Toward Explainable Deep Neural Network based Anomaly Detection

Kasun Amarasinghe[1], Kevin Kenney[2], Milos Manic[1]
Virginia Commonwealth University, Richmond, Virginia, USA[1]
Idaho National Laboratory, Idaho Falls, Idaho, USA[2]
amarasinghek@vcu.edu, kevin.kenney@inl.gov, misko@ieee.org

***Abstract*—Anomaly detection in industrial processes is crucial for general process monitoring and process health assessment. Deep Neural Networks (DNNs) based anomaly detection has received increased attention in recent work. Albeit their high accuracy, the black-box nature of DNNs is a drawback in practical deployment. Especially in industrial anomaly detection systems, explanations of DNN detected anomalies are crucial. This paper presents a framework for DNN based anomaly detection which provides explanations of detected anomalies. The framework answers the following questions during online processing: 1) "why is it an anomaly?" and 2) "what is the confidence?" Further, the framework can be used offline to evaluate the "knowledge" of the trained DNN. The framework reduces the opaqueness of the DNN based anomaly detector and thus improves human operators' trust in the algorithm. This paper implements the first steps of the presented framework on the benchmark KDD-NSL dataset for Denial of Service (DoS) attack detection. Offline DNN explanations showed that the DNN was detecting DoS attacks based on features indicating destination of connection, frequency and amount of data transferred while showing an accuracy around 97%.**

Keywords— Deep Learning; Deep Neural Networks; Explainable AI; Layer wise Relevance Propagation; Anomaly Detection;

I. INTRODUCTION

Modern industrial processes are continuously growing in complexity, structure and degree of automation [1]. With this increasing complexity, reliability and security of systems are a concern and thus continuous monitoring of the systems is essential [1]. In system monitoring, identifying anomalous behavior in data is crucial [2]. This process, named anomaly detection and novelty detection has many forms such as network intrusion detection, fault detection and condition monitoring [2]. As a result, research on anomaly detection methodologies have been a popular topic in the last few decades.

One of the major approaches is model-based techniques [3]–[5]. However, the drawback of model based techniques is that they require a priori knowledge and the mathematical knowledge of the system. Therefore, data-driven process monitoring techniques are attractive alternatives as they require minimal a priori knowledge about the system [1]. In data driven techniques, traditionally, "shallow" Artificial Neural Networks (ANNs) and Support Vector Machines (SVM) have been very popular in anomaly detection applications [6], [7]. ANNs have been successfully used in anomaly detection in many applications such as intrusion detection [8]–[10] and hardware component fault detection [11]. SVMs, especially one-class SVMs have been a very popular approach in many anomaly detection approaches [12]–[14]. Further, methods such as Random Forests have been successfully applied to anomaly detection [15].

In recent years, Deep learning or Deep Neural Networks (DNNs) gained immense popularity [16]–[18]. DNNs learn features with multiple layers of abstraction and thus are capable of modeling complex patterns [19], [20]. Therefore, naturally DNNs based anomaly detection methodologies have received increased attention in the last few years. Methods such as Restricted Boltzmann Machines, Autoencoders, Long Short Term Memories, Deep Multi-Layer Perceptrons and Convolutional Neural Networks have been proposed in the literature [21]–[29]. Despite their impressive accuracies, DNNs are still used as black boxes due to their opaque highly non-linear structure [30]. This poses a limitation in practical applications, especially in mission critical systems where it is important to have insight about DNN predictions. It is important to verify whether DNN's high accuracy is due to the correct reasons or by exploiting special artifacts in the dataset [31]. Further, these mission critical systems largely employ humans-in-the-loop for controls and monitoring and when machine learning based decision support systems are deployed, the human trust on the machine learning models is extremely crucial.

This paper presents a framework for explainable DNN based anomaly detection. The presented framework provides explanations of the DNN based anomaly detector. In the proposed framework, when the DNN detects an anomalous event, in addition to the prediction, the system provides the user with the following: 1) The confidence of the prediction, 2) A textual description of the detected anomaly and 3) the factors which was relevant in making the prediction. This framework enables the user to get justifications of the DNN decisions in addition to the decision itself, i.e. the user gets to see whether the DNN is "doing the right thing for the right reasons". During online processing, when an anomaly is detected, the user can obtain a justification from the DNN which answers the questions "why is it an anomaly?" and "how certain are you that it's an anomaly?" Further, a domain expert can use the framework offline to evaluate DNNs "knowledge" and make decisions about deployment. In addition to the framework, this paper presents an initial implementation of the framework. The offline knowledge evaluation is discussed in the implementation.

The rest of the paper is organized as follows. Section II elaborates on the concept of interpretability/ explainability of DNNs. Section III presents the framework we propose and outlines the methodology of the initial implementation of the

978-1-5386-5024-0/18/$31.00 ©2018 IEEE

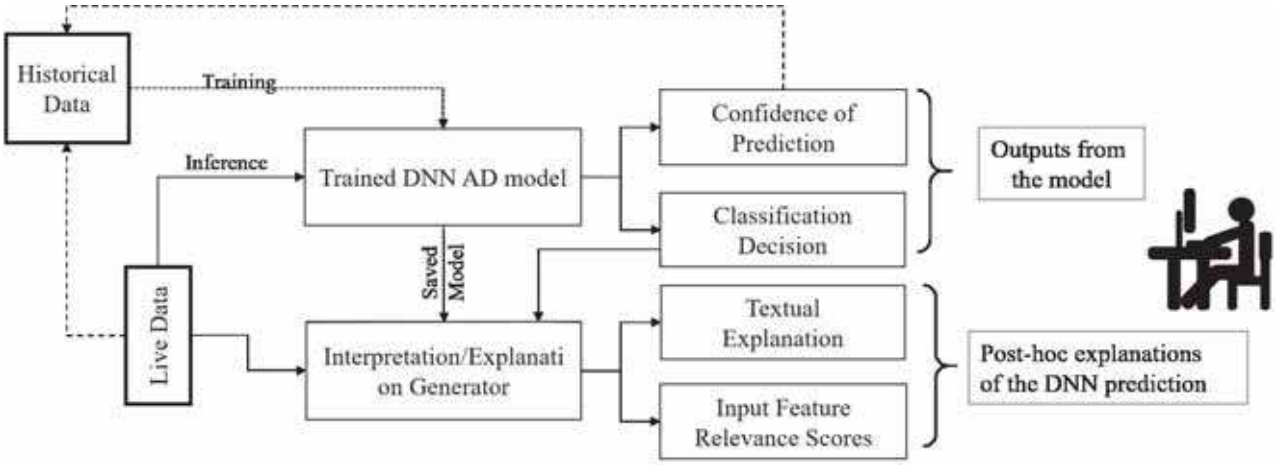

Figure 1: The Presented framework for Explainable DNN based Anomaly detection. The decision maker is the human operator. The operator gets the DNN predictions and the justifications.

framework, Section IV presents the details about the experiments conducted and their results, and finally Section V concludes the paper.

II. Explainable Deep Neural Networks

This section elaborates on the concepts of explainability and interpretability of DNNs and overviews the existing consensus on the taxonomy.

Gunning identifies an explainable machine learning system as one that answers the following questions. 1) "Why did it do that?" 2) "Why didn't it do something else?" 3) "When does it succeed?" 4) "When does it fail?" 5) "When can it be trusted?" and 6) "how can an error be corrected?" [32]. Improving interpretability of Deep Neural Networks remain to be an open research area. The terms interpretability and explainability are used interchangeably in the domain and Lipton pointed out that a formal definition on what an interpretable DNN is needed [33]. The only formal definition of interpretability and explainability is provided in [31]. Montanvon et al. propose a distinction between the terms "explanation" and "interpretation" in the context of DNNs. Since it is out of the scope of this paper, we will be using the words interchangeably and in the context of this paper, "interpretation" or "explanation" would infer shedding light into the DNNs predictions. Despite the lack of a formal definition, DNN interpretability can be categorized into two broad categories,: 1) model transparency and 2) model functionality [33], [34].

Transparency of the model refers to understanding what the network has learned and the reasons behind the concepts it has learned. Transparency can be viewed in three parameters: 1) decomposability, 2) simulatability and 3) algorithmic transparency[34]. Decomposability is whether there is an intuitive explanation for the model parameters. Algorithmic transparency relates to the ability to explain the inner workings of the learning algorithm. Simulatability refers to the ability of a human using the input data together with the model to reproduce every calculation that's necessary to make the prediction, allowing a human to understand the changes in the model parameters during the training process. Given the complexity of DNNs, achieving these three components is not a trivial task. Further, it is assumed that the simulatability is very low in DNNs and hence most of the research is focused on improving decomposability and algorithmic transparency [34].

Model functionality explanations can be used to explain predictions by the model. This facet of interpretable DNNs is also called post-hoc explanation generation [33]. Post-hoc explanation generation entails understanding a pre trained model, i.e. the trained model is available and methods attempt to gain a functional understanding of the trained model [31]. Post-hoc explanations can be generated in four different ways. First method is to provide textual justifications of the DNN predictions. This involves providing a semantically meaningful description of the model's output and the reasons behind the output. Therefore, it requires a combination of models. Second method is to provide justifications through different visualizations of parameters. Third, local explanations are used to gain insight on the model's behavior. For instance, in DNN's the gradient of the output with respect to the inputs can be used to identify the local changes that are influenced by the input vector [35]. The focus of this paper is on post-hoc explanations for DNN based anomaly detection algorithms.

III. Explainable Deep Neural Networks based Anomaly Detection

This section presents a framework for DNN based anomaly detection and explanation of DNN based decisions. When using DNNs for process monitoring in mission critical systems such as critical infrastructure security, interpretability is almost as important as the prediction accuracy. Therefore, the deployed DNN models should be able to provide explanations of their predictions. In this paper, we present a framework for explainable DNNs for process monitoring. Figure 1 shows the presented framework.

The focus of presented framework is on generating post-hoc explanations for the DNN predictions. Therefore, in addition to the DNN prediction the following outputs are generated to

improve the user's trust on the DNN prediction: 1) relevance of input features for the DNN prediction, 2) the confidence of estimation, and 3) textual justification. Therefore the user has the capability of validating the DNNs decisions. For instance, when an anomaly is detected, the framework will provide the user 1) the confidence the DNN has on the estimation (a probability score or a fuzzy membership grade), 2) input feature relevance scores, indicating what input features "drove" the DNNs decision and 3) textual description of the anomaly (can be a preset description of the anomaly type, or an IF-THEN type linguistic summary). Further a textual summary can be given to the user summarizing all the aspects: e.g. *DoS Attack WITH high confidence BECAUSE "connections to same host in last 2 seconds" is high*.

In this paper, the first steps of the framework implementation is presented. A feed forward DNN is used to identify anomalies in a data stream and input feature relevance is calculated for classification decisions. In order to validate the "knowledge" of the DNN, the relevance scores for each class is extracted. It has to be noted that in a practical scenario, this process is carried out offline, i.e. prior to deployment of the DNN. First, the DNN based anomaly detection method is presented. Then, the input feature relevance calculation is presented.

A. Deep Neural Networks based Anomaly Detection

In this paper, anomaly detection is carried out using a supervised DNN, i.e. the DNN is trained with labeled data.

An input record can be considered as $X \in \mathbb{R}^d$, where each input patter X is composed of a set of input features: $X = \{x_d\}$ where *d* denotes the d[th] feature in the dataset. In addition to the input features, each X is associated w ith its label Y. Therefore, the function of the DNN is a classification function where the mapping learned by the DNN can be expressed as $f: \mathbb{R}^d \rightarrow \mathbb{R}^+$.

The DNN consists of an input layer, an output layer and a multiple hidden layers of neurons. Each hidden layer neuron is activated as:

$$a_j^{(l+1)} = g(\sum_i a_i^{(l)} w_{ij}^{(l,l+1)} + b_j^{(l+1)}) \quad (1)$$

where, $a^{(l)}$ is the activation of the l[th] layer, $w_{ij}^{(l,l+1)}$ is the weight of the connection between i[th] neuron in layer *l* and j[th] neuron in layer *l+1* and $b_j^{(l+1)}$ is the bias of the j[th] neuron in layer *l+1*. Rectified Linear Unit (ReLu) neurons are used in the hidden layers, i.e. $g(\cdot)$ can be expressed as:

$$g(x) = \max(0, x) \quad (2)$$

The output layer is a softmax layer to obtain the probability distribution across the classes. Therefore, for each class the probability is calculated using the softmax function as follows:

$$P(Y = y_i|X) = \frac{e^{a_i}}{\sum_j e^{a_j}} \quad (3)$$

where the input values (a_i) are calculated using eq.(1) with $g(\cdot)$ being a linear pass through function. The cross entropy loss is minimized in the training of the DNN. The optimization process is carried out with a gradient based optimizer together with error back propagation.

B. Calculation of Input Feature Contributions to DNN Predictions

The focus of this work is generating post-hoc explanations for the DNN. In this paper, the explanations of the DNN is provided in terms of input feature relevance scores that indicate the contribution each input feature made to the detected anomaly. This results in a quantitative measure as to how much influence a certain feature had in the DNN's predictions, which helps the user understand what input features contributed to decisions that the DNN made. In this paper, the input feature relevance is calculated by decomposing the composite function of the DNN using a method named Layer-wise Relevance Propagation (LRP). It has to be noted that this section assumes that the DNN architecture takes the form described in the subsection above.

Layer-wiser relevance propagation (LRP) was introduced by Bach et al. as an approach for understanding the contribution of each pixel to image classification decisions made by the DNN [36]. LRP in its general form assumes that the classification algorithm can be decomposed into several layers of computation. The first layer is considered as the inputs and the last layer is the real-valued prediction of the classifier. It is assumed that each dimension of each layer has a relevance score $(R_d^{(l)})$ where *d* is the dimension and the *l* is the layer. The idea is to find the relevance scores for the layer *l* when the relevance scores for layer *(l+1)* is available. The decomposition is carried out so that the following rule of the conservation holds.

$$f(x) = \cdots = \sum_{d \in l+1} R_d^{(l+1)} = \sum_{d \in l} R_d^{(l)} = \cdots = \sum_d R_d^{(1)} \quad (4)$$

Where $f(x)$ is the output and $R_d^{(1)}$ indicates the relevance score of the d[th] dimension of the input layer.

In calculating these relevance scores, the multilayered architecture of the DNN can be leveraged and the relevance scores can be propagated in a backward pass, i.e. the relevance scores of a lower level can be expressed as a function of upper level relevance scores. The relevance scores are back-propagated in "messages", $R_{i \leftarrow j}$ (from neuron *j* in *l+1* to neuron *i* in *l*) such that, the relevance conservation property holds as follows.

$$\sum_i R_{i \leftarrow j}^{(l,l+1)} = R_j^{(l+1)} \quad (5)$$

In the same way, the relevance score for i[th] neuron in the *l* layer can be expressed as:

$$R_i^{(l)} = \sum_j R_{i \leftarrow j}^{(l,l+1)} \quad (6)$$

This relevance score distribution can be done based on the ratio of pre-activations as follows:

TABLE 1: FEATURES OF KDD-NSL DATASET. THIS NOT THE COMPLETE SET OF FEATURES. ONLY A SUBSET OF FEATURES ARE PRESENTED

No.	Feature Name	Feature Type	Description
1	Duration	Basic features	Time length of connection
2	Protocol type		Protocol used in the connection
3	Service		Destination network service used
4	Flag		Status of the connection
5	Src_bytes		Number of bytes transferred from source to destination in a single connection
6	Dst_bytes		Number of data bytes transferred from destination to source in single connection
7	Land		Whether the source and destination port numbers are the same
8	Wrong fragment		Total number of total fragments in the connection
9	Urgent		Number of urgent packets in the connection
23	Count	Time related traffic features	Number of connections to the same destination as the current connection's destination in the past two seconds
24	Srv_count		Number of connections to the same port as the current connection's port in the last two seconds
25	Serror_rate		The percentage of connections that have activated the one or more of flags S0-S3 among the connections aggregated in count (23)
26	Srv_error_rate		The percentage of connections that have activated the one or more of flags S0-S3 among connections aggregated in Srv_count (24)
27	Rerror_rate		Percentage of connections that have Flag REJ activated in among the connections aggregated in count (23)
28	Srv_error_rate		Percentage of REJ activated among connections aggregated in srv_count (24)
32	Dst_host_count	Host based traffic features	Number of connections with the same destination IP

$$R_{i\leftarrow j}^{(l,l+1)} = \left(\frac{a_i^{(l)} w_{ij}^{(l,l+1)}}{\sum_i a_i^{(l)} w_{ij}^{(l,l+1)} + b_j^{(l+1)}}\right) \cdot R_j^{(l+1)} \tag{7}$$

The major drawback of the above propagation rule is that if the activations of the neuron (the denominator) is small, the relevance scores can get unboundedly large. To bound the scores, the αβ method [36] is used. The αβ method can be expressed as follows:

$$R_{i\leftarrow j}^{(l,l+1)} = \left(\alpha \frac{a_i w_{ij}^{+}}{\sum_i a_i w_{ij}^{+} + b_j^{+}} + \beta \frac{a_i w_{ij}^{-}}{\sum_i a_i w_{ij}^{-} + b_j^{-}}\right) \cdot R_j^{(l+1)} \tag{8}$$

Where, $a_i w_{ij}^{+}$ and b_j^{+} are the positive portion of the activations and the negative portion is indicated by "-". Please note that the super scripted layer notations have been stripped off to simplify the notation and still *i* and *j* are considered to be indices associated with layers *l* and *l+1* respectively. Therefore, when relevance scores are propagated across layers to the input layer, the relevance score of each input feature *d*; $R_d^{(1)}$ can be obtained. For each DNN prediction, the relevance scores can be obtained.

In this study, we use these relevance measures offline for providing insight to the user about the trained model. The mean relevance of each input feature for each anomaly class is reported to the user. These average relevance scores show the input features which drove the DNNs decision when the specific anomaly class was detected. The average relevance that each feature had per class is calculated as follows:

$$R_c^d = \frac{1}{N_c} \sum_k^{N_c} R_{x_{kd}}^{(1)} \tag{9}$$

Where R_c^d is the average relevance of the d^{th} feature for class *c*, N_c. is the number of data points in the labeled data set belonging to class *c*, $R_{x_{kd}}^{(1)}$ is the relevance of the dth dimension of the kth data record.

IV. EXPERIMENTS

This section elaborates the experimental setup, and the results obtained from the experiments. First, the dataset used in the study is presented. Then, the experimental results are presented and discussed.

A. NSL-KDD Dataset

KDD Cup 1999 Dataset (KDD99) was used for the Third International Knowledge Discovery and Data Mining Competition [37]. The KDD dataset is a network intrusion detection dataset. The NSL-KDD dataset is a modified version of the benchmark KDD 99 dataset. The NSL-KDD was proposed to remove the issues the KDD dataset contained such as redundant records.

Each data record in the NSL-KDD dataset is a "connection". A connection is defined as a sequence of TCP packets recorded within a well-defined time window between a well-defined source and a destination IP address. Each connection is labeled [38]. NSL-KDD dataset comprises of separate train and test sets. The training dataset contains 21 different types of attacks. The test set contains 37 types of attacks. In order to make the task more realistic, the test dataset is not from the same probability distribution as the train set and test set has more attack types that

TABLE 3: THE DNN ARCHITECTURES TESTED

Model Name	Hidden Layer Architecture
3HL_Config-1	(256, 128, 64)
3HL_Config-2	(100, 100, 100)
4HL_Config-1	(256, 128, 64, 32)
4HL_Config-2	(100, 100, 100, 100)

doesn't appear in the training data. All the attacks in the dataset can be grouped into four higher level attack categories: 1) Denial of Service (DoS), 2) Probe, 3) U2R, and 4) R2L. This study only focuses on the DoS attacks, i.e. the DNN based anomaly detection algorithm performs a binary classification in the presented work.

The dataset contains 41 features in total that can be categorized as 1) basic features, 2) content related features, 3) time related traffic features and 4) host based traffic features. These features were derived from the raw data by Stolfo et al. [38] Table 1 lists a selected feature set along with their description, since the discussion is formed around these features. The complete feature set is not presented due to space constraints.

B. Experimental Results

Experimentation was carried out to assess the DNNs on two fronts: 1) classification accuracy (ability to make the right decision), and 2) interpretability (the ability to make the right decision for the right reasons). The goal of the experiments was to get insight into what the DNN was learning and produce post-hoc explanations of its predictions. Using the method described in Section III, the relevance of each input feature for each classification decision was calculated and averaged across the two classes. These relevance scores are used as means to interpret the DNN predictions.

As mentioned, a two class problem was considered and the DNN based classifier was trained to distinguish between normal communication and DoS attacks. Deep Feed Forward Networks were used in the experiments. Several test configurations were used to observe the relevance of input features on the classification decisions. More specifically, experimental set up was changed by changing the number of features used for the classification and by changing the DNN architecture. The changes in input relevance to classification was observed for these different test cases. The DNN architecture was changed in-terms of the depth and number of total neurons. DNN architectures that were used are given in Table 3. Two different feature sets were considered: 1) The basic traffic features and 2) the complete 41 features.

Classification accuracies across models were evaluated for the different test configurations. For the basic set of features (Features 1-9 in Table 1), models were able to produce predictions with accuracies ~93% on the test dataset. (See Table 2). Then, the complete feature set was used to perform the classification with the same DNN architectures as the previous test. In this test, it was noticed that all the DNNs improved their accuracies to ~97% on the test dataset. Therefore, it can be seen that the classification accuracy was improved slightly with the augmented feature set.

Relevant input features were compared across models for the test configurations. For the basic features, it was noticed that all

TABLE 2: CLASSIFICATION ACCURACIES OBTAINED BY DIFFERENT DNN ARCHITECTURES FOR THE TWO FEATURE SETS

Model	Basic Features		Complete Feature set	
	Train Accuracy (%)	Test Accuracy (%)	Train Accuracy (%)	Test Accuracy (%)
3HL_Config-1	95.9889	88.3191	98.6217	96.1572
3HL_Config-2	89.3754	**93.7852**	97.442	**97.2372**
4HL_Config-1	92.0553	93.4274	97.5925	97.1113
4HL_Config-2	86.8467	85.8743	97.1882	97.0318

the DNN models "agreed" on the important features for detecting a DoS attack. For DoS detection models gave a higher relevance to features No 03 and 04, the destination network service and the status of the connection respectively (see Figure 2). When using the complete feature set, all the models seemed to "agree" on the most relevant features for identifying "DoS". It was observed that the DNNs assigned a higher relevance to features that indicated the number of connection to the same destination in a short period of time, number of connections to the same IP and the number of bytes transferred in a single connection (features 05, 23, 32 in Table 1). See Figure 3 for the relevance scores comparison.

Therefore, experimental results showed that all the DNN models were able to achieve high classification accuracies on the KDD-NSL dataset on the two classes. However, the most important takeaway from the experiments is the insight that can be gleaned about the DNN decision making process from the LRP based input relevance scores. For example, when comparing the two feature sets used, the classification accuracies achieved had relatively small difference (~3%). However, it can be argued that when using the basic features, despite the high classification accuracy, the features that the DNN gave high relevance to when identifying DoS doesn't help a domain expert to validate the model. The DNNs can be modeling the artifacts in this specific dataset. However, when using the complete feature set, the features that were considered by the DNNs were, number of connections to the same destination and the frequency of connections to the same host and number of data transferred. These are factors directly related with a DoS attack. Therefore, these "valid reasons for classification", enables the human operator to validate the "knowledge" of the DNN. As a result, black box nature of the DNN based Anomaly Detection can be reduced. As a corollary of this, the human operator's trust on the anomaly detection algorithm increases since they have a way of evaluating whether the DNN based anomaly detection is "doing the right thing for the right reasons".

V. CONCLUSIONS AND FUTURE WORK

This paper presented a framework for explainable DNNs based anomaly detection in process monitoring. In the presented framework, post-hoc explanations of DNN predictions are presented to the user along with the prediction. This paper presented the initial implementation of the framework where a DNN based anomaly detection algorithm is developed on a network intrusion detection dataset. The DNN was trained in a supervised manner and the classification was carried out to identify DoS attacks from normal communication. Several DNN

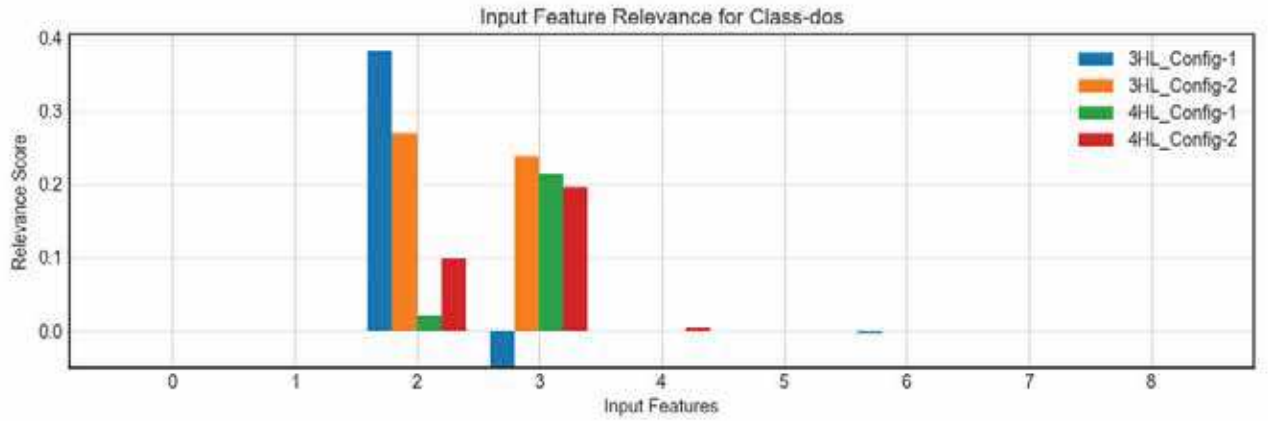

Figure 2: Input feature relevance scores (DoS class) when DNN is trained with the 9 basic features. It can be seen that features 3 (service) and 4 (flag) receive the highest relevance in the DNN for DoS. Using this information, the operator can say that the DNN model needs to be improved since the "reasons for picking DoS" is not intuitive to the domain expert

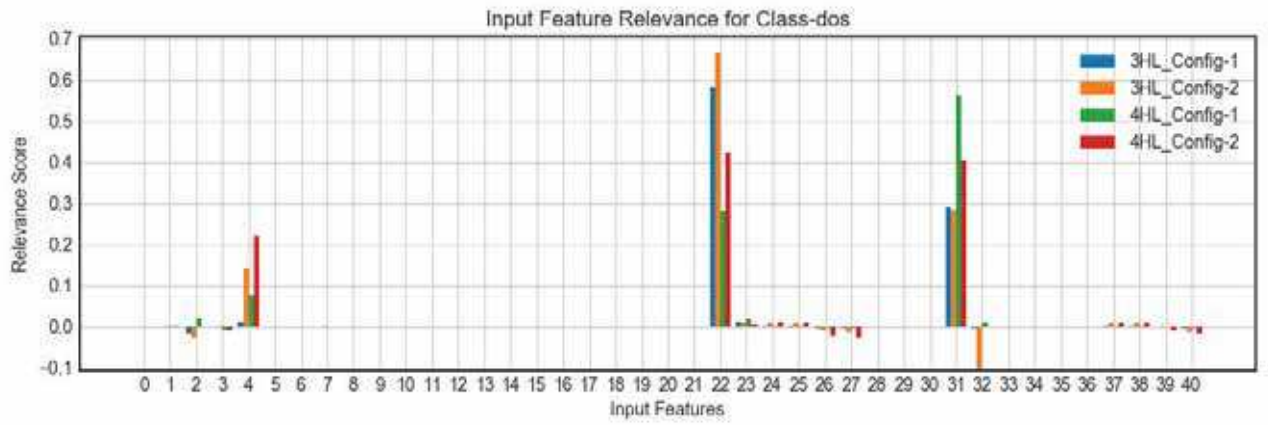

Figure 3: Input feature relevance scores (DoS class) when DNN is trained with the complete set 41 features. It can be seen that features 5 (Src bytes), 23 (count) and 32 (Dst_host_count) are given highest relevance by all the models. This information helps the domain expert to say that the DNN's performance is acceptable since the "reasons for picking DoS" is intuitive to the domain expert

architectures (all Deep MLPs) were experimented with and it was observed that DNNs were able to produce accuracies of 97% on the test dataset. The explanations of the DNNs were produced in terms of relevance scores for each input feature for each classification prediction. Experimental results showed that the DNNs gave a higher relevance to number of connections, connection frequency and amount of data transferred when classifying as "DoS". With this knowledge, a domain expert can intuitively assess the "knowledge" of the DNN based anomaly detection algorithm. The main implications of this work are the following: 1) provides a method for the user to interact with the DNN based anomaly detector, 2) builds human operator's trust on the behavior of the model and 3) provides a method for assessing the "knowledge" of the DNN based anomaly detection model. The next steps of this work are to provide confidence information about the DNN based anomaly detection predictions and provide textual justifications.

REFERENCES

[1] S. Yin, S. X. Ding, X. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Trans. Ind. Electron.*, vol. 61, no. 11, pp. 6414–6428, 2014.

[2] L. Aguayo and G. A. Barreto, "Novelty Detection in Time Series Using Self-Organizing Neural Networks: A Comprehensive Evaluation," *Neural Process. Lett.*, vol. 47, no. 2, pp. 717–744, Aug. 2017.

[3] H. Li, J. Yu, C. Hilton, and H. Liu, "Adaptive Sliding-Mode Control for Nonlinear Active Suspension Vehicle Systems Using T–S Fuzzy Approach," *IEEE Trans. Ind. Electron.*, vol. 60, no. 8, pp. 3328–3338, Aug. 2013.

[4] M. Blanke and J. Schröder, *Diagnosis and fault-tolerant control*. Springer, 2006.

[5] J. Seshadrinath, B. Singh, and B. K. Panigrahi, "Vibration Analysis Based Interturn Fault Diagnosis in Induction Machines," *IEEE Trans. Ind. Informatics*, vol. 10, no. 1, pp. 340–350, Feb. 2014.

[6] A. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surv. Tutorials*, vol. PP, no. 99, p. 1, 2015.

[7] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, pp. 1702–1707.

[8] O. Linda, T. Vollmer, and M. Manic, "Neural Network based Intrusion Detection System for critical infrastructures," in *2009 International Joint Conference on Neural Networks*, 2009, pp. 1827–1834.

[9] J. Ryan, M.-J. Lin, and R. Miikkulainen, "Intrusion Detection with Neural

Networks."

[10] R. P. Lippmann and R. K. Cunningham, "Improving intrusion detection performance using keyword selection and neural networks," *Comput. Networks*, vol. 34, no. 4, pp. 597–603, Oct. 2000.

[11] P. Bangalore and L. B. Tjernberg, "An Artificial Neural Network Approach for Early Fault Detection of Gearbox Bearings," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 980–987, Mar. 2015.

[12] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognit.*, vol. 58, pp. 121–134, Oct. 2016.

[13] R. Perdisci, G. Gu, and W. Lee, "Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems," in *Sixth International Conference on Data Mining (ICDM'06)*, 2006, pp. 488–498.

[14] Yanxin Wang, Johnny Wong, and A. Miner, "Anomaly intrusion detection using one class SVM," in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pp. 358–364.

[15] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust Random Cut Forest Based Anomaly Detection On Streams," 2016.

[16] S. Wang and J. Jiang, "Learning Natural Language Inference with LSTM," 2015.

[17] D. L. Marino, K. Amarasinghe, and M. Manic, "Simultaneous generation-classification using LSTM," in *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*, 2017.

[18] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with Deep Bidirectional LSTM," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 273–278.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[20] I. Goodfellow, B. Yoshua, and A. Courville, *Deep Learning*. MIT Press, 2016.

[21] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep Structured Energy Based Models for Anomaly Detection."

[22] C. Zhou and R. C. Paaenroth, "Anomaly Detection with Robust Deep Autoencoders."

[23] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection," Jul. 2016.

[24] A. M. Vartouni, S. S. Kashi, and M. Teshnehlab, "An anomaly detection method to detect web attacks using Stacked Auto-Encoder," in *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*, 2018, pp. 131–134.

[25] S. Chauhan and L. Vig, "Anomaly detection in ECG time signals via deep long short-term memory networks," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015, pp. 1–7.

[26] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016, pp. 130–139.

[27] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, Dec. 2013.

[28] S. Zhou, W. Shen, D. Zeng, M. Fang, Y. Wei, and Z. Zhang, "Spatial–temporal convolutional neural networks for anomaly detection and localization in crowded scenes," *Signal Process. Image Commun.*, vol. 47, pp. 358–368, Sep. 2016.

[29] J. R. Medel and A. Savakis, "Anomaly Detection in Video Using Predictive Convolutional Long Short-Term Memory Networks," Dec. 2016.

[30] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models," Aug. 2017.

[31] G. Montavon, W. Samek, and K.-R. Müller, "Methods for Interpreting and Understanding Deep Neural Networks," Jun. 2017.

[32] D. Gunning, "Explainable Artificial Intelligence (XAI) Explainable AI – What Are We Trying To Do ?," *Def. Adv. Res. Proj. Agency*, pp. 1–18, 2017.

[33] Z. C. Lipton, "The Mythos of Model Interpretability," Jun. 2016.

[34] S. Chakraborty *et al.*, "Interpretability of Deep Learning Models: A Survey of Results," *IEEE Smart World Congr. DAIS - Work. Distrib. Anal. Infrastruct. Algorithms Multi-Organization Fed.*, 2017.

[35] D. Baehrens *et al.*, "How to Explain Individual Classification Decisions," 2009.

[36] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS One*, vol. 10, no. 7, pp. 1–46, 2015.

[37] "KDD-CUP-99 Task Description." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/task.html. [Accessed: 23-Apr-2018].

[38] S. J. Stolfo, Wei Fan, Wenke Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: results from the JAM project," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2, pp. 130–144.