorsed Trans. Security Safety*, vol. 3, no. 9, p. e2, 2016.
[18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorban, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, vol. 1, 2018, pp. 108–116.