Anomaly Detection and Localization in NFV Systems: an Unsupervised Learning Approach

Seyed Soheil Johari*, Nashid Shahriar†, Massimo Tornatore‡, Raouf Boutaba*, Aladdin Saleh§
*David R. Cheriton School of Computer Science, University of Waterloo, {ssjohari | rboutaba}@uwaterloo.ca
†Department of Computer Science, University of Regina, nashid.shahriar@uregina.ca
‡Politecnico di Milano, massimo.tornatore@polimi.it
§Rogers Communications Canada Inc., aladdin.saleh@rci.rogers.com

***Abstract*—Due to the scarcity of labeled faulty data, Unsupervised Learning (UL) methods have gained great traction for anomaly detection and localization in Network Functions Virtualization (NFV) systems. In a UL approach, training is performed on only normal data for learning normal data patterns, and deviation from the norm is considered as an anomaly. However, it has been shown that even small percentages of anomalous samples in the training data (referred to as contamination) can significantly degrade the performance of UL methods. To address this issue, we propose an anomaly-detection approach based on the Noisy-Student technique, which was originally introduced for leveraging unlabeled datasets in computer-vision classification problems. Our approach not only provides robustness against training-data contamination, but also can leverage this contamination to improve anomaly-detection accuracy. Moreover, after an anomaly is detected, localization of the anomalous virtualized network functions in an unsupervised manner is a challenging task in the absence of labeled data. For anomaly localization in NFV systems, we propose to exploit existing local AI-explainability methods to achieve a high localization performance and propose our own novel AI-explainability method, specifically designed for the anomaly-localization problem in NFV, to improve the performance further. We perform a comprehensive experimental analysis on two datasets collected on different NFV testbeds and show that our proposed solutions outperform the existing methods by up to 22% in anomaly detection and up to 19% in anomaly localization in terms of F1-score.**

I. Introduction

Virtualization represents a revolutionary change in the networking industry, similar to the change brought in the computer industry in the 80's. A prolific application of virtualization in networking is Network Functions Virtualization (NFV). NFV allows decoupling network or service functions from the underlying hardware by implementing them as software appliances, called Virtual Network Functions (VNFs), on virtualized commodity hardware. Numerous existing deployments of VNFs have already shown the potential to achieve near-hardware performance and to provide ample opportunities for network optimization and cost reduction [1, 2]. Nonetheless, provisioning and managing VNF-based services introduce additional complexity due to dynamic network topologies, multiple layering, and lack of network visibility. This increased complexity makes VNFs more failure-prone than dedicated hardware-based solutions [3], [4], [5]. Therefore, detecting anomalous behavior in an NFV system and localizing its origin is of paramount importance to ensure high reliability for virtualized services.

978-1-6654-0601-7/22/$31.00

The complex inter-dependencies and multi-faceted fault characteristics in NFV systems render traditional anomaly-detection and localization approaches inefficient as they rely mostly on expert-based fixed thresholds [6], [7]. On the other hand, Machine Learning (ML) methods, in particular Deep Learning (DL) methods, have shown promising results in developing adaptive and efficient mechanisms to detect and localize potential anomalies in a dynamic NFV system by capturing hidden dependencies among a variety of performance metrics. However, most of these existing ML-based approaches utilize Supervised Learning (SL) algorithms that require abundant labeled faulty instances to achieve satisfactory performance, while labeled network faulty data is a scarce resource and generally unavailable in sufficient volumes for two main reasons: *i)* labeling data requires domain experts to annotate logs of anomalous scenarios, and *ii)* only a small amount of the monitored data from the NFV system is related to faulty scenarios [7].

UL-based anomaly detection on multi-dimensional data can help to circumvent the requirement of abundant labeled faulty instances. Other than reducing the reliance on labeled data and domain experts' knowledge, another benefit of a UL approach from an operational point of view is that it would not be biased on some specific failure scenarios and can provide a more generalized protection against many sorts of anomalous behavior in the NFV system. For example, UL methods based on Recurrent Neural Networks (RNNs) are well-known for anomaly detection on temporal data, however, training RNNs generally is a long and computationally expensive process that requires a large volume of training data [8]. Therefore, RNNs are not suitable for NFV environments with dynamic topologies where periodic software updates often change the system functionality, with the consequence that the ML models need to be often re-tuned with newly collected data [8].

Autoencoder is another popular UL method that generally trains faster and requires much less training data than RNNs to reach a satisfactory performance [8]. Autoencoders have been utilized for unsupervised anomaly detection in many network management tasks, including anomaly detection in NFV architectures [6], [9]. These methods assume the availability of a training dataset that purely consists of normal instances. However, it is often unavoidable that the historical

data collected from the NFV architecture includes a few anomalous samples, i.e., we expect our historical data to have some degree of contamination. Different studies have shown that even small percentages of contamination in the training data can significantly degrade the performance of Autoencoders in UL anomaly-detection methods [10], [11].

In this paper, we propose a novel unsupervised anomaly-detection approach for NFV systems for the case in which training data is contaminated. Inspired by the Noisy-Student [12] concept used in computer vision, we first train a Deep Autoencoding Gaussian Mixture Model (DAGMM) [11], (i.e., a UL anomaly-detection method) on the contaminated training data as the teacher model, initially treating the training data as it has no contamination. Then, we use DAGMM to remove potential anomaly instances (i.e., to clean the training data), and from these removed instances, we pseudo-label a few samples that DAGMM has classified as anomalies with very high confidence (pseudo-labeling [13] is the process of using a trained ML model to predict labels for unlabelled data). In this way, we compensate for some of the information that we potentially lose during the data cleaning process. The cleaned dataset and pseudo-labeled anomalies are then fed to the student model, which is our novel architecture consisting of an Autoencoder cascaded with a a semi-supervised anomaly-detection model called Deviation Network [14]. We also use data augmentation in feature space [15], a data augmentation method proposed for non-image data, to add noise to the extracted pseudo-labeled anomalies to improve the model's generalization.

Once anomaly detection is successfully achieved with the approach described above, we focus on developing an unsupervised approach for anomaly localization, i.e., localizing the anomalous VNF after an anomaly is detected. Localization is also a challenging task in a UL approach, as there might be no labeled anomalous instances in the training data, and distinguishing between different failure scenarios can only be done by comparing the detected anomaly with normal instances. In our paper, we utilize SHapley Additive exPlanations (SHAP) [16], a local AI-explainability method, to reach a high localization performance.

To summarize, the main paper contributions are:

- We propose a novel Noisy-Student-based unsupervised anomaly-detection approach that is robust against contamination (some anomalous samples) in the training data and can leverage this contamination to improve the anomaly-detection performance.
- We exploit SHAP, a local AI-explainability method, to achieve a high performance in the unsupervised anomaly-localization task. We also propose our novel AI-explainability method, specifically designed for the localization problem, to further improve the performance.
- We show the effectiveness of our proposed solutions by comprehensive experimental evaluation on two datasets from two different NFV testbeds.

The rest of the paper is organized as follows. Section II discusses the related work. Section III is a detailed description of our anomaly-detection and localization approaches. In Section IV, experimental analysis is presented based on datasets from two different NFV testbeds. Section V concludes the paper.

II. Related Work

A. ML-based Fault Detection and Diagnosis

Many works leveraged SL for fault detection and diagnosis, including fault management of NFV environments [7], [17], [18]. However, as discussed, SL methods require abundant labeled network faulty data that usually is unavailable in sufficient volumes. Some existing works addressed the issue of lack of labeled data by investigating unsupervised ML techniques. The most common UL approach in these works consists in training an Autoencoder on a dataset consisting of only normal samples and performing anomaly detection based on the overall reconstruction error of the Autoencoder. This type of Autoencoder-based UL method has been used in [6] and [19] for anomaly detection in an NFV architecture, in [9] for anomaly detection in Radio Access Network (RAN) cell trace data collected from multiple Evolved NodeBs, and in [20] for detecting anomalous symptoms in 5G RAN. However, all these works assume the availability of a training dataset that consists only of normal samples with no contamination.

B. Unsupervised Anomaly Detection with Contamination

Some existing studies from the ML field (e.g., [10], [11]) addressed the issue of training-data contamination for unsupervised anomaly detection on multi-variant data. Authors in [10] and [11] try to increase robustness against contamination by performing density estimation on the features extracted by the Autoencoder prior to anomaly detection. But, their robustness against contamination is partial, while our approach even leverages the contamination to improve anomaly-detection performance ([11] is one of our compared approaches).

C. Anomaly Localization

Different types of anomaly-localization techniques were proposed for fault diagnosis in network management. Model-based localization methods leverage the knowledge of network topology to build abstraction models, such as dependency graphs, to represent correlations among different metrics and events of the network that can be used for fault localization [21], [22], [23], [24]. However, these techniques are not suitable for NFV architectures that have dynamic topologies [7]. Some data-driven fault localization approaches label different anomalous instances based on the type and location of the fault and turn the problem into a multi-classification one [7], [25], but these methods require abundant labeled anomalous samples to reach good performances. When anomaly detection is performed by an Autoencoder, most existing unsupervised localization approaches try to localize the anomalous VNF by analyzing reconstruction errors of different features in the Autoencoder [26], [27]. However, it is shown that these approaches usually lead to poor localization performances [26]. More recently, AI-explainability methods have been utilized for anomaly localization [9], [28]; however, the methods in

these works only provide some basic information to facilitate the procedure for the domain experts, whereas in our paper, we use AI-explainability algorithms to perform the anomaly-localization task in a fully automated manner.

III. Anomaly-Detection and Localization Approach

A. Problem Formulation

We are monitoring the health state of an NFV system that consists of a network of k VNFs: $vnf^1, vnf^2, ..., vnf^k$. In each time step t, we collect n_j metrics from vnf^j and denote these metrics as $vnf_t^j \in R^{1\times n_j}$. Examples of these metrics could be the general performance metrics shared by all the VNFs, such as CPU utilization and incoming/outgoing packet rates, or more exclusive performance metrics related to the functionality of each individual VNF that can vary from one VNF to another (e.g., call success ratio). Therefore, x_t, our data sample for time step t that represents the status of the entire NFV system consists of the combination of all the collected metrics from all the VNFs: $x_t = \{vnf_t^1, vnf_t^2, ..., vnf_t^k\} \in R^{1\times d}$, where $d = \sum_{j=1}^{k} n_j$ is the total number of metrics collected from the NFV system. We have the historical data of the first n time steps: $X = \{x_1, x_2, ..., x_n\}$, $X \in R^{n\times d}$. We assume the system works in normal circumstances for most of this period; however, the historical data also includes some anomalous samples and is not entirely made of normal instances. Similar to [11], we assume up to 5 percent of the training data might consist of anomalous samples (i.e., up to 5% contamination in the training data). Given this historical data of the NFV system for the first n time steps as the training data, we have two objectives in this problem:

- Anomaly Detection: Detecting anomalous behavior of the system after time step n, i.e., determining the behavior of the system $y_{n+i} \in \{0, 1\}$ (0: normal, 1: anomaly), for samples $x_{n+i}^{test} \in R^{1\times d}$, $i > 0$.
- Anomaly Localization: Localizing the VNF responsible for the anomalous behavior of the system after the detection of an anomaly, i.e., determining r such that vnf^r is the location of the anomaly.

B. Noisy-Student-based Unsupervised Anomaly-Detection (NSUAD) Approach

To overcome the issue of contamination in the training data, we propose an unsupervised anomaly-detection approach based on the Noisy-Student method and call it NSUAD. The Noisy-Student method was introduced in [12] for leveraging unlabeled datasets in computer vision classification problems. In this method, a neural network is trained on the available labeled dataset to reach an initial good performance. Then, this neural network is used as the teacher model to generate pseudo-labels on the unlabeled dataset. Afterward, a larger neural network is trained on the combination of labeled and pseudo-labeled data, and it is called the student model. The key idea in this method that makes the student model have a better performance than the teacher model is that by injecting noise to input data through data augmentation during the learning of the student model, the student achieves better generalization.

1) Teacher Model

In a Noisy-Student approach, we need a teacher model that achieves an initial good performance in the anomaly-detection task, and its predictions are utilized for training a student model that outperforms the teacher. In our problem, we do not have a labeled dataset to train the teacher with; instead, for the teacher model to achieve an initial good performance in anomaly detection, we train a Deep Autoencoding Gaussian Mixture Model (DAGMM) [11], an unsupervised anomaly-detection method, on a fraction (e.g., 40%) of the training data, initially treating the training data as it has no contamination. We train DAGMM on a fraction (e.g., 40% in our experiments) of the training data instead of the whole training data to avoid overfitting on the anomalous samples. In other words, DAGMM treats its training samples as normal instances, therefore, training it on a fraction of data (instead of all of it) would lead to fewer false negatives when we are observing the teacher's prediction on the training data. Since we are interested in extracting pseudo-labeled anomalies from the training samples, having fewer false negatives is more desirable to us than having fewer false positives in this stage.

As we mentioned earlier, DAGMM is shown to be robust against contamination in the training data to some extent [11]; therefore, it is a good choice for our teacher model. The output of DAGMM for a sample is an energy value representing the sample's potential to be an anomaly. The higher the energy of a sample is, the higher is the probability that the sample is an anomaly. After training DAGMM on a fraction of training data, we observe its prediction (energy) of the whole training samples and denote it as E_X:

$$E_X = DAGMM(X),\ E_X \in R^{n\times 1}$$

2) Cleaning Training Data and Extracting Pseudo-labeled Anomalies

We consider $\rho_1\%$ of training samples with the lowest energy as a cleaned training dataset (X_c) that is expected to be much less contaminated than the original training data (thr_1 is defined as the ρ_1^{th} percentile of the energy values in E_X):

$$thr_1 = percentile^{\rho_1}(E_X)$$

$$X_c = X[E_X < thr_1],\ X_c \in R^{s\times d}$$

In our experiments, we chose $\rho_1 = 93\%$ to disregard a slightly more percentage than the highest considered contamination percentage (5%) in the training data. However, X_c still is contaminated (to a lesser degree, though), and we potentially would lose some valuable information by disregarding $(100 - \rho_1)\%$ of the training data. To compensate for this lost information, we try to extract some pseudo-labeled anomalies from the training samples that have the highest energy in the DAGMM method. If μ_E and σ_E are the average and standard deviation of the energy values of samples in X_c, then we define thr_2 as:

$$thr_2 = \mu_E + b \times \sigma_E, b \geq 1$$

In our experimental analysis, we chose $b = 2$, but its value does not have a big impact on the approach performance as long as it is not too large. By this definition, most of the samples in X_c would have a lower energy value than thr_2, and a considerable number of anomalous samples in the training data would have a higher energy value than thr_2; therefore, we also separate all the training samples with higher energy value than thr_2 as the potential contamination data, and denote it as X_{cont}:

$$X_{cont} = X[E_X > thr_2]$$

Now, we choose the top $\rho_2\%$ samples with the highest energy values (e.g., top 20%) in X_{cont} as pseudo-labeled anomalies, and denote as X_{anom} (training samples that DAGMM has classified as anomalies with very high confidence):

$$E_{cont} = DAGMM(X_{cont})$$
$$thr_3 = percentile^{(100-\rho_2)}(E_{cont})$$
$$X_{anom} = X_{cont}[E_{cont} > thr_3]$$

In a Noisy-Student method, the key idea for achieving a better performing student model is to add noise to the pseudo-labeled data through data augmentation to improve the generalization of the student model. Therefore, we utilize the data augmentation method for non-image data proposed in [15], called "data augmentation in feature space (DAiFS)," to add noise to the extracted pseudo-labeled anomalies, and denote the obtained noisy anomalous pseudo-label samples as $\tilde{X}_{anom}$:

$$\tilde{X}_{anom} = DAiFS(X_{anom})$$

Then, our new training data that the student model of NSUAD will be trained on is the combination of X_c and $\tilde{X}_{anom}$:

$$X_{new} = \{X_c, \tilde{X}_{anom}\},\ X_{new} \in R^{m \times d}$$

3) Student Model

For the student model, we propose a novel architecture, consisting of an Autoencoder and a Deviation Network (DevNet), to be trained on X_{new}. The proposed student model is presented in Fig. 1. DevNet [14] is a semi-supervised learning anomaly-detection approach that is designed to utilize a few available labeled anomalies to improve the anomaly-detection performance; therefore, it is a well-suited approach to take advantage of the extracted pseudo-labeled anomalies in our problem. In DevNet, a single anomaly score value is directly learned from the input sample, and this anomaly score would be the anomaly-detection criterion. However, since the anomaly score is a direct mapping from the input space to a single scalar value, DevNet might not perform well when dealing with high-dimensional data. To address this issue, for the student model, we integrate DevNet with an Autoencoder in a novel architecture, where Devnet learns the anomaly score from low-dimensional features extracted from the input data by the Autoencoder, instead of learning it directly from the input samples. These low-dimensional features contain valuable information regarding the anomalous behavior of the input samples, thus, can improve the performance of DevNet and the scalability of the student model for large NFV systems.

Autoencoder has two parts, an encoder and a decoder. The encoder is a Feed-Forward Neural Network (FNN) that maps the input sample x to a low-dimensional representation, and the decoder is another FNN that tries to reconstruct the input sample from this low-dimensional representation. Reconstruction error ($rec_{err}(x)$) of the Autoencoder would be defined as the difference between the reconstructed input sample and the actual input sample:

$$encoder = FNN(input\_shape = d, output\_shape = d_1)$$
$$d_1 << d$$
$$decoder = FNN(input\_shape = d_1, output\_shape = d)$$
$$Autoencoder : x_{rec} = decoder(encoder(x))$$
$$rec_{err}(x) =< ||x_{rec} - x||, \frac{x_{rec} \cdot x}{||x_{rec}|| \times ||x||} >$$

Accordingly, we can define the overall latent features (denoted as $z(x)$) extracted from the input by the Autoencoder as the combination of the decoder's output and the reconstruction error features:

$$z(x) = \{encoder(x), rec_{err}(x)\},\ z(x) \in R^{1 \times (d_1+2)}$$

Now, in NSUAD, these latent features extracted by the Autoencoder are fed to the DevNet model for anomaly detection. DevNet is a simple neural network that learns a scalar anomaly score $\Phi(z)$ value from its input z. In the training of DevNet, anomaly scores of normal samples (samples in X_c) are forced to be close to a reference average, and anomaly scores of anomalous samples ($\tilde{X}_{anom}$) are forced to deviate significantly from this average. To achieve this goal while simultaneously minimizing the reconstruction error of the Autoencoder, we define the following loss function for end-to-end training of the whole proposed student model (Autoencoder + DevNet) on X_{new}:

$$L(x) = \begin{cases} \alpha||x_{rec} - x|| + |\frac{\Phi(z(x)) - \mu_R}{\sigma_R}|, x \in X_c \\ \\ max(0, a - \frac{\Phi(z(x)) - \mu_R}{\sigma_R}), x \in \tilde{X}_{anom} \end{cases}$$

Where $\alpha > 0$ determines the effect of the two components of loss function for normal samples (we chose $\alpha = 5$ in our experiments). Same as in [14], we assume that anomaly scores of normal samples have a standard normal distribution $F : N(0, 1)$, and for training a batch of training samples, we draw l observations from this distribution: $(r_1, r_2, ..., r_l)$. Then, the reference average is calculated based on these l drawn observations: $\mu_R = \frac{1}{l}\sum_{i=1}^{l} r_i$ and $\sigma_R = \frac{1}{l}\sum_{i=1}^{l}(r_i - \mu_R)^2$. Similar to the loss function in [14], our loss function pushes anomaly scores of normal samples in X_c to be near μ_R and requires anomaly scores of samples in $\tilde{X}_{anom}$ to deviate from μ_R with at least $a > 0$ confidence level. We chose $a = 5$ to ensure significant deviation from μ_R for our anomalous samples. Moreover, in NSUAD, we are also considering the

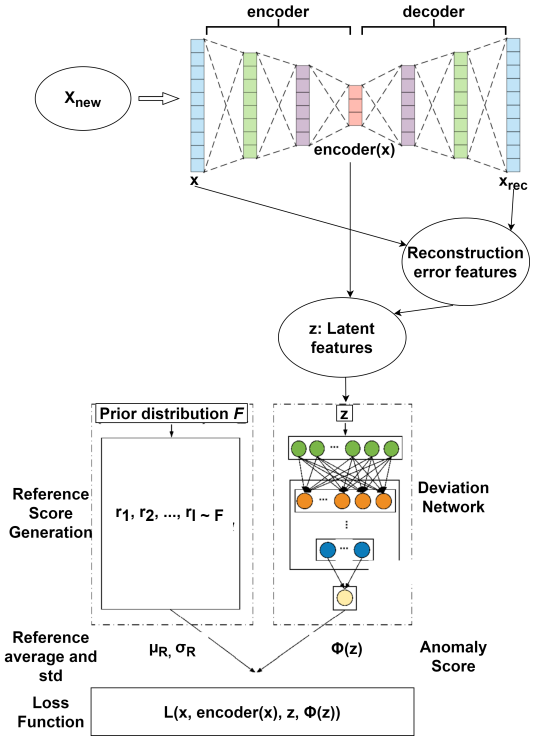

Fig. 1. The proposed student model to be trained on X_{new}. The latent features (low-dimensional representation and reconstruction error features) are extracted from the input by the Autoencoder and are fed to the Deviation Network. Then, the anomaly score output of the Deviation Network alongside average anomaly score of some normal samples (μ_R) that is calculated according to a prior distribution F are fed to the loss function for end-to-end training of the whole model.

minimization of reconstruction error of the Autoenncoder for our normal samples in the loss function. Moreover, same as in [14], for creating a batch of training samples with size β, we select $\beta/2$ samples from X_c and $\beta/2$ samples from $\tilde{X}_{anom}$. In the end, based on the loss function, parameters of the Autoencoder and the DevNet are optimized by performing a gradient descent step. In the evaluation phase, a test sample is considered as an anomaly if its anomaly score exceeds a predefined threshold (e.g., $a/2$).

C. Anomaly Localization

After an anomaly is detected, localizing the VNF that is responsible for the anomalous behavior of the system is the next important step in fault management of NFV systems. In this paper, we have utilized the local AI-explainability method SHAP [16] to perform the localization task in a fully automated manner and achieve high localization accuracy. Afterward, we propose our own novel AI-explainability method, called "Cluster Permutation", customized specifically for our problem to further improve the localization performance. In the end, we show how cascading Cluster Permutation with SHAP can help us to reach an even higher performance.

1) Localization with SHAP

The complete procedure of anomaly localization with the SHAP method [16] is presented in Algorithm 1. To determine the contribution of each feature for the anomalous behavior of a detected anomaly sample, SHAP gets the detected anomaly x_i^{test}, the detection model NSUAD, and a set of background input samples to explore the input space with. Giving the entire training samples as the background data to the SHAP method would result in a high impractical execution time. A possible solution is to apply k-means on the training data and give the obtained centroids, as representatives of the training data, to the SHAP method. In our experiments, we applied k-means on X with 50 clusters and gave the corresponding centroids $Cents$ to the SHAP method as background data to achieve a reasonable execution time. $shap\_values_i^{test} \in R^{1\times d}$, the output of the SHAP method for the detected anomaly x_i^{test}, has a value for each feature that represents the effect of the feature on the detection model's output for this detected anomaly. In the next step, for each vnf^j, we separate elements of $shap\_values_i^{test}$ corresponding to the features of vnf^j and denote it as $shap\_values_i^{test}(j)$. Finally, the anomaly location is chosen to be the VNF whose SHAP values have the highest first norm. We have also considered a scaling factor $c_j \in C$ for each $shap\_values_i^{test}(j)$ to avoid any potential biases towards/against some specific VNFs. For example, in our experiments, we chose $c_j = \frac{1}{\sqrt{n_j}}$ to avoid being biased towards the VNFs from which we collect a higher number of features (n_j was the number of features collected from vnf_j). Moreover, it is important to note that we chose the first norm instead of the second norm because it allows the algorithm to consider a higher number of features in localizing the anomaly, where the second norm might focus the algorithm on a few specific features (i.e., features with large SHAP values become too dominant in determining the anomaly location) and degrade the localization performance.

Algorithm 1 Anomaly Localization with SHAP

1: **function** SHAPLOCAL($x_i^{test}, NSUAD, Cents, C$)
2: $shap\_values_i^{test} \leftarrow SHAP(x_i^{test}, USUAD, Cents)$
3: $shap\_values_i^{test}(j) \leftarrow$ elements of $shap\_values_i^{test}$ related to vnf^j
4: $location \leftarrow argmax_j(c_j||shap\_values_i^{test}(j)||_1)$
5: **return** $location$

2) Localization with Cluster Permutation

SHAP is a general model-agnostic AI-explainability method that can explain the output of all types of ML methods and is not specific to our problem. To achieve a higher localization performance, we propose our own novel local AI-explainability method, "Cluster Permutation," customized for the specific task of anomaly localization in NFV systems. The overall procedure of Cluster Permutation is presented in Algorithm 2. In this algorithm, to check whether vnf^j is the location of the detected anomaly $x_i^{test} = \{vnf_i^1, vnf_i^2, ..., vnf_i^k\}$, we first define the metrics related to vnf^j in x_i^{test} as $\overline{x_i^{test}(j)}$, and the remaining metrics in x_i^{test} as $\widehat{x_i^{test}(j)}$. We do the same separation on all the samples in X_c to obtain $\overline{X_c(j)}$ and $\widehat{X_c(j)}$. Then, by the K-Nearest Neighbors (KNN) algorithm, we find the indices of the nearest neighbors of $\widehat{x_i^{test}(j)}$ in the

$\widehat{X_c(j)}$ dataset. These neighbors would be normal samples that resemble our detected anomaly sample when we are excluding metrics of vnf^j. Our intuition is that if vnf^j is the location of the anomaly, by replacing the current metrics of vnf^j in the detected anomaly with metrics related to vnf^j in normal neighbors, the new created sample would have a lower degree of anomalous behavior, and its anomaly score would be lower and closer to μ_R. For each vnf^j, we measure the average change in the anomaly score for all the neighbor samples, and the vnf^j that has the highest decrease in anomaly score when its metrics are replaced by normal neighbors' metrics is chosen as the location of the anomaly. However, if there is no decrease in the anomaly score for any of the VNFs or the decrease in the anomaly score is less than a threshold (Thr) that is given to the algorithm, the output of the algorithm is "undecided". One important point is that even though X_c is much less contaminated than the original training data, it still has some degree of contamination, but in this algorithm, we are treating X_c as it only includes normal samples. However, this issue would not be problematic if we choose K in the KNN algorithm large enough so that the obtained neighbors include enough normal samples for the algorithm to work properly. Another important point is that the first for loop in Algorithm 2 can be executed in parallel, therefore, in terms of execution time, it will be scalable for large NFV systems as well.

3) Localization with Cluster Permutation + SHAP

As mentioned earlier, Cluster Permutation's output might be "undecided" if there is no decrease in the anomaly score for any of the VNFs in the procedure. In our experiments, we observed that Cluster Permutation is very accurate when it has a valid output, but its output becomes "undecided" for some detected anomalies. To overcome this issue, in a combined approach, we first run the Cluster Permutation algorithm for a detected anomaly, and then if its output was "undecided," we run the SHAP method to find the location of the anomaly; otherwise, we choose Cluster Permutation's output as the location of the anomaly. Our experimental results showed that this combined approach outperforms SHAP since Cluster Permutation is very accurate when it has a valid output.

Algorithm 2 Anomaly Localization with Cluster Permutation

function CLUSTERPER($x_i^{test}, X_c, NSUAD, Thr$)
  $\Delta\Phi \leftarrow \{\}$
  for $j = 1$ to k **do**
    $\overline{x_i^{test}(j)} = \{vnf_i^j\} \in R^{1\times n_j}$
    $\widehat{x_i^{test}(j)} = (x_i^{test} \setminus \overline{x_i^{test}(j)}) \in R^{1\times(d-n_j)}$
    $\overline{X_c(j)} = \{\overline{x_{c1}(j)}, \overline{x_{c2}(j)}, ..., \overline{x_{cs}(j)}\}$
    $\widehat{X_c(j)} = \{\widehat{x_{c1}(j)}, \widehat{x_{c2}(j)}, ..., \widehat{x_{cs}(j)}\}$
    Run KNN in $\widehat{X_c(j)}$ for $\widehat{x_i^{test}(j)}$
    Find neighbors: $\widehat{X_{cp}(j)}, p \in \mathbb{NEIGHB}$
    $\Delta\phi \leftarrow 0$, $c \leftarrow 0$
    for $p \in \mathbb{NEIGHB}$ **do**
      $x\_mod_i^{test} \leftarrow \{vnf_i^1, ..., vnf_i^{j-1}, \overline{x_{cp}(j)}, ..., vnf_i^k\}$
      $z_i^{test} \leftarrow z(x_i^{test})$
      $z\_mod_i^{test} \leftarrow z(x\_mod_i^{test})$
      $\Delta \leftarrow |\frac{\Phi(z_i^{test})-\mu_R}{\sigma_r}| - |\frac{\Phi(z\_mod_i^{test})-\mu_R}{\sigma_R}|$
      if $\Delta > 0$ **then**
        $\Delta\phi \leftarrow \Delta\phi + \Delta$
        $c \leftarrow c + 1$
    if $c > 0$ **then**
      $\Delta\Phi.append(\frac{\Delta\Phi}{c})$
    else
      $\Delta\Phi.append(0)$
  $location \leftarrow argmax(\Delta\Phi)$
  $\Delta\Phi\_max \leftarrow max(\Delta\Phi)$
  if $(\Delta\Phi\_max > Thr)$ **then**
    $output \leftarrow location$
  else
    $output \leftarrow$ "$undecided$"
  return $output$

IV. Experimental Results

A. Datasets

1) Public Dataset

Our first dataset is from the "ITU AI/ML in 5G" challenge [29]. It was generated in an NFV-based test environment that simulates a 5G IP core network. The target topology of the NFV testbed is shown in Fig. 2a and consists of 5 VNFs: two IP core nodes (TR-01 and TR-02), two internet gateway routers (IntGW-01 and IntGW-01), and a router reflector (RR-01), each hosted on a different Virtual Machine (VM). Various performance metrics, such as CPU utilization and network incoming/outgoing packet rates, are collected from each VNF per minute. For evaluation of the anomaly-detection mechanisms, the following fault scenarios are injected to one of the VNFs periodically: 1) node failure, i.e., an unplanned reboot of a VNF. 2) interface failure, i.e., a failure that causes an interface to be down, and 3) packet loss/delay, i.e., an event that causes packet loss/delay on an interface. We label each faulty instance according to the location of the fault (1:5, the VNF to which the fault is injected), and that would be the target that our localization algorithms should predict. A well-structured version of this dataset can be found in [30]. For this dataset, the training data includes 3870 normal samples, and the test data includes 3505 normal samples and 1122 anomalies. For creating a certain level of contamination in the training data, we choose some anomalous samples at random from a set that includes an equal number of different types of anomalies and add those anomalies to the training data.

2) Synthetic Dataset

Our second dataset is from our experimental NFV testbed depicted in Fig. 2b, and consists of 4 open-source VNFs: a Firewall (iptables [31]), an intrusion detection System (Suricata [32]), a deep packet inspector (nDPI [33]), and a flow monitor (ntopng [34]). We have adopted this topology from [17] and implemented it on the SAVI testbed [35], where each of the VNFs is hosted on a different (VM). We used Apache Bench for traffic generation in the testbed. We have collected 61 resource-related metrics (CPU, Disk, memory, and network) from all the VNFs every 5 seconds (see [17] for a complete list of collected metrics). We injected one of

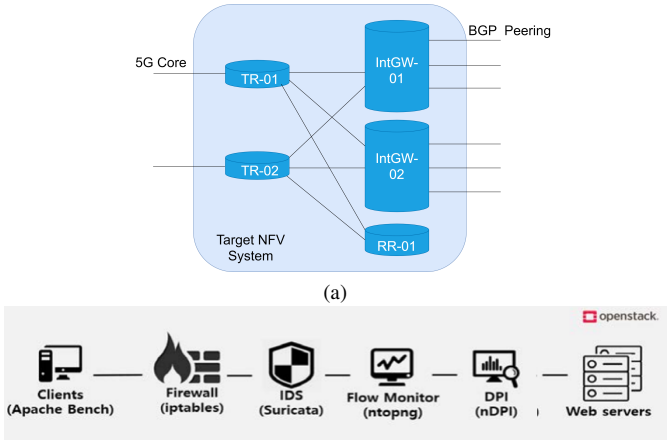

Fig. 2. NFV system of (a) Public Dataset and (b) Synthetic Dataset.

the flowing faults to one of the VNFs periodically to generate faulty instances to evaluate our anomaly-detection techniques: 1) *CPU stress* by the stress-ng [36] tool that increases the CPU usage in the VM, 2) *disk stress* by the stress-ng [36] tool that increases the disk usage in the VM, and 3) *network stress* by the tc [37] tool where network delay of one of the interfaces in a VM increases. For this dataset, the training data includes 3500 normal samples, and the test data includes 1500 normal samples and 1500 anomalous samples.

3) Data Preprocessing

We perform the necessary data prepossessing tasks before feeding the data to the ML models, including replacing the accumulative values (e.g., number of packets sent) with their numeric difference, data normalization due to the different dynamic ranges of the collected metrics, metric selection, etc. The compared approaches were evaluated on a server with 2x20 core Intel Xeon Silver 4114 2.20GHz CPU, 187 GB memory, and NVIDIA Tesla P40 GPU.

B. Compared Approaches

1) Baseline

The baseline approach used to compare our detection approach with is the Autoencoder-based anomaly-detection algorithm in [6] and [9], where the Autoencoder is trained on only normal samples and then based on the overall reconstruction error of the Autoencoder, anomalies are separated from normal samples in the test data. The baseline approach for localization is a conventional approach where the VNF whose features have been reconstructed more poorly is chosen as the location of the fault. Since [6] and [9] have worked with different datasets that are not publicly available, we have designed the best Autoencoder architecture for our datasets by trying different architectures, from very shallow to very deep, to reach the best possible outcome for the Baseline. The final chosen Autoencoder architecture (number of nodes in each layer) for the Public Dataset is $\{116, 90, 60, 30, 10, 30, 60, 90, 116\}$, and for the Synthetic Dataset is $\{61, 40, 20, 8, 20, 40, 61\}$. Moreover, similar to [9], we added L2-regularization to the Autoencoder network to improve the model's robustness against contamination in the training data.

2) DAGMM

Our teacher model, DAGMM [11], trained on the whole training data, is another compared approach. DAGMM has shown to be robust against contamination in the training data to some extent since it performs density estimation on features extracted from its Autoencoder. Therefore, it is a good choice to be compared with our approach when dealing with contaminated training data. The localization task here is the same as the Baseline.

3) MSCRED

Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) was proposed in [26] for unsupervised anomaly detection and localization (diagnosis) in multivariate time series data. In this approach, inter-correlation between different metrics is calculated with different temporal window sizes, and a Recurrent Encoder-Decoder DL model is trained to construct these inter-correlations for normal instances. Then, anomaly-detection and localization tasks are performed based on the reconstruction errors of these inter-correlations values.

4) Our Proposed Methods

NSUAD is our proposed anomaly-detection solution described in Section III.B. SHAP (Algorithm 1), ClusterPer (Algorithm 2), and ClusterPer+SHAP (Section III.C.3) are our proposed anomaly-localization approaches that utilize NSUAD as their detection model.

C. Detection Results

The performance metrics for anomaly detection (namely, Precision, Recall, and F1-score) of different approaches are shown in Table I. In the first row of Table I, we report the detection performance metrics when there is no contamination in the training data ($\delta = 0\%$), so we can clearly observe the effect of contamination on these approaches in the next experiments. We can see that when there is no contamination, DAGMM has higher Precision, Recall, and F1-score than Baseline and MSCRED, especially for the Public Dataset, so it is a good choice for our teacher model. In the next rows of Table I, we change the contamination percentage (δ) in the training data from 1% to 5% and report the detection performance. We can see that contamination in the training data significantly degrades the performance of Baseline, MSCRED, and DAGMM, and this degradation becomes more severe as δ increases. Also, in both datasets, DAGMM has a better performance than Baseline and MSCRED (especially in the Public Dataset), and its performance degrades less significantly compared to the Baseline and MSCRED as δ increases; therefore, DAGMM is confirmed to be an excellent choice as the teacher model in our approach.

Moreover, our detection approach NSUAD has a better performance than DAGMM, MSCRED, and Baseline when δ is 1%, and, unlike the other approaches, the detection performance of NSUAD improves when δ increases up to 5% since it takes advantage of the extracted pseudo-labeled anomalies in the contaminated training data. More specifically, increasing δ (percentage of anomalous samples) in the training data has one negative effect and one positive effect on NSUAD's

TABLE I
Anomaly-detection results of different methods on contaminated training data with the contamination percentage (δ) varying from 0% to 5%

δ	Public Dataset												Synthetic Dataset											
	Baseline			DAGMM			MSCRED			NSUAD (ours)			Baseline			DAGMM			MSCRED			NSUAD (ours)		
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1
0	80.8	46.6	58.9	**83.3**	**82.1**	**82.7**	76.4	62.2	68.4	-	-	-	83.4	78.5	80.9	**85.8**	**83.7**	**84.7**	76.2	73.3	74.7	-	-	-
1	77.6	31.4	44.8	80.1	80.2	80.1	71.5	55.9	62.3	**82.7**	**81.6**	**82.1**	80.3	71.2	75.5	82.1	78.8	80.4	65.6	62.7	64.0	**86.1**	**83.5**	**84.8**
2	76.9	31.1	44.5	79.3	78.9	79.1	68.6	49.3	57.3	**84.1**	**83.5**	**83.8**	78.9	64.1	70.7	77.3	75.2	76.2	49.3	48.8	49.1	**90.4**	**87.1**	**88.7**
3	76.1	30.0	43.0	75.8	74.8	75.3	57.7	42.1	48.6	**85.4**	**83.8**	**84.6**	77.2	60.9	68.1	74.5	73.6	74.1	41.2	38.7	39.9	**92.7**	**90.5**	**91.6**
4	75.8	29.5	42.6	73.5	72.6	72.9	41.5	32.6	36.5	**86.1**	**84.4**	**85.2**	70.8	59.5	64.6	73.6	71.8	72.7	33.9	30.5	32.1	**93.2**	**91.1**	**92.1**
5	74.5	29.2	42.0	71.8	71.7	71.7	27.8	24.3	25.9	**86.5**	**84.4**	**85.4**	66.7	52.6	58.8	70.3	69.9	70.1	20.3	19.7	20.0	**93.1**	**91.4**	**92.2**

performance. The negative effect is that as δ increases, the performance of NSUAD's teacher model (DAGMM) degrades. The positive effect is that NSUAD extracts more pseudo-labeled anomalies when δ is higher. In our experiments, when we increase δ up to 5%, the positive effect seems to be more important than the negative effect; therefore, the overall performance of NSUAD becomes better for higher contamination percentages in the training data.

However, we expect that after a certain level of contamination, the degradation in the teacher model's performance (the negative effect) becomes significant enough to decrease NSUAD's overall performance. For example, in the Public Dataset, when we increased δ to 7%, the teacher model's F1-score declined to 67.5%, and NSUAD's F1-score degraded to 81.6%, which is lower than the case where δ was 5%. For very high contamination levels in the training data, Active Learning [38] approaches should be utilized to label the data with the help of domain experts.

D. Localization Results

The performance metrics for anomaly localization (namely, Precision, Recall, F1-score, number of truly detected anomalies in the detection phase, and average execution time) of different approaches for the detected anomalies in both datasets are shown in Table II. As discussed, the localization methods use a detection model for their process and localize the anomalies that are detected by that detection model. NSUAD is the detection model for SHAP, ClusterPer, and ClusterPer + SHAP localization methods (with a training data that is 4% contaminated), and the detection models for the other localization methods are the same as their names according to Section IV.B. Therefore, the number of truly detected anomalies is different for different approaches. According to the results of Table II, our SHAP method outperforms Baseline, MSCRED, and DAGMM in both datasets despite having more detected anomalies. In the Public Dataset, SHAP has a better performance than ClusterPer because ClusterPer's output was "undecided" for many of the detected anomalies, but in the Synthetic Dataset, ClusterPer has a better performance than SHAP because its output was "undecided" in only a few cases. Either way, in both datasets, cascading ClusterPer and SHAP (ClusterPer + SHAP) results in a better F1-score than only using SHAP and achieves the highest performance among the compared approaches. Moreover, Baseline has the lowest average execution time in both datasets since its procedure is a simple analysis of the reconstruction error of different features, while SHAP has the highest execution time due to its relatively complex calculations. However, our proposed ClusterPer localization approach localizes anomalies significantly faster than SHAP, therefore, ClusterPer + SHAP (the method with the highest F1-score) leads to a reasonable execution time that is considerably lower than only using SHAP.

TABLE II
Anomaly localization results of the compared approaches

Public Dataset					
Method\Performance	Prec.	Recall	F1	#Anom.	Time(ms)
Baseline	62.4	65.1	62.7	516	**12**
DAGMM	58.3	58.2	58.2	913	13
MSCRED	73.6	73.4	73.4	698	17
ClusterPer (ours)	68.6	68.8	68.5	938	21
SHAP (ours)	87.7	87.6	87.4	938	108
ClusterPer+SHAP (ours)	**92.4**	**92.5**	**92.3**	938	37
Synthetic Dataset					
Method\Performance	Prec.	Recall	F1	#Anom.	Time(ms)
Baseline	73.3	73.1	73.2	157	**9**
DAGMM	64.2	64.4	64.2	167	**9**
MSCRED	78.5	78.6	78.5	110	12
ClusterPer (ours)	91.5	91.4	91.5	182	19
SHAP (ours)	84.5	84.4	84.4	182	97
ClusterPer+SHAP (ours)	**93.2**	**93.1**	**93.1**	182	22

V. Conclusion and Future Work

We proposed an unsupervised method for anomaly detection in NFV systems that is robust against training-data contamination up to a certain percentage and can also leverage the contamination to improve anomaly-detection performance, unlike state-of-the-art unsupervised approaches whose performances degrade when the training data is contaminated. Moreover, we described how utilizing SHAP and our proposed AI-explainability method, called "Cluster Permutation", can achieve high performance in the anomaly-localization task. By a complete experimental analysis on two datasets from two different NFV systems, we showed that in terms of F1-score, our proposed solutions outperform other state-of-the-art unsupervised methods by up to 22% and 19% in anomaly-detection and localization tasks, respectively.

For future work, we can focus on improving the generalization of our detection/localization models, evaluating their applicability on larger and more complex datasets, and also including identification of the type of faults in an unsupervised manner in the NFV system.

Acknowledgement: This work was supported in part by Rogers Communications Canada Inc. and in part by a Mitacs Accelerate Grant.

REFERENCES

[1] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[2] B. Yi, X. Wang, K. Li, M. Huang *et al.*, "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, pp. 212–262, 2018.

[3] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[4] B. Han, V. Gopalakrishnan, G. Kathirvel, and A. Shaikh, "On the resiliency of virtual network functions," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 152–157, 2017.

[5] J. Nam, J. Seo, and S. Shin, "Probius: Automated approach for VNF and service chain analysis in software-defined NFV," in *Proceedings of the Symposium on SDN Research*, 2018, pp. 1–13.

[6] F. Schmidt, A. Gulenko, M. Wallschläger, A. Acker, V. Hennig, F. Liu, and O. Kao, "Iftm-unsupervised anomaly detection for virtualized network function services," in *2018 IEEE International Conference on Web Services (ICWS)*. IEEE, 2018, pp. 187–194.

[7] A. Elmajed, A. Aghasaryan, and E. Fabre, "Machine learning approaches to early fault detection and identification in NFV architectures," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2020, pp. 200–208.

[8] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3395–3404.

[9] A. Chawla, P. Jacob, S. Feghhi, D. Rughwani, S. van der Meer, and S. Fallon, "Interpretable unsupervised anomaly detection for RAN cell trace analysis," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–5.

[10] J. Fan, Q. Zhang, J. Zhu, M. Zhang, Z. Yang, and H. Cao, "Robust deep auto-encoding gaussian process regression for unsupervised anomaly detection," *Neurocomputing*, vol. 376, pp. 180–190, 2020.

[11] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International conference on learning representations*, 2018.

[12] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 687–10 698.

[13] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, no. 2, 2013, p. 896.

[14] G. Pang, C. Shen, and A. van den Hengel, "Deep anomaly detection with deviation networks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 353–362.

[15] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," *arXiv preprint arXiv:1702.05538*, 2017.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768–4777.

[17] J. Hong, S. Park, J.-H. Yoo, and J. W.-K. Hong, "Machine learning based SLA-aware VNF anomaly detection for virtual network management," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–7.

[18] C. Sauvanaud, K. Lazri, M. Kaâniche, and K. Kanoun, "Anomaly detection and root cause localization in virtual network functions," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2016, pp. 196–206.

[19] A. Diamanti, J. M. S. Vilchez, and S. Secci, "LSTM-based radiography for anomaly detection in softwarized infrastructures," in *2020 32nd International Teletraffic Congress (ITC 32)*. IEEE, 2020, pp. 28–36.

[20] L. F. Maimó, Á. L. P. Gómez, F. J. G. Clemente, M. G. Pérez, and G. M. Pérez, "A self-adaptive deep learning-based system for anomaly detection in 5G networks," *IEEE Access*, vol. 6, pp. 7700–7712, 2018.

[21] J. M. Sánchez, I. G. B. Yahia, and N. Crespi, "Self-modeling based diagnosis of services over programmable networks," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. IEEE, 2016, pp. 277–285.

[22] Q. Zhu, T. Tung, and Q. Xie, "Automatic fault diagnosis in cloud infrastructure," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 1. IEEE, 2013, pp. 467–474.

[23] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 13–24, 2007.

[24] A. Samir and C. Pahl, "Detecting and localizing anomalies in container clusters using Markov models," *Electronics*, vol. 9, no. 1, p. 64, 2020.

[25] C. Pham, L. Wang, B. C. Tak, S. Baset, C. Tang, Z. Kalbarczyk, and R. K. Iyer, "Failure diagnosis for distributed systems using targeted fault injection," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 503–516, 2016.

[26] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1409–1416.

[27] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[28] A. Terra, R. Inam, S. Baskaran, P. Batista, I. Burdick, and E. Fersman, "Explainability methods for identifying root-cause of SLA violation prediction in 5g network," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–7.

[29] "itu-ai-ml-in-5g-challenge". [Online]. Available: https://www.ieice.org/~rising/AI-5G/

[30] [Online]. Available: https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-032-KDDI-naist-lsm

[31] "iptables". [Online]. Available: http://ipset.netfilter.org/iptables.man.html

[32] "suricata - open source ids/ips/nsm engine". [Online]. Available: https://suricata-ids.org/

[33] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "ndpi: Open-source high-speed deep packet inspection," in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2014, pp. 617–622.

[34] "ntopng - high-speed web-based traffic analysis and flow collection". [Online]. Available: https://www.ntop.org/products/traffic-analysis/ntop/

[35] J.-M. Kang, H. Bannazadeh, and A. Leon-Garcia, "Savi testbed: Control and management of converged virtual ict resources," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, 2013, pp. 664–667.

[36] [Online]. Available: https://manpages.ubuntu.com/manpages/artful/man1/stress-ng.1.html

[37] [Online]. Available: https://wiki.debian.org/TrafficControl

[38] B. Settles, "Active learning literature survey," 2009.