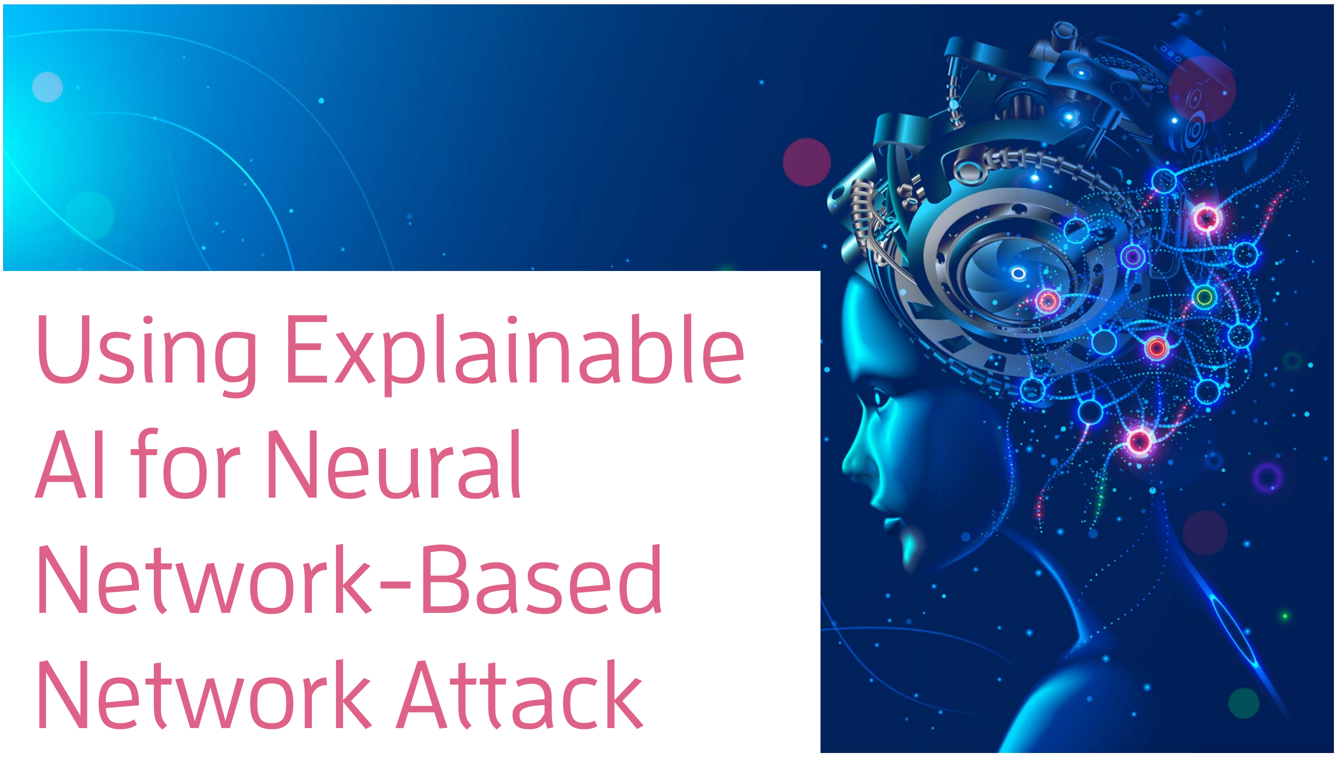


Using Explainable AI for Neural Network-Based Network Attack Detection

Qingtian Zou, The Pennsylvania State University
Lan Zhang, Northern Arizona University
Xiaoyan Sun, Worcester Polytechnic Institute
Anoop Singhal, National Institute of Standards and Technology
Peng Liu, The Pennsylvania State University

Neural network (NN)-based network intrusion detection systems (NIDSs) are becoming popular these days due to their notable advantages. This article reviews the current application of explainable artificial intelligence techniques and tools for explaining the behavior of the NIDS.

Digital Object Identifier 10.1109/MC.2023.3342602
Date of current version: 6 May 2024

Recently, deep learning and neural networks (NNs) have demonstrated exceptional performances in many areas, including image recognition, robotics, and natural language processing. Compared to traditional machine learning models (for example, decision trees (DTs), random forests, support vector machines, and so on), NNs have notable advantages including the abilities of dealing with complex data, reduced reliance on feature extraction, comparatively better performances, and so on.

However, due to the black-box nature of NNs, understanding how models make decisions is often difficult, both in a general sense (explanation) and in the context of a specific input (interpretation). This has hindered the applications of NNs in some critical missions where understanding rationales of the decision-making is imperative. This is especially true in the security domain. System administrators want to not only determine if the system

is under attack but also identify what indicators signal the potential attack, as the indicators are often the cues for knowing the root causes. For example, an NN-based network intrusion detection system (NIDS) usually detects intrusions including denial of service, probing, botnet, and other logic flaw exploiting attacks.[1] It can take network packets or postprocessing network data (for example, network flow data) as input and report whether (specific kinds of) attacks exist. If a NN-based NIDS detects an attack, the security analyst will want to know which packets lead to the attack, where the attack packets come from, and sometimes even which bytes of the packets are malicious. To achieve this goal, explainable artificial intelligence (XAI) is a valuable tool that can be used to "crack open the black box."

Generally speaking, XAI techniques can be categorized into two categories. One is to explain the NN as a whole, and the other is to interpret why a specific output is given for a specific input. The first category provides network-level explanations, while the second category provides per-data-sample explanations. In the literature, the second category has been extensively investigated, resulting in development of well-known tools such as Shapley additive explanations (SHAP)[2] and local interpretable model-agnostic explanations (LIME).[3] In contrast, the first category is less investigated. Nevertheless, DTs recently attracted researchers' attentions as a promising way to provide network-level explanations. With the provided training data samples and the corresponding NN's outputs (for each sample), the goal is to train a DT to emulate the performance of the NN, so that the NN's decision-making process can be approximated using the learned DT.

In this article, we seek to provide a critical review within the NIDS domain about the current applications of XAI techniques/tools in both the network-level explanations category and the per-data-sample interpretation category. To make the review insightful, we focus on the subtle connections between network-level explanations and per-data-sample interpretations. We have also conducted preliminary experiments to compare the two explanations using the NN-based Domain Name System (DNS) cache poisoning detection as a case study.

This has hindered the applications of NNs in some critical missions where understanding rationales of the decision-making is imperative.

PRIOR WORKS

XAI has several aspects. Aside from the existence of explanation, there are also principles in XAI such as meaningfulness, explanation accuracy, and knowledge limits.[4] Generally speaking, the goal of XAI is to reveal the decision-making process of AI models. There are also different criteria to categorize XAI techniques.[5] For example, whether the model is self-explainable,[6,7,8] or post hoc explanation is applied on a given model[2]; whether to surrogate a complicated model with a simpler or even self-explainable one[9]; and whether to explain individual data samples[10] or explain the model's overall behavior.[11,12,13,14]

One of the XAI techniques that have been tailored to cybersecurity problems with domain-specific knowledge is Trustee.[9] It learned high-fidelity and low-complexity DTs for network security problems. The DT is easy to interpret and can be used to identify the most important features for making predictions, as well as to understand the decision-making process used by the model. Humans can comprehend and retrace how AI models came to a specific output based on the DTs. However, DTs also have some limitations when used for explaining black-box deep learning models. For example, DTs may not be able to capture complex relationships between input features, and they may be sensitive to noise and outliers in the data. Therefore, it is important to carefully evaluate the use of DTs for explaining AI models used in network security.

DEEP LEARNING FOR DETECTING NETWORK ATTACKS

To demonstrate using XAI techniques for NN explanation, we will use an example NN from our prior work.[1] In the prior work, we proposed to use deep learning for detecting two network attacks: address resolution protocol (ARP) poisoning and DNS cache poisoning attacks. In this article, we choose the DNS cache poisoning detection rather than the ARP poisoning detection as the example to demonstrate the XAI techniques. This is due to the following reasons: 1) traditional machine learning techniques such as DTs and random forests can already achieve good performance for ARP poisoning detection; 2) the CNN for DNS cache poisoning detection is a more complex NN model compared to the MLP model for ARP poisoning, so it can better show the benefits of XAI techniques. Data and codes for training the detection NN have been published on GitHub.[a]

DNS poisoning works by spoofing DNS responses, with which the attacker can trick the victim into falsified mappings between domain names and IP addresses and, thus, redirect the network communication. It exploits the lack of response verification in the corresponding protocol. As a result, the victims cannot verify whether the packets come from a genuine host or attacker. DNS poisoning is difficult to be detected with traditional detection methods (for example, signatures, rules, anomaly detections, and so on) because spoofing is applied (that is, attacker packets are intentionally crafted to be indistinguishable from normal packets). Therefore, our prior work proposed to use a convolutional neural network (CNN) for detecting DNS cache poisoning. The model's accuracy, F1 score, and detection rate are all above 99%. However, little is known about how the CNN judges data samples to achieve this performance.

For example, DTs may not be able to capture complex relationships between input features, and they may be sensitive to noise and outliers in the data.

TWO COMMON EXPLANATION METHODS

Explanation DT

As a traditional machine learning technique, DTs are known for extracting decision rules that are easy to understand, and they have been repurposed to explain the complex NNs. The basic idea is to let the trained NN tutor the DT. Suppose there is a dataset containing data samples and their corresponding ground truths. Based on this dataset, a trained NN will generate a prediction for each data sample. To explain this NN, a DT can be trained with data samples and the NN's predictions. Note that the NN's predictions but not ground truths are used because DT here is used for explaining the NN but not for solving the original problem. The basic assumption for tree-based explanation is that, *if the DT and the NN can reach consensuses on the majority of data samples, then the DT should have learned the inner logic of the NN.*

Figure 1 shows a learned explanation DT with respect to the DNS cache poisoning detection NN. We used TRUSTEE,[9] a framework to explain machine learning models, to generate the DT. From the DT, a security analyst can gain the following insights of the tutor NN.

Feature importance. Which features are used to classify a data sample can be easily known from which features are shown in those judgment nodes. If a feature does not appear in the DT, it means that it is not important for DT's classifications at all. The importance of those features can also be inferred based on where they are located in the DT. The better a certain feature can separate the benign and malicious data samples, the closer this feature will appear to the root node in the tree.

Class distribution. Class distribution is shown in the "values" in every node of the DT, which represent the numbers of benign and malicious data samples in the specific node. For example, the root node in Figure 1 shows "value = [9,419, 9,215]," meaning that there are 9,419 benign data samples and 9,215 malicious data samples. Following the left branch from the root node, the next judgment node shows "value = [9,419, 1,718]," meaning that there are 9,419 benign data samples and 1,718 malicious data samples. The other 9,215–1,718=7,497 data samples, which are all malicious, are directed to the right branch following the root node. Similarly, by looking at this value in every node, the class distribution between benign and malicious can be clearly inferred.

Shapley values

Shapley values produced by SHAP,[2] a widely used tool to provide per-data-sample explanations, are used for measuring the impact of a certain feature's value toward the predicted result. Different from TRUSTEE, Shapley values are used for local interpretation. In other words, it is used to interpret how a prediction result is reached for a specific data sample. Specifically, Shapley values can tell how each feature contributes to the predicted results.

For example, Table 1 shows the interpretation for a benign data sample in DNS cache poisoning detection. A positive Shapley value means that the feature's value pushes the classification result toward being malicious, and a negative Shapley value does the contrary. Only the top-10 most positively and negatively contributing features and their Shapley values are shown. Specifically for this data sample, the most positively-contributing feature is the bit 1 of the `DNS` layer's `authority` RR field in the fourth packet in the data sample. The most negatively-contributing feature is the bit 6 of the `IP` layer's `ttl` field in the fourth packet in the data sample.

CONNECTIONS BETWEEN NETWORK-LEVEL AND PER-DATA-SAMPLE EXPLANATIONS

Since the explanations provided by DTs and SHAP are two different kinds of explanations, we believe that no meaningful conclusions on "which explanations are better" could be drawn. However, we observe that from the perspective of whether the explanations are in

[a] Chapter 5 in https://github.com/PSUCyberSecurityLab/AIforCybersecurity.

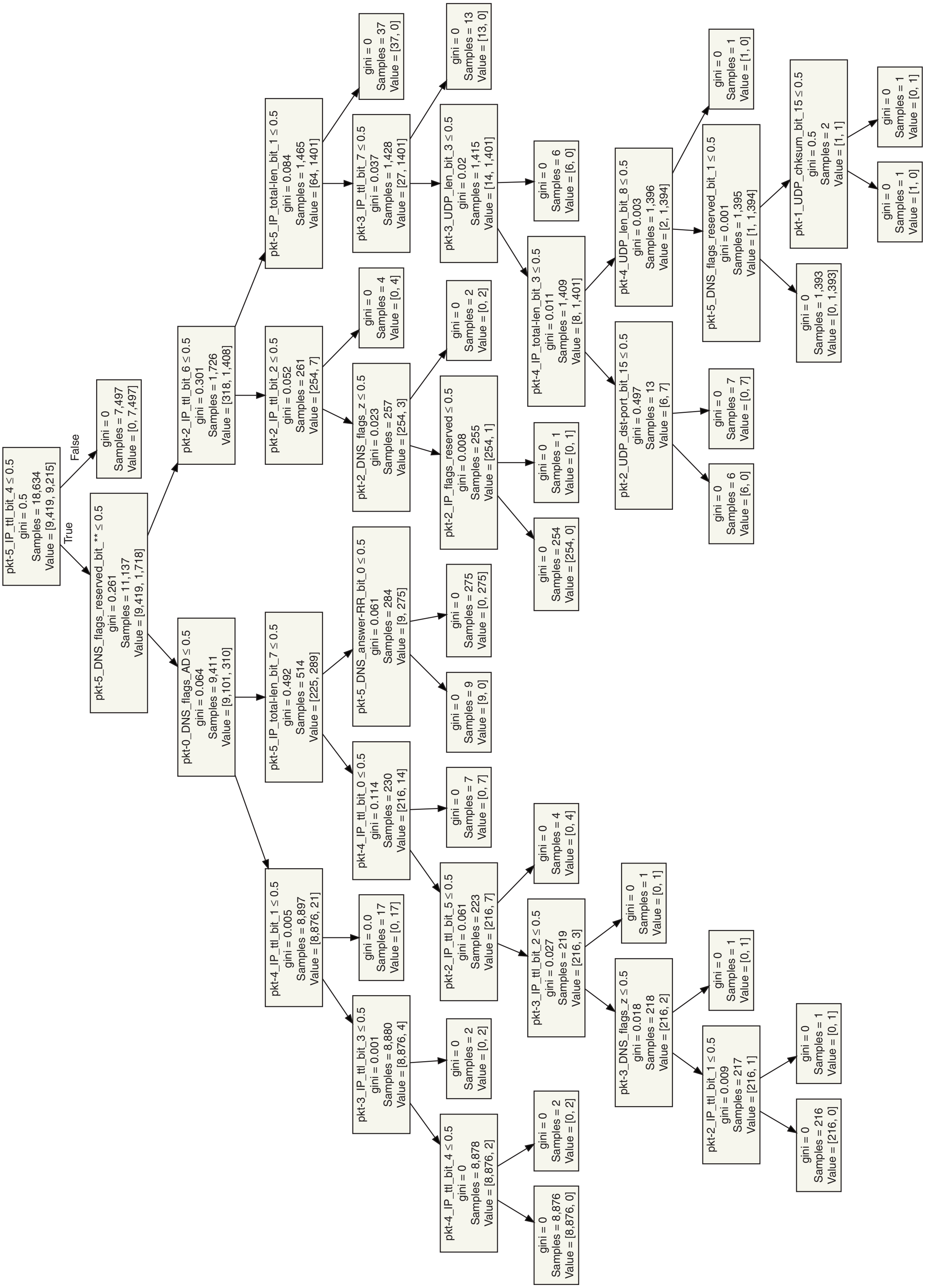

pkt-5_IP_ttl_bit_4 ≤ 0.5 gini = 0.5 Samples = 18,634 Value = [9,419, 9,215]
True
False
gini = 0 Samples = 7,497 Value = [0, 7,497]
pkt-5_DNS_flags_reserved_bit_** ≤ 0.5 gini = 0.261 Samples = 11,137 Value = [9,419, 1,718]
pkt-2_IP_ttl_bit_6 ≤ 0.5 gini = 0.301 Samples = 1,726 Value = [318, 1,408]
pkt-0_DNS_flags_AD ≤ 0.5 gini = 0.064 Samples = 9,411 Value = [9,101, 310]
pkt-5_IP_total-len_bit_1 ≤ 0.5 gini = 0.084 Samples = 1,465 Value = [64, 1401]
gini = 0 Samples = 37 Value = [37, 0]
pkt-3_IP_ttl_bit_7 ≤ 0.5 gini = 0.037 Samples = 1,428 Value = [27, 1401]
gini = 0 Samples = 13 Value = [13, 0]
pkt-3_UDP_len_bit_3 ≤ 0.5 gini = 0.02 Samples = 1,415 Value = [14, 1,401]
gini = 0 Samples = 6 Value = [6, 0]
pkt-4_IP_total-len_bit_3 ≤ 0.5 gini = 0.011 Samples = 1,409 Value = [8, 1,401]
pkt-4_UDP_len_bit_8 ≤ 0.5 gini = 0.003 Samples = 1,396 Value = [2, 1,394]
gini = 0 Samples = 1 Value = [1, 0]
pkt-5_DNS_flags_reserved_bit_1 ≤ 0.5 gini = 0.001 Samples = 1,395 Value = [1, 1,394]
pkt-1_UDP_chksum_bit_15 ≤ 0.5 gini = 0.5 Samples = 2 Value = [1, 1]
gini = 0 Samples = 1 Value = [0, 1]
gini = 0 Samples = 1 Value = [1, 0]
gini = 0 Samples = 1,393 Value = [0, 1,393]
pkt-2_UDP_dst-port_bit_15 ≤ 0.5 gini = 0.497 Samples = 13 Value = [6, 7]
gini = 0 Samples = 7 Value = [0, 7]
gini = 0 Samples = 6 Value = [6, 0]
pkt-2_IP_ttl_bit_2 ≤ 0.5 gini = 0.052 Samples = 261 Value = [254, 7]
gini = 0 Samples = 4 Value = [0, 4]
pkt-2_DNS_flags_z ≤ 0.5 gini = 0.023 Samples = 257 Value = [254, 3]
gini = 0 Samples = 2 Value = [0, 2]
pkt-2_IP_flags_reserved ≤ 0.5 gini = 0.008 Samples = 255 Value = [254, 1]
gini = 0 Samples = 1 Value = [0, 1]
gini = 0 Samples = 254 Value = [254, 0]
pkt-5_IP_total-len_bit_7 ≤ 0.5 gini = 0.492 Samples = 514 Value = [225, 289]
pkt-5_DNS_answer-RR_bit_0 ≤ 0.5 gini = 0.061 Samples = 284 Value = [9, 275]
gini = 0 Samples = 275 Value = [0, 275]
gini = 0 Samples = 9 Value = [9, 0]
pkt-4_IP_ttl_bit_0 ≤ 0.5 gini = 0.114 Samples = 230 Value = [216, 14]
gini = 0 Samples = 7 Value = [0, 7]
pkt-2_IP_ttl_bit_5 ≤ 0.5 gini = 0.061 Samples = 223 Value = [216, 7]
gini = 0 Samples = 4 Value = [0, 4]
pkt-3_IP_ttl_bit_2 ≤ 0.5 gini = 0.027 Samples = 219 Value = [216, 3]
gini = 0 Samples = 1 Value = [0, 1]
pkt-3_DNS_flags_z ≤ 0.5 gini = 0.018 Samples = 218 Value = [216, 2]
gini = 0 Samples = 1 Value = [0, 1]
pkt-2_IP_ttl_bit_1 ≤ 0.5 gini = 0.009 Samples = 217 Value = [216, 1]
gini = 0 Samples = 1 Value = [0, 1]
gini = 0 Samples = 216 Value = [216, 0]
pkt-4_IP_ttl_bit_1 ≤ 0.5 gini = 0.005 Samples = 8,897 Value = [8,876, 21]
gini = 0.0 Samples = 17 Value = [0, 17]
pkt-3_IP_ttl_bit_3 ≤ 0.5 gini = 0.001 Samples = 8,880 Value = [8,876, 4]
gini = 0 Samples = 2 Value = [0, 2]
pkt-4_IP_ttl_bit_4 ≤ 0.5 gini = 0 Samples = 8,878 Value = [8,876, 2]
gini = 0 Samples = 2 Value = [0, 2]
gini = 0 Samples = 8,876 Value = [8,876, 0]

FIGURE 1. Learned explanation DT[9] with respect to the DNS cache poisoning detection NN.

agreement with the experts' domain knowledge, the explanations provided by both DTs and SHAP are actually located on the same spectrum. In particular, the DT explanations provided by TRUSTEE are located near the right end of the spectrum due to the fact that TRUSTEE requires the decision rules in DTs "to be largely in agreement with the experts' domain knowledge."[9] In contrast, the Shapley value explanations provided by SHAP are located near the left end of the spectrum since the Shapley value of each feature is determined by the feature's average marginal contribution, which is calculated based on the output (that is, probabilities of each of the classes/labels involved in the classification task) of the black-box model itself. In other words, the explanations on the left end tend to illustrate the inner workings of the black-box model.

Note that the NN's predictions but not ground truths are used because DT here is used for explaining the NN but not for solving the original problem.

The aforementioned observation indicates that on the "spectrum" of whether the explanations are in agreement with the experts' domain knowledge, researchers could gain new insights through the "distance" between the explanations near the left end of the spectrum and those near the right end. Since the explanations provided by DTs and SHAP are not located on the same side of the spectrum, the extent to which they are "aligned with each other" indicates the extent to which expert-comprehensible explanations are aligned with the inner workings of the black-box model. To gain new insights into the extent to which expert-comprehensible explanations are aligned with the inner workings of the black-box model, we want to assess the extent to which the explanations provided by DTs and SHAP are aligned. However, we must avoid directly checking whether the high-Shapley-value features play a major role in the DTs because the Shapley values provided by SHAP explain how a black-box model makes decisions near a particular data point. On the other hand, based on the observation that each root-node-to-leave-node path in a DT explains how a black-box model makes decisions for a subset of the data samples, subset-level alignment assessment could be conducted.

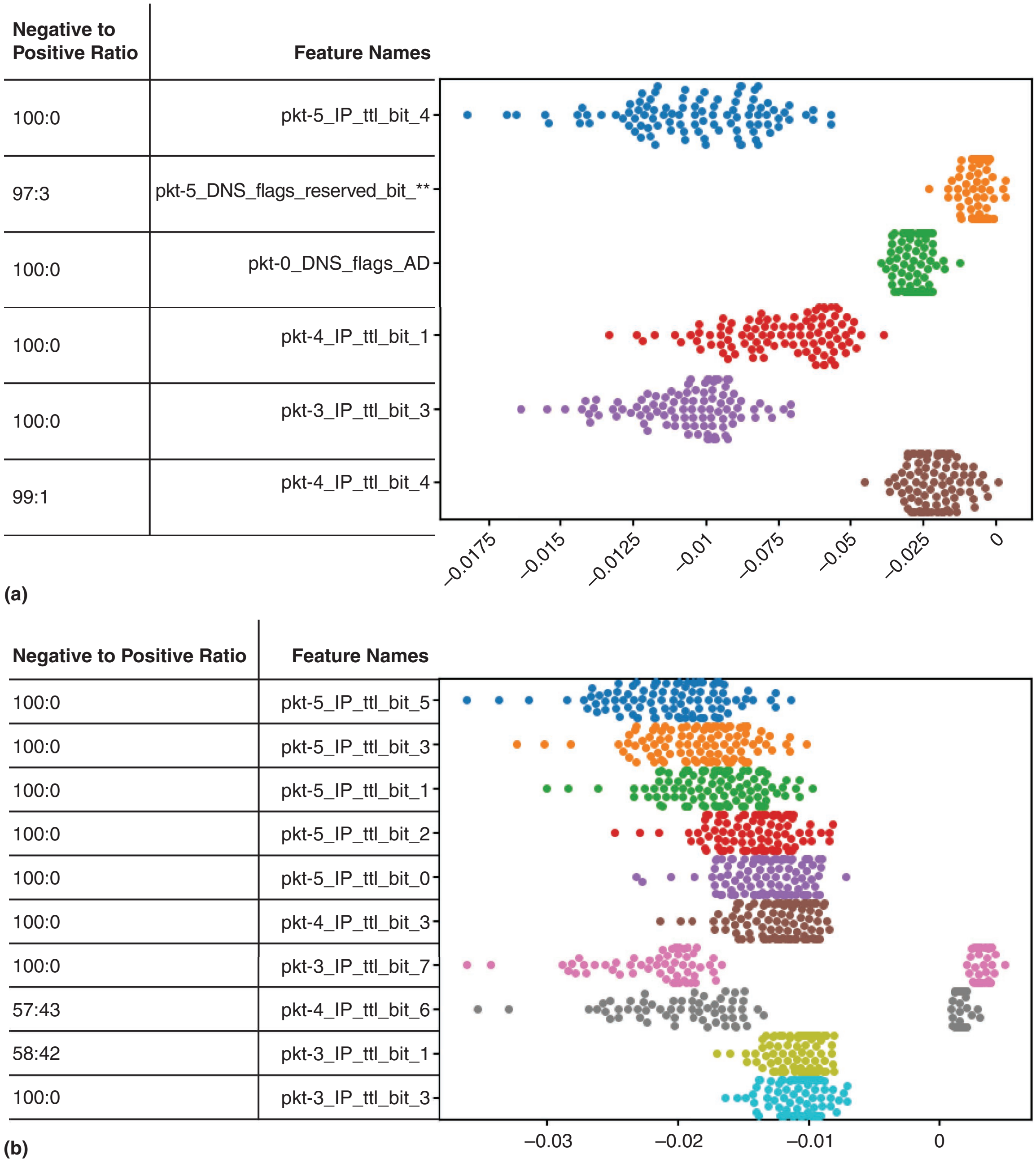
TABLE 1. Shapley values for an example benign data sample.

Positively-contributing features	Shapley values	Negatively-contributing features	Shapley values
pkt-4_DNS_authority-RR_bit_1	0.008 292	pkt-4_IP_ttl_bit_6	−0.025 208
pkt-5_DNS_authority-RR_bit_1	0.006 453	pkt-4_IP_ttl_bit_7	−0.023 393
pkt-3_UDP_src-port_bit_15	0.005 856	pkt-5_IP_ttl_bit_5	−0.022 140
pkt-5_IP_total-len_bit_1	0.005 790	pkt-5_IP_ttl_bit_1	−0.020 732
pkt-3_DNS_flags_reserved_bit_**	0.004 204	pkt-5_IP_ttl_bit_3	−0.018 543
pkt-5_UDP_len_bit_7	0.003 944	pkt-1_UDP_len_bit_3	−0.016 521
pkt-4_UDP_len_bit_7	0.003 764	pkt-5_IP_ttl_bit_0	−0.016 137
pkt-4_IP_total-len_bit_1	0.003 690	pkt-4_IP_ttl_bit_3	−0.015 278
pkt-4_UDP_len_bit_5	0.003 503	pkt-5_IP_ttl_bit_2	−0.015 069
pkt-4_IP_total-len_bit_5	0.003 455	pkt-1_DNS_flags _non-auth	−0.014 437

First, since the entire subset corresponds to the same decision-making path in the DT, the DT explanations "tell" a domain expert that the subset of the data samples are classified based on the features on the corresponding root-node-to-leave-node path in the DT. We call this set of features *Set 1* features. Second, since SHAP can provide the Shapley values of each contributing feature in classifying each member of the subset of data samples, we may differentiate all the involved features based on whether a feature contributes to the decision-making of majority of the data samples. We call this set of features *Set 2* features. By neglecting the features that only contribute to the decision-making of a minority of the data samples, we reduce the risk introduced by SHAP sometimes providing misleading explanations for some particular data samples. Third, if *Set 1* is a subset of *Set 2*, we say that expert-comprehensible explanations are not conflicting with the inner workings of the black-box model; if *Set 2* has quite a few members that are not in *Set 1*, we say that expert-comprehensible explanations are not completely reflecting the inner workings of the black-box model.

Specifically, we aggregate Shapley local interpretations of multiple data samples to simulate global interpretations. Shapley values of multiple data samples are added up with respect to every feature, and then we find the top positively-contributing features and top negatively-contributing features, similar to what is presented in Table 1. Due to time and computing resource constraints, it is not feasible to apply Shapley to all data samples. Instead, we decided to look into a specific group of data samples. The selected group of data samples includes only benign data samples, corresponding to the leaf node with the most benign data samples as shown in Figure 1 (referred to as major benign path). We randomly chose 100 data samples from the group, upon which Shapley local interpretation is applied to save computation resources. We have done this multiple times and found that they all show similar results. For example, they all choose the same most important features, and SHAP's ranking of them is also the same. In the following paragraphs, we will only show two result charts for simplicity.

Figure 2(a) shows the Shapley values for features shown in the major benign path. If the Shapley value is negative, it means that the value for this feature is pushing the classification result to be benign; if the Shapley value is positive, it means that the classification result is being pushed to be malicious. For every data sample, one point will be added to every feature because there will be one Shapley value for every feature. To the left of the figure, we also show the ratio of negative data points to positive data



Negative to Positive Ratio	Feature Names
100:0	pkt-5_IP_ttl_bit_4
97:3	pkt-5_DNS_flags_reserved_bit_**
100:0	pkt-0_DNS_flags_AD
100:0	pkt-4_IP_ttl_bit_1
100:0	pkt-3_IP_ttl_bit_3
99:1	pkt-4_IP_ttl_bit_4

Negative to Positive Ratio	Feature Names
100:0	pkt-5_IP_ttl_bit_5
100:0	pkt-5_IP_ttl_bit_3
100:0	pkt-5_IP_ttl_bit_1
100:0	pkt-5_IP_ttl_bit_2
100:0	pkt-5_IP_ttl_bit_0
100:0	pkt-4_IP_ttl_bit_3
100:0	pkt-3_IP_ttl_bit_7
57:43	pkt-4_IP_ttl_bit_6
58:42	pkt-3_IP_ttl_bit_1
100:0	pkt-3_IP_ttl_bit_3

FIGURE 2. Beeswarm charts for 100 randomly selected data samples in the major benign path of TRUSTEE. (a) Shapley results for features selected by TRUSTEE. (b) Shapley results for other important features.

points: the number of data points with negative Shapley values to the number with positive Shapley values. It can be inferred from this figure that most Shapley values are negative, meaning that Shapley local interpretation agrees with TRUSTEE that, for the selected data samples, these six features' values are pushing them toward being benign.

Figure 2(b) shows the Shapley values for features that are most negatively affecting. By most negatively affecting, we mean that these features have the smallest summation of Shapley values of all randomly selected data samples from the major benign path. Feature pkt-3_IP_ttl_bit_3 is identified by Shapley as one of the top-10 most negatively-contributing features, but all other important features chosen by TRUSTEE are not in the top-10 list. Clearly, Shapley thinks that there are features that have larger negative impact on data samples' classification results, more impactful than the six features chosen by TRUSTEE. This is probably due to the fact that TRUSTEE and Shapley local interpretation inspect different amount of data samples. The TRUSTEE DT is built by inspecting all data samples, but Shapley local interpretation only inspects a subset of data samples. Specifically, the DT shown in Figure 1 inspects 18,644 data samples, of which 9,419 are benign data samples, and 9,215 are malicious data samples. However, Shapley local interpretation results shown in Figure 2(a), and (b) are from 100 data samples randomly sampled from the leftmost leaf node in Figure 1, where there are only 8,876 benign data samples. Clearly, the two results are based on different distributions of data samples. Another reason might be that TRUSTEE and Shapley's "important features" do not have the same meaning. TRUSTEE's important features are "important" because these features can effectively discriminate benign data samples from malicious ones, while Shapley's important features are "important" because these features' values contribute the most to the data samples' classification results. These two sets of "important features" are important in different aspects, so it might be reasonable that the two sets include different features.

These two sets of "important features" are important in different aspects, so it might be reasonable that the two sets include different features.

Main finding

On the one hand, expert-comprehensible DT explanations do not conflict with the inner workings of the black-box model; on the other hand, expert-comprehensible DT explanations are not completely reflecting the inner workings of the black-box model.

This article compares two XAI techniques, TRUSTEE and SHAP, in explaining the neutral networks. The results show that different explanation methods may not fully agree with each other at some points. Differences may stem from different explanation mechanisms, the choice of data sample subsets, or different perspectives of the explanations. In the future work, we will investigate the potential causes for such misalignment between explanations provided by the DTs and SHAP. C

ACKNOWLEDGMENT

This article is not subject to copyright in the United States. Commercial products are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the identified products are necessarily the best available for the purpose. Peng Liu was supported by NIST 60NANB22D144, NSF CNS-2019340, and NSF ECCS-2140175. Xiaoyan Sun was supported by NSF DGE-2105801. The corresponding author is Peng Liu.

REFERENCES

1. Q. Zou, A. Singhal, X. Sun, and P. Liu, "Deep learning for detecting logic-flaw-exploiting network attacks: An end-to-end approach," *J. Comput. Secur.*, vol. 30, no. 4, pp. 541–570, 2022, doi: 10.3233/JCS-210101.
2. E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowl. Inf. Syst.*, vol. 41, no. 3, pp. 647–665, 2014, doi: 10.1007/s10115-013-0679-x.
3. M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 13–17, 2016, pp 1135–1144, doi: 10.1145/2939672.2939778.
4. P. J. Phillips et al., "Four principles of explainable artificial intelligence," National Institute of Standards and Technology, Gaithersburg, MD, USA, NISTIR 8312, 2021. Accessed: Nov. 29, 2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8312.pdf?trk=public_post_comment-text
5. V. Arya et al., "One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques," 2019, *arXiv:1909.03012 [cs, stat]*.
6. O. Bastani, C. Kim, and H. Bastani, "Interpreting blackbox models via model extraction," 2017, *arXiv:1705.08504*.
7. H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Interpretable & explorable approximations of black box models," 2017, *arXiv:1707.01154*.
8. O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable reinforcement

learning via policy extraction," in *Proc. 32nd Int. Conf. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 2499–2509, doi: 10.5555/3327144.3327175.

9. S. Jacobs, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, and L. Z. Granville, "AI/ML for network security: The emperor has no clothes," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, 2022, pp 1537–1551, doi: 10.1145/3548606.3560609.
10. K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013, *arXiv:1312.6034*.
11. A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3145–3153, doi: 10.5555/3305890.3306006.
12. S. Lapuschkin, A. Binder, G. Montavon, K.-R. Müller, and W. Samek, "The LRP toolbox for artificial neural networks," *J. Mach. Learn. Res.*, vol. 17, pp. 3938–3942, Jan. 2016.
13. M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3319–3328, doi: 10.5555/3305890.3306024.
14. P. K. Mudrakarta, A. Taly, M. Sundararajan, and K. Dhamdhere, "Did the model understand the question?" 2018, *arXiv:1805.05492*.

QINGTIAN ZOU is a Ph.D. student at The Pennsylvania State University, State College, PA 16803 USA. Contact him at qzz32@psu.edu.

LAN ZHANG is an assistant professor at Northern Arizona University, Flagstaff, AZ 86011 USA. Contact her at lfz5092@psu.edu.

XIAOYAN SUN is an associate professor with the Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA 01609 USA. Contact her at xsun7@wpi.edu.

ANOOP SINGHAL is a senior computer scientist in the Computer Security Division, National Institute of Standards and Technology, Gaithersburg, MD 20899 USA. He is a Life Senior Member of IEEE. Contact him at anoop.singhal@nist.gov.

PENG LIU is the Raymond G. Tronzo, MD Professor of Cybersecurity, and director of the Cyber Security Lab, The Pennsylvania State University, State College, PA 16803 USA. Contact him at pxl20@psu.edu.

Digital Object Identifier 10.1109/MC.2024.3386242