Anomaly Detection and Localization in NFV Systems by Utilizing Masked-Autoencoder and XAI

Seyed Soheil Johari*, Nashid Shahriar†, Massimo Tornatore‡, Raouf Boutaba*, Aladdin Saleh§
*Department 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*—The integration of Network Functions Virtualization (NFV) systems into mobile edge and core networks has heightened the need for effective anomaly detection and localization methods. The complexity of NFV demands robust mechanisms for network resilience, security, and performance. Machine Learning approaches have demonstrated promising solutions in crafting adaptive and efficient mechanisms for detecting and localizing potential anomalies within NFV systems. Particularly, Unsupervised Learning (UL) methods have garnered significant attention for their potential to detect anomalies without the need for labeled data. However, UL methods are susceptible to even minor levels of anomalous samples in the training data, termed contamination, which can severely compromise their performance. This paper proposes a novel approach using the Noisy-Student technique for anomaly detection. It addresses data contamination by combining a density-estimation teacher model for pseudo-labeling with a weakly-supervised student model based on a Masked Autoencoder trained on the pseudo-labeled data. For anomaly localization, we introduce a heuristic tailored for our anomaly detection model and two Explainable Artificial Intelligence (XAI)-based approaches applicable to any detection model. Extensive experiments on three NFV datasets demonstrate superior performance, with up to a 20% improvement in anomaly detection and up to a 22% improvement in localization, in terms of F1-score.**

***Index Terms*—Anomaly Detection, Anomaly Localization, NFV**

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 promising application of virtualization in networking is 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. Furthermore, with the continuous advancement and widespread adoption of VNFs especially in mobile computing environments, the potential for achieving near-hardware performance and realizing substantial opportunities for network optimization and cost reduction has become increasingly evident. As NFV deployments continue to proliferate in mobile computing, the effective implementation of anomaly detection and localization techniques is crucial to ensure the resilience, security, and efficiency of these dynamic and complex systems, ultimately enabling the realization of their full potential in enhancing mobile network capabilities [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 root cause is of paramount importance to ensure high reliability for virtualized services.

The complex inter-dependencies and multi-faceted fault characteristics in NFV systems render traditional anomaly-detection and localization approaches inefficient as they typically identify malfunctions by metrics crossing a threshold configured by some field expert [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 [8], [9]. However, most of these existing ML-based approaches utilize Supervised Learning (SL) algorithms that require abundant labeled faulty instances to achieve satisfactory performance. Unfortunately, labeled network faulty data is a scarce resource and generally unavailable in sufficient volumes for two main reasons: *i)* labeling data often 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 alleviate the need for abundant labeled faulty instances. Autoencoder is one of the most popular UL method that has been utilized for unsupervised anomaly detection in many network management tasks, including anomaly detection in NFV architectures [6], [10]. 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 system includes a few anomalous samples, i.e., we expect 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 [11], [12].

In this paper, we propose a novel unsupervised anomaly-detection approach for NFV systems when training data is contaminated (a problem arising in most practical scenarios). Inspired by the Noisy-Student [13] concept used in computer vision, we first train a Deep Autoencoding Gaussian Mixture Model (DAGMM) [12], (i.e., a UL anomaly-detection method

based on density estimation) on the contaminated training data as the teacher model. 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 [14] 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 a Masked Autoencoder (MAE) [15] cascaded with a weakly-supervised anomaly-detection model called Deviation Network [15]. We also use data augmentation in feature space [16], 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 done 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 anomaly locations can only be done by comparing the detected anomaly with normal instances. In this paper, we first propose a heuristic method that utilizes the output of our MAE-based detection model for different masking scenarios of the MAE to determine the anomaly location. While the proposed heuristic is specifically designed for the MAE-based detection model, we also propose two general localization methods based on XAI that are applicable to any black-box anomaly detection model. We show the effectiveness of our proposed solutions through comprehensive experimental evaluations on three datasets from different NFV testbeds. The first dataset is generated in an NFV-based test environment that simulates a 5G IP core network. The second dataset is from our experimental NFV testbed that resembles Multi-access Edge Computing (MEC) in topology. The last dataset is collected by [17] from the ClearWater project, which is an NFV-based open source implementation of an IP Multimedia Subsystem (IMS) for cloud platforms. In the evaluation results, we observed improvement up to 24% in anomaly detection and up to 22% in anomaly localization in terms of F1-score compared to the state-of-the-art methods.

This paper is an extended version of the work presented in [18]. In this extended version, we modified the student model of the anomaly detection architecture to learn more generalizable features from the input data through a MAE, which is a transformer-based Autoencoder model. As we will see in the next sections, utilizing MAE in the student model not only improves the anomaly detection performance, but can also provide valuable information for the localization task based on its output for different masking scenarios. In this extended version, we build on this idea by introducing a novel localization method called Mask Permutation, which systematically identifies contributing features by analyzing reconstruction quality under different masking patterns. Moreover, in the extended version, we compare our student model with other weakly-supervised methods from the state-of-the-art in a new set of experiments. We also compare our proposed anomaly detection method with SL methods in terms of generalization capability, and show that unlike SL models, our method is able to detect failure scenarios that are unseen during training. Finally, we expand our discussion of the related work and experimental results.

In summary, existing NFV anomaly detection methods struggle with training data contamination, limiting the effectiveness of unsupervised approaches that assume clean data. Additionally, weakly-supervised methods often rely on simplistic feature extraction, reducing their generalization in dynamic NFV environments. For anomaly localization, prior techniques depend on manual heuristics, lacking automation and interpretability. To address these limitations, we introduce MNSUAD for robust anomaly detection and Mask Permutation, SHAP, and Cluster Permutation for fully automated localization, improving accuracy and interpretability in NFV systems. The main contributions of this paper are as follows:

- **Masked Noisy-Student-Based Unsupervised Anomaly Detection (MNSUAD) Approach:** We propose MNSUAD, a novel unsupervised anomaly detection method for NFV systems that effectively mitigates the impact of training data contamination. The approach utilizes the Noisy-Student paradigm by employing a density-estimation-based teacher model (DAGMM) to clean and pseudo-label the data, followed by training a weakly-supervised student model on the cleaned dataset augmented with pseudo-labeled anomalies. The novel architecture of our student model integrates a Masked Autoencoder (MAE) with Deviation Networks (DevNet). The MAE learns generalizable feature representations from input data, while DevNet maps these representations to anomaly scores, improving detection accuracy and scalability in NFV environments.
- **Fully Automated Anomaly Localization Methods:** We develop three *anomaly localization techniques*: (i) Mask Permutation, which utilizes outputs from the MAE-based detection model for different masking scenarios to determine the anomaly location, and (ii) two general XAI-based methods (SHAP and Cluster Permutation) that enable interpretable and automated root cause analysis applicable to any black-box anomaly detection model.
- **Extensive Evaluation on Realistic NFV Datasets:** We comprehensively evaluate our proposed approaches on three NFV datasets from distinct testbeds, showing up to a 20% improvement in anomaly detection F1-score and up to a 22% improvement in anomaly localization F1-score over state-of-the-art methods.

For convenience, a complete list of acronyms used in this paper is provided in Table I.

II. Related Work

A. Achieving Near-Hardware Performance in NFV

NFV decouples network functions from specialized hardware, enabling their deployment as software on commodity servers, and has shown potential to approach near-hardware

TABLE I
List of acronyms used in the paper

Acronym	Definition
AI	Artificial Intelligence
CL	Contrastive Learning
DAGMM	Deep Autoencoding Gaussian Mixture Model
DAiFS	Data Augmentation in Feature Space
DevNet	Deviation Network
DL	Deep Learning
IMS	IP Multimedia Subsystem
KNN	K-Nearest Neighbors
LOE	Latent Outlier Exposure
MAE	Masked Autoencoder
MEC	Multi-access Edge Computing
ML	Machine Learning
MSCRED	Multi-Scale Convolutional Recurrent Encoder-Decoder
NFV	Network Functions Virtualization
NSUAD	Noisy-Student-Based Unsupervised Anomaly Detection
RNN	Recurrent Neural Network
SCARF	Semantically Coherent Adversarially Robust Features
SL	Supervised Learning
UL	Unsupervised Learning
VAE	Variational Autoencoder
VNF	Virtual Network Function
XAI	Explainable Artificial Intelligence
XGBoost	Extreme Gradient Boosting

performance through various optimizations. Surveys by [1, 2, 19, 20] emphasize how Telecom Service Providers leverage NFV to reduce CAPEX/OPEX while achieving flexible and scalable service provisioning, using high-performance commercial off-the-shelf hardware. These architectures support dynamic VNF instantiation and elastic scaling based on workload demands, allowing efficient resource utilization and service agility. Moreover, authors in [21] and [22] highlight that, with proper VNF placement and scheduling strategies, NFV platforms can meet stringent QoS requirements, rivaling hardware-based solutions. These studies collectively establish that with virtualization-aware design and orchestration (e.g., ETSI MANO), NFV can deliver near-hardware performance suitable for production-grade telecom and 5G deployments.

B. ML-based Fault Detection and Diagnosis

Many works leveraged SL for fault detection and diagnosis, including fault management of NFV environments [7, 23–27]. Fault detection and diagnosis in NFV systems have been addressed using various techniques, including threshold-based methods, statistical approaches, machine learning models, and hybrid techniques [9]. Threshold-based approaches rely on predefined limits for performance metrics, triggering alerts when deviations occur [28]. Statistical methods analyze metric correlations and apply anomaly scoring to detect abnormal behavior [29]. Machine learning-based techniques and hybrid approaches that combine statistical and machine learning techniques [30] have also been applied for fault diagnosis in NFV systems. The work in [24] used Random Forest (RF) as a supervised learning approach for detection and root cause localization of Virtual Machine anomalies in NFV infrastructures including anomalous CPU consumption, memory leaks, excessive number of disk accesses, packet losses, latency increases, and heavy workload. The authors in [25] trained Stacked and Bidirectional LSTM models on a large amount of multivariate time series data collected from the NFV system in cloud environments for early detection of performance degradation and service failures.

Similar to [25], the proposed approach in [26] leveraged sequential deep learning methods like RNNs and Transformers to capture temporal dependencies and sequential patterns in the data for anomaly detection in VNF chains. Authors in [7] generated faulty scenarios of network latency, CPU resource shortage, and excessive disk I/O in the ClearWater IMS test-bed through fault injection tools, and evaluated the performance of RF, extreme gradient boosting (XGBOOST), max-likelihood classification, and K-nearest neighbors (KNN) methods in detecting and classifying the mentioned network failures. Contrastive learning has also been applied for fault detection in tabular data. For example, the study in [31] uses contrastive loss to learn class-specific dependencies for anomaly detection, while [32] introduces SemanticMask, leveraging feature semantics to improve contrastive learning-based anomaly detection. However, contrastive learning methods for anomaly detection often rely on well-constructed positive and negative sample pairs and can suffer from representation collapse or unstable training when labeled anomalies are limited. Recent works have also explored ML-based fault detection in Digital Twins [33, 34]. For example, the study in [33] proposes a deep recurrent graph convolutional model to detect, isolate, and accommodate sensor faults, leveraging spatial-temporal dependencies to improve reliability.

However, as discussed earlier, SL methods require abundant labeled network-fault data that usually is unavailable in sufficient volumes. Some existing works addressed the issue of lack of labeled data by investigating unsupervised ML techniques [6, 10, 24, 35–39]. For example, Authors in [24] evaluated the performance of three shallow unsupervised anomaly detection approaches (Isolation Forest, Local Outlier Factor, and One-Class SVM) on a dataset collected from the Vodafone NFV infrastructure that spans across multiple data centers in 11 European countries. A common Deep UL approach that is shown to outperform shallow UL methods for high-dimensional data 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 [35] for anomaly detection in an NFV architecture, in [10] for anomaly detection in Radio Access Network (RAN) cell trace data collected from multiple Evolved NodeBs, in [36] for detecting anomalous symptoms in 5G RAN, and in [37] for anomaly detection on a cloudified mobile core architecture. However, all these works assume the availability of a training dataset that consists only of normal samples *with no contamination*.

C. Unsupervised Anomaly Detection with Contamination

Some existing studies from the ML field (e.g., [11], [12]) addressed the issue of training-data contamination for unsupervised anomaly detection on multi-variate data. Authors

in [11] and [12] increase robustness against contamination by performing density estimation on the features extracted by the Autoencoder prior to anomaly detection. Similarly, [40] introduces a self-supervised anomaly detection approach based on Latent Outlier Exposure (LOE) that jointly infers normal and anomalous labels during training, leveraging a dual-loss optimization strategy to improve anomaly detection in contaminated datasets. However, contamination still causes a significant degradation in the performance of these methods (partial robustness), while our approach even leverages the contamination to improve anomaly-detection performance ([12] and [40] are among the compared approaches in our experimental evaluations).

D. Weakly-supervised Anomaly Detection

Some works utilize weakly-supervised methods to leverage a limited number of labeled anomalous samples for boosting the anomaly detection performance. For instance, the approach in [41] uses a pair of VAE models, one model trained on the unlabeled data for learning reconstruction of normal samples, and another model which initially is an exact copy of the first trained VAE model, but fine-tuned on the limited number of labeled anomalous samples. At the end, the difference between the anomaly scores calculated by the two VAE models is considered as the ultimate metric for anomaly detection. In the experimental evaluations on eBay's search back-end systems, it is shown that this method outperforms the supervised ensemble method designed by domain experts at eBay. Moreover, there are many weakly-supervised methods proposed in the ML literature for anomaly detection of multi-dimensional data, including DevNet [15], V-DevNet [42], D-SAD [43], and PRO [44]. We describe these approaches with more detail in Section VI.D, as they are implemented and evaluated in our experimental analysis.

E. 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 [45–49]. However, these techniques are not suitable for NFV architectures that have dynamic topologies [7]. Some data-driven methodologies employ a labeled dataset to tackle root cause analysis through a multi-classification approach [50–53]. For instance, authors in [50] address root cause analysis for wireless network failures by treating it as a time-series classification task. They achieve this by converting time-series data into fixed-size feature vectors and then employing an ensemble method that combines XGBoost, rule set learning, attribution models, and graph algorithms. However, these methods require abundant labeled anomalous samples to achieve satisfactory performance.

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 [54]. However, it is shown that these approaches usually lead to poor localization performances [54]. More recently, XAI methods have been utilized for anomaly localization [10], [55]; however, the methods in these works only provide some basic information to facilitate the procedure for the domain experts. In contrast, in our paper, we use XAI algorithms to perform the anomaly-localization task in a fully automated manner.

III. 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 $vnf_t^j \in \mathbb{R}^{1\times n_j}$. 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: $x_t = \{vnf_t^1, vnf_t^2, ..., vnf_t^k\} \in \mathbb{R}^{1\times d}$, 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: $X = \{x_1, x_2, ..., x_n\}$, $X \in \mathbb{R}^{n\times d}$. 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 [12], we assume up to 5% 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 $x_{n+i}^{test} \in \mathbb{R}^{1\times d}$, $i > 0$.
- 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.

IV. Masked Noisy-Student-based Unsupervised Anomaly-Detection (MNSUAD) 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 MNSUAD. The Noisy-Student method was introduced in [13] 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. Afterwards, another neural network (usually larger in size and with more parameters) is trained on the combination of labeled and pseudo-labeled data, and is called the student model. The key idea behind this method, which enables the student model to

TABLE II
Comparison of the related works included in our experiments based on their key characteristics

Anomaly Detection Method	Supervised/ Unsupervised	Localization Approach	Handles Contamination?	Main Contribution
Autoencoder [6, 10]	Unsupervised	Reconstruction Error	No	General anomaly detection in NFV systems
DAGMM [12]	Unsupervised	Reconstruction Error	Partial	Robustness to data contamination
SemanticMask [32]	Unsupervised	Not Provided	No	Structured feature space for tabular data
MSCRED [54]	Unsupervised	Inter-correlation Analysis	No	Time-series anomaly detection
DevNet [15]	Weakly-Supervised	Not Provided	No	Proper utilization of a few labeled anomalies
LOE [40]	Weakly-Supervised	Mutual Information Analysis	Yes	Robustness to noisy datasets via latent label refinement
MNSUAD (ours)	**Unsupervised**	**Mask Permutation / XAI**	**Yes**	**Improved detection/localization in contaminated data**

outperform the teacher model, lies in injecting noise into the input data via data augmentation during the student model's learning process. This augmentation enhances the student's ability to generalize, resulting in improved performance.

A. 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) [12], an unsupervised anomaly-detection method, on a fraction (e.g., 40% in our experiments) of the training data, initially treating the training data as it has no contamination. We train the DAGMM on a fraction 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 data) 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 at this stage.

As we mentioned earlier, DAGMM is shown to be robust against contamination in the training data to some extent [12]; 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, 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:

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

B. 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):

$$thr_1 = percentile^{\rho_1}(E_X) \tag{2}$$

$$X_c = X[E_X < thr_1],\ X_c \in \mathbb{R}^{s\times d} \tag{3}$$

In our experiments, we chose $\rho_1 = 93\%$ to disregard a slightly larger percentage than the highest considered contamination percentage (5%) in the training data. However, X_c still could be 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 extract some pseudo-labeled anomalies from the training samples that have the highest energy calculated by 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:

$$thr_2 = \mu_E + b \times \sigma_E, b \geq 1 \tag{4}$$

In our experiments, we chose $b = 2$, but its value does not have a big impact on the performance of the approach 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 values than thr_2 as the potential contamination data, and denote it as X_{cont}:

$$X_{cont} = X[E_X > thr_2] \tag{5}$$

Now, we define thr_3 to 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):

$$E_{cont} = DAGMM(X_{cont}) \tag{6}$$

$$thr_3 = percentile^{(100-\rho_2)}(E_{cont}) \tag{7}$$

$$X_{anom} = X_{cont}[E_{cont} > thr_3] \tag{8}$$

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 [16], called "data augmentation in feature space (DAiFS)," to add noise to the extracted pseudo-labeled anomalies. We denote the obtained noisy anomalous pseudo-labeled samples

as $\tilde{X}_{anom}$:

$$\tilde{X}_{anom} = DAiFS(X_{anom}) \tag{9}$$

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

$$X_{new} = \{X_c, \tilde{X}_{anom}\},\ X_{new} \in \mathbb{R}^{e \times d} \tag{10}$$

In summary, the three-step thresholding approach ensures effective detection and utilization of minority anomaly samples by leveraging DAGMM-based density estimation within the Noisy-Student framework. Unlike traditional unsupervised methods that disregard data contamination, this method systematically refines the dataset as follows. First, the cleaning step only keeps the low-energy samples to create a contamination-free training set. Second, the pseudo-labeling step extracts high-confidence anomalies based on energy scores. Third, the augmentation step enhances model generalization by adding noise to pseudo-labeled anomalies. This process minimizes false classifications while leveraging anomalies to improve learning, making the approach more robust than standard unsupervised techniques.

C. Student Model

The actual task of anomaly detection is performed by a weakly-supervised student model trained on $X_{new} = \{X_c, \tilde{X}_{anom}\}$, which treats the samples belonging to X_c as normal samples and samples in $\tilde{X}_{anom}$ as known anomalies. For the student model, we propose a novel architecture, consisting of an MAE [15] and a Deviation Network (DevNet), to be trained on X_{new}. The proposed student model is presented in Fig. 1. DevNet [15] is a weakly-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. DevNet uses a neural network to directly learn a single anomaly score from each input sample, which serves as the basis for anomaly detection. However, since this 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. Moreover, DevNet typically requires a large number of training samples to avoid overfitting, which might not be available in our case. To address both of these issues, for the student model, we integrate DevNet with an MAE in a novel architecture, where Devnet learns its anomaly score from the rich features learned by the MAE from the input data, instead of learning it directly from the input samples. MAE is a transformer-based Auto-encoder model that has been shown to learn very generalizable representation from raw data [15]. These generalizable features contain valuable information regarding the anomalous behavior of the input samples, thus, can improve the performance of DevNet and also the scalability of the student model for large NFV systems. In the following, we first describe the architecture of MAE and DevNet individually, and then illustrate how they are integrated together in our proposed student model for performing anomaly detection.

MAE Architecture: MAE is a transformer-based Auto-encoder model that similarly to other transformer models considers the input data as a sequence of tokens (e.g., words of a sentence in NLP or patches of images in computer vision). In our case, metrics of each VNF ($vnf_t^j, j \in \{1, 2, ..., k\}$) are the k input tokens for our MAE model. To convert the input tokens to vectors of equal dimension d_{enc}, we use k linear projection layers with trainable weights (with input size n_j and output size d_{enc}) that transform each $vnf_t^j \in \mathbb{R}^{1 \times n_j}$ to $z_1^j \in \mathbb{R}^{1 \times d_{enc}}$, such that the input $x_t = \{vnf_t^1, vnf_t^2, ..., vnf_t^k\}$ is converted into $z_1(x_t) = \{z_1^1, z_1^2, ..., z_1^k\} \in \mathbb{R}^{k \times d_{enc}}$. In transformer models, positional encoding features [15] are added to the input so the model is informed of the order of tokens in the input sequence (in our case, the model would know which token is related to which one of the VNFs). The positional encoding features ($pos \in \mathbb{R}^{k \times d_{enc}}$) have the same dimension d_{enc} for each token, so after adding the positional encoding we have:

$$z_2(x_t) = z_1(x_t) + pos \tag{11}$$

$$z_2(x_t) \in \mathbb{R}^{k \times d_{enc}} \tag{12}$$

The idea for learning a very generalizable representation in MAE is to randomly mask (remove) a portion of the encoder's input data (a fixed number of the input tokens), and forcing the decoder to reconstruct the entire input data from the incomplete information that is encoded by the encoder. This design follows the masked autoencoder paradigm, which enables the encoder to focus on learning robust representations by reconstructing missing data, a strategy that improves generalization under partial or noisy observations. This is particularly beneficial in NFV environments where input features can be incomplete, delayed, or noisy due to real-time monitoring limitations. For the sake of simplicity, in our approach, we assume that only the metrics of one of the VNFs (one input token) is randomly selected to be masked from the encoder. However, all the following algorithms can be easily extended to cases where the metrics of more than one VNF are masked. So, we first randomly (with uniform distribution) select the number m from $\{1, 2, ..., k\}$, and remove the features related to the m-th VNF from z_2 to obtain z_3:

$$z_3(x_t, m) = \{z_2^1, z_2^2, ..., z_2^{m-1}, z_2^{m+1}, ..., z_2^k\} \tag{13}$$

$$z_3(x_t, m) \in \mathbb{R}^{(k-1) \times d_{enc}} \tag{14}$$

Now, z_3 is the input to the MAE's encoder. This encoder is a standard ViT encoder [15], which is a series of transformer blocks that transforms the input to a latent representation $z_4(x_t, m) \in \mathbb{R}^{(k-1) \times d_{enc}}$. Then, before feeding this latent representation to the decoder to reconstruct the original input, we need to concatenate it with a random trainable vector of dimension d_{enc}, called mask token [15], to inform the decoder that the information in the m-th position (metrics regarding the m-th VNF) is masked. Let's denote the mask token as M_{token}, then the output $z_5(x_t, m)$ will be the concatenation

of $z_4(x_t) = \{z_4^1, z_4^2, ..., z_4^{k-1}\}$ and M_{token}:

$$z_5(x_t, m) = \{z_4^1, z_4^2, ..., z_4^{m-1}, M_{token}, z_4^m, ..., z_4^{k-1}\} \quad (15)$$

$$z_5(x_t, m) \in \mathbb{R}^{k \times d_{enc}} \quad (16)$$

Similar as for the encoder, we need to add positional encoding features to z_5 before feeding it to the decoder:

$$z_6(x_t, m) = z_5(x_t, m) + pos \quad (17)$$

$$z_6(x_t, m) \in \mathbb{R}^{k \times d_{enc}} \quad (18)$$

Now we use a single projection layer (with input size d_{enc} and output size d_{dec}) to project each token to a new dimension d_{dec} to obtain $z_7(x_t, m) \in \mathbb{R}^{k \times d_{dec}}$. $z_7(x_t, m)$ will be the input to our decoder. The decoder is also a series of transformer blocks that transforms $z_7(x_t, m)$ to another representation $z_8(x_t, m) \in \mathbb{R}^{k \times d_{dec}}$. Since MAE's goal is to reconstruct the original input sample, we should also have a final linear projection layer (with input size $k \times d_{dec}$ and output size d) at the end of the decoder to project the decoder's output $z_8(x_t, m)$ to a vector of dimension d, denoted as $z_9(x_t, m)$ (unlike previously mentioned projection layers that were applied separately on each token, this final projection layer is applied once on the entire decoder's output):

$$z_9(x_t, m) = \{z_9^1, z_9^2, ..., z_9^k\} \quad (19)$$

$$z_9^j \in \mathbb{R}^{1 \times n_j} \quad (20)$$

$$z_9(x_t, m) \in \mathbb{R}^{1 \times d}, d = \sum_{j=1}^{k} n_j \quad (21)$$

The MAE is trained according to the following loss function, which is the mean squared error (MSE) of the masked VNF original metrics and those reconstructed by the decoder:

$$L_{MAE} = MSE(vnf_t^m, z_9^m) \quad (22)$$

DevNet Architecture: DevNet is a weakly-supervised anomaly detection method that can utilize a limited number of known anomalies to improve anomaly detection performance. DevNet's architecture is a simple neural network that learns a scalar anomaly score $\Phi(x)$ value from its input samples x. DevNet's neural network architecture consists of three hidden layers with ReLU activation functions, incorporating L2 regularization to prevent overfitting. The final layer outputs an anomaly score using a linear activation function. In the training of DevNet, anomaly scores of normal samples are forced to be close to a reference average, and anomaly scores of known anomalous samples are forced to deviate significantly from this average. To achieve this goal, the following loss function is used for training DevNet:

$$L_{Dev}(x) = \begin{cases} |\frac{\Phi(x) - \mu_R}{\sigma_R}|, & \text{if } x \text{ is normal} \\ max(0, a - \frac{\Phi(x) - \mu_R}{\sigma_R}), & \text{if } x \text{ is anomalous} \end{cases} \quad (23)$$

Same as in [15], 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 $\sigma_R^2 = \frac{1}{l}\sum_{i=1}^{l}(r_i - \mu_R)^2$. Similar to the loss function in [15], 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. In our experiments, we chose $a = 5$ to ensure significant deviation from μ_R for our anomalous samples. Moreover, 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 DevNet are optimized by performing a gradient descent step.

Integrating MAE with DevNet: In our proposed student model depicted in Fig. 1, we integrate MAE with DevNet and train the entire model on X_{new} in an end-to-end fashion. More specifically, rather than using raw input samples, the model first processes them through the MAE before passing features into DevNet. The MAE encoder extracts latent features, while reconstruction error features are computed based on the difference between the input and output of MAE. Both the latent and reconstruction error features are then fed into DevNet to compute anomaly scores of the samples. So, the new input of DevNet, $x'(x_t, m)$, is:

$$x'(x_t, m) = \{z_6(x_t, m), rec_{err}(x_t, m)\} \in \mathbb{R}^{1 \times (k \times d_{enc} + k)} \quad (24)$$

$$rec_{err}(x_t, m) = \{||z_9^i - x_t^i||_2, i \in \{1, 2, ..., k\}\} \quad (25)$$

The student model is trained according to the following loss function, which is a combination of the loss functions of MAE and DevNet:

$$L(x_t) = \begin{cases} \alpha_1 \times L_{MAE} + |\frac{\Phi(x'(x_t, m)) - \mu_R}{\sigma_R}|, & \text{if } x_t \in X_c \\ max(0, a - \frac{\Phi(x'(x_t, m)) - \mu_R}{\sigma_R}), & \text{if } x_t \in \tilde{X}_{anom} \end{cases} \quad (26)$$

Where $L_{MAE} = MSE(vnf_t^m, z_9^m)$, and $\alpha_1 > 0$ determines the effect of the two components of loss function for normal samples. At inference time, we calculate the average of $\Phi(x'(x_a, m))$ for a test sample x_a when m varies from 1 to k, and the test sample is considered as an anomaly if this average value exceeds a predefined threshold (e.g., $a/2$).

Regarding the scalability of our proposed method, it is important to highlight that the transformer architecture used for feature representation efficiently handles large input sizes. The self-attention mechanism, with a complexity of $O(k^2)$ relative to the number of input tokens, remains computationally feasible due to the high parallelism enabled by modern hardware accelerators. Unlike sequential models, transformers process all input tokens simultaneously, ensuring that an increase in the number of VNFs has a manageable impact on execution time. The architecture's ability to represent complex relationships among tokens further enhances computational

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3582195

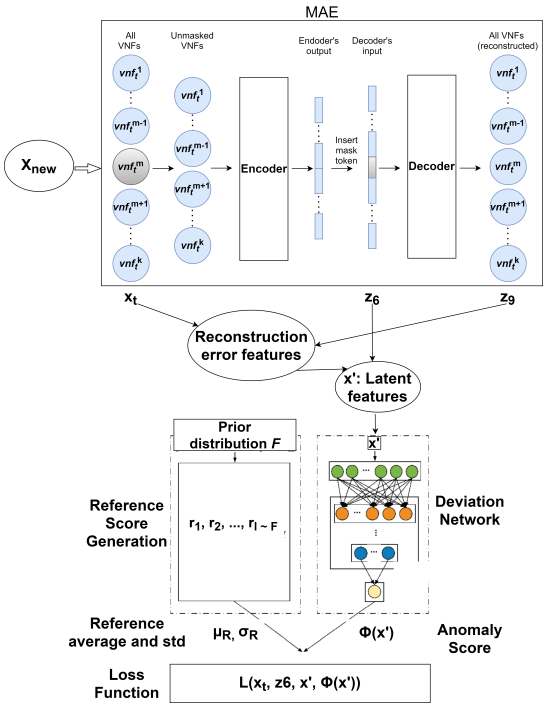

Fig. 1. The proposed student model to be trained on X_{new}. The latent features (low-dimensional representation and reconstruction error features) are extracted from the input by the MAE and are fed to the Deviation Network. Then, the anomaly score output of the Deviation Network alongside average anomaly 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.

efficiency without introducing excessive additional parameters.

V. 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 assume the reasonable assumption that as the probability of multiple simultaneous failures is very small, the anomalous behavior originates only from one of the VNFs, and our localization objective is to pinpoint the corresponding VNF. We first present an anomaly localization algorithm called "Mask Permutation" that utilizes the outputs of the our anomaly detection model for different masking scenarios to determine the anomaly location (exclusive for our anomaly detection model MNSUAD). Furthermore, we propose two more generic algorithms for anomaly localization based on XAI methods that are applicable to any black-box ML-based anomaly detection model. All three algorithms perform the anomaly localization task in a fully-automated manner without requiring domain experts' intervention.

A. Localization with Mask Permutation

The goal of Mask Permutation is to observe the output of the detection model for different masking scenarios and determine the anomaly location based on the information provided by these different outputs. Let us assume x_a is a detected anomaly sample. If vnf^j is the anomaly location, we expect the following behavior from our detection model:

When metrics of vnf^j are masked: if metrics of vnf^j are masked, we would expect the reconstruction error for vnf^j to be high (big difference between predicted values and actual values of vnf^j's metrics). The reconstruction error can be calculated according to the following:

$$\Omega(x_a, j) = MSE(vnf_a^j, z_9^j(x_a, j)) \tag{27}$$

Moreover, since the metrics of the anomalous VNF are not given to the MAE, we would expect a relatively lower anomaly score when metrics of vnf^j are masked (we expect $\phi(x_a, j) = |\frac{\Phi(x'(x_a,j))-\mu_R}{\sigma_R}|$ to be lower relative to when the metrics of a VNF other than vnf^j are masked).

When metrics of vnf^j are not masked: Let us assume metrics of vnf^i are masked ($i \neq j$). Then, the anomaly score calculated by the detection model would be $\Phi(x'(x_a, i))$. Now, assume we replace the metrics of vnf^j (the anomalous VNF) with values that the MAE predicts for it when it is masked (i.e., $z_9^j(x_a, j)$) to create a synthetic sample $x_{syn}(j)$:

$$x_{syn}(j) = \{vnf_a^1, ..., vnf_a^{j-1}, z_9^j(x_a, j), vnf_a^{j+1}, ..., vnf_a^k\} \tag{28}$$

Since the metrics of the anomalous vnf^j are replaced with their predicted values $z_9^j(x_a, j)$, we expect the resulting synthetic sample to exhibit a lower anomaly score than the original one when another VNF ($vnf^i, i \neq j$) is masked; in other words: $|\frac{\Phi(x'(x_{syn}(j),i))-\mu_R}{\sigma_R}| \leq |\frac{\Phi(x'(x_a,i))-\mu_R}{\sigma_R}|$. We denote this difference as $\Delta\Phi(x_a, j, i)$:

$$\phi(x_a, i) = |\frac{\Phi(x'(x_a, i)) - \mu_R}{\sigma_R}| \tag{29}$$

$$\phi(x_{syn}(j), i) = |\frac{\Phi(x'(x_{syn}(j), i)) - \mu_R}{\sigma_R}| \tag{30}$$

$$\Delta\Phi(x_a, j, i) = \phi(x_a, i) - \phi(x_{syn}(j), i) \tag{31}$$

Based on these expectations, we define a global localization score (GLS) for each vnf^j and denote it as $G(j)$:

$$G(j) = \alpha_2 \times \Omega(x_a, j) - \phi(x_a, j) + \frac{\sum_{i \neq j} \Delta\Phi(x_a, j, i)}{k-1} \tag{32}$$

Where $\alpha_2 > 0$ adjusts the effect of the reconstruction error component of the GLS. The higher the $G(j)$ score is, the higher is the probability that vnf^j is the location of the system's anomalous behavior. We calculate $G(j)$ for all the k VNFs ($j \in \{1, 2, ..., k\}$) and choose the VNF with the highest GLS to be the anomaly location.

Although our current implementation does not leverage parallelism, as the number of VNFs is small in our experiments, the mask permutation method employed in our approach is inherently parallelizable due to the independence of masking scenarios. Each masking scenario can be processed independently, allowing the workload to be distributed across multiple computational units, such as GPUs or processing nodes. This parallelism significantly reduces computational overhead when scaling to larger datasets, ensuring efficient execution without compromising performance.

B. Localization with SHAP

The complete procedure of anomaly localization with the SHAP method [56] is presented in Algorithm 1. To determine the contribution of each feature to the decision of the anomaly detection model, SHAP gets the detected anomaly sample x_a, the anomaly detection model (e.g., MNSUAD), 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 highly 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_{new} (e.g., with 50 clusters) and gave the corresponding centroids $Cents$ to the SHAP method as background data to achieve a reasonable execution time. $Shap_{x_a} \in \mathbb{R}^{1\times d}$, the output of the SHAP method for the detected anomaly x_a, 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 $Shap_{x_a}$ corresponding to the features of vnf^j and denote it as $Shap^j_{x_a} \in \mathbb{R}^{1\times n_j}$. Finally, the anomaly location is chosen to be the VNF whose SHAP values ($Shap^j_{x_a}$) have the highest first norm. We have also considered a scaling factor $c_j \in C$ for each $Shap^j_{x_a}$ 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 being 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. The SHAP localization method can be applied to any ML-based anomaly detection model that calculates an anomaly score for any given data sample.

Algorithm 1 Anomaly Localization with SHAP

1: **function** SHAPLOCAL($x_a, MNSUAD, Cents, C$) ▷ Input: detected anomaly x_a, anomaly detection model (MNSUAD), background data centroids (Cents), scaling factors (C)
2: $Shap_{x_a} \leftarrow SHAP(x_a, MNSUAD, Cents)$ ▷ Compute SHAP values for x_a using centroids as background data
3: $Shap^j_{x_a} \leftarrow$ elements of $Shap_{x_a}$ related to vnf^j ▷ Extract SHAP values corresponding to each VNF
4: $location \leftarrow argmax_j(c_j||Shap^j_{x_a}||_1)$ ▷ Identify VNF with the highest scaled SHAP first norm
5: **return** $location$ ▷ Return the VNF most responsible for the anomaly

C. Localization with Cluster Permutation

SHAP is a general model-agnostic XAI method that is designed to be applicable to any ML task. For improving the accuracy and reducing the computational complexity of our localization method, we propose our own novel XAI algorithm (called Cluster Permutation) that is specifically designed for the task of anomaly localization but is still applicable to any anomaly detection model. 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 $x_a = \{vnf^1_a, vnf^2_a, ..., vnf^k_a\}$, we first define the metrics related to vnf^j in x_a as $\underline{x_a(j)}$, and the remaining metrics in x_a as $\widehat{x_a(j)}$. We do the same separation on all the samples in X_c (a total of s samples) 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 $\widehat{x_a(j)}$ 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, replacing its current metrics in the detected anomaly with those from normal neighbors should reduce the anomaly's impact. As a result, the newly created sample (x^{mod}_a) would exhibit a lower degree of anomalous behavior. Consequently, 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. The vnf^j with the highest decrease in anomaly score, when its metrics are replaced by normal neighbors' metrics, is chosen as the anomaly location. However, if none of the VNFs show a decrease in anomaly score, or if the decrease is less than a given threshold (Thr), the algorithm outputs "undecided". So, in this case, we ran the ShapLocal algorithm (Algorithm 1) to determine the anomaly location. The Cluster Permutation method is also applicable to any ML-based anomaly detection model that computes an anomaly score for a given data sample.

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 if 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.

VI. Experimental Results

In this section, we first introduce the three datasets on which we evaluated our anomaly detection and localization methods. Then, we describe the-state-of-the-art approaches that are compared with our proposed methods. Following this, we conduct a detailed examination and analysis of the results obtained from applying our methods to detect and localize anomalies within these datasets.

A. Datasets

ITU Dataset: Our first dataset is from the "ITU AI/ML in 5G" challenge [57]. 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. 2(a) and consists of 5 VNFs: two IP core nodes (TR-01 and TR-02), two internet gateway routers (IntGW-01 and IntGW-02), 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

Algorithm 2 Anomaly Localization with Cluster Permutation

function CLUSTERPER($x_a, X_c, MNSUAD, Thr$) ▷ Detected anomaly, clean dataset, model, threshold
  $\Delta\Phi \leftarrow \{\}$ ▷ Store anomaly score changes for each VNF
  for $j = 1$ to k **do** ▷ Iterate over VNFs
    $\overbrace{x_a(j)} := \{vnf_a^j\} \in \mathbb{R}^{1\times n_j}$ ▷ Extract vnf^j features
    $\overbrace{x_a(j)} := (x_a \setminus \overbrace{x_a(j)})$ ▷ Remove vnf^j from sample
    Run KNN in $\overbrace{X_c(j)}$ for $\overbrace{x_a(j)}$
    Find nearest neighbors: $\overbrace{X_{cp}(j)}, p \in \mathbb{NEIGHB}$
    $\Delta\phi \leftarrow 0$, $c \leftarrow 0$
    for $p \in \mathbb{NEIGHB}$ **do**
      $x_a^{mod} \leftarrow \{vnf_a^1, ..., \overbrace{x_{cp}(j)}, ..., vnf_a^k\}$
      $\Delta \leftarrow |\frac{\Phi(x_a)-\mu_R}{\sigma_R}| - |\frac{\Phi(x_a^{mod})-\mu_R}{\sigma_R}|$
      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
      $\Delta\Phi.append(0)$
  $location \leftarrow argmax(\Delta\Phi)$ ▷ Select most affected VNF
  if $(max(\Delta\Phi) > Thr)$ **then**
    $output \leftarrow location$
  else
    $output \leftarrow ShapLocal(x_a)$ ▷ Fallback to SHAP
  return $output$

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 [58]. For this dataset, the training data includes 3870 normal samples, and the test data includes 3505 normal samples and 1112 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. When extracting pseudo-labeled anomalies from this training data according to the procedure in Section IV.B, we observed that, as expected, the number of extracted pseudo-labeled anomalies was larger for higher contamination rates. Specifically, the average number of extracted pseudo-labeled anomalies was 25, 41, 57, 65, and 72 for contamination rates of 1%, 2%, 3%, 4%, and 5%, respectively.

MEC Dataset: Our second dataset is from our experimental NFV testbed depicted in Fig. 2(b), and consists of 4 open-source VNFs: a Firewall (iptables [59]), an intrusion detection System (Suricata [60]), a deep packet inspector (nDPI [61]), and a flow monitor (ntopng [62]). We have adopted the topology in Fig. 2(b) from [27] and implemented it on the SAVI testbed [63], 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 [27] for a complete list of collected metrics). We injected one of the following faults to one of the VNFs periodically to generate faulty instances to evaluate our anomaly-detection techniques: 1) *CPU stress* by the stress-ng [64] tool that increases the CPU usage in the VM, 2) *disk stress* by the stress-ng [64] tool that increases the disk usage in the VM, and 3) *network stress* by the tc [65] 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 200 anomalous samples. Similarly, we extract pseudo-labeled anomalies from the training data of this dataset according to the procedure in Section IV.B. The number of extracted pseudo-labeled anomalies (on average for multiple runs) was 28, 52, 72, 84, and 89 for 1%, 2%, 3%, 4%, and 5% contamination rates, respectively.

IMS Dataset: This dataset is collected by [17] from the ClearWater project, which is an NFV-based open source implementation of an IMS for cloud platforms. The IMS consists of the ten components depicted in Fig. 2(c), each deployed on a docker container. The dataset contains the performance metrics of the three most important components, namely Bono, Sprout, and Homestead, which are responsible for controlling sessions initiated by users. The performance metrics are collected from the containers every 5 seconds. For generating faulty data instances, they inject CPU fault, Memory fault, or i/o fault randomly to one of these three components (at each fault scenario, one of these fault types is injected to one of the three main components). The training data here includes 480 normal samples and the test data has 120 normal samples and 700 anomalous samples. Similarly, we extract pseudo-labeled anomalies from the training data of this dataset according to the procedure in Section IV.B. The number of extracted pseudo-labeled anomalies (on average for multiple runs) was 3, 6, 11, 14, and 16 for 1%, 2%, 3%, 4%, and 5% contamination rates, respectively.

Data Preprocessing: We perform the necessary data preprocessing 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. Feature statistics (mean ± standard deviation) for different fault type scenarios for our three datasets are reported in TABLE III.

B. Compared Approaches

Baseline: The baseline approach used to compare our detection approach with is the Autoencoder-based anomaly-detection algorithm in [6] and [10], 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. Since [6] and [10] have worked with different datasets that are not publicly available, we have designed the best Autoencoder architecture for our datasets by trying different architectures,

TABLE III
Feature statistics (mean ± standard deviation) for different fault types across the three NFV datasets. Feature values are normalized to [0,1] but reported in percentage format for better interpretability.

Dataset		CPU Utilization				Incoming Packet Rate			Outgoing Packet Rate	
ITU	**Fault Type**	IntGW-01	IntGW-02	RR-01	TR-01	IntGW-01	IntGW-02	TR-01	IntGW-02	TR-02
	Normal	92.5 ± 2.5	93.3 ± 2.2	94.3 ± 2.4	89.2 ± 2.0	85.3 ± 3.0	50.0 ± 2.8	49.5 ± 3.0	85.7 ± 3.5	50.0 ± 3.0
	Node Failure	72.0 ± 18.5	74.0 ± 17.8	79.5 ± 16.2	73.8 ± 14.5	84.8 ± 4.0	49.0 ± 18.8	50.2 ± 19.1	72.1 ± 17.0	49.5 ± 18.3
	Interface Failure	89.5 ± 7.0	89.7 ± 7.2	94.5 ± 4.0	88.9 ± 7.5	82.0 ± 18.5	48.5 ± 20.7	48.3 ± 21.3	73.0 ± 19.8	48.7 ± 20.5
	Packet Loss	91.7 ± 4.5	92.1 ± 4.8	94.6 ± 3.5	89.0 ± 4.3	83.9 ± 8.5	48.3 ± 7.8	48.1 ± 9.0	83.2 ± 7.3	49.0 ± 7.5
Dataset		**CPU Utilization**			**Disk Utilization**			**Outgoing Packet Rate**		
MEC	**Fault Type**	iptables	Suricata	nDPI	iptables	Suricata	nDPI	iptables	Suricata	nDPI
	Normal	81.3 ± 6.0	82.0 ± 6.7	83.1 ± 7.3	78.5 ± 12.8	76.2 ± 9.6	51.0 ± 12.6	50.2 ± 21.2	66.0 ± 18.6	51.1 ± 12.8
	CPU Stress	83.0 ± 8.2	83.7 ± 5.7	84.9 ± 6.2	80.3 ± 8.4	76.8 ± 11.3	51.2 ± 16.0	50.5 ± 23.5	67.1 ± 19.1	51.4 ± 15.2
	Disk Stress	81.0 ± 11.5	81.7 ± 9.3	82.9 ± 7.6	80.0 ± 15.3	78.6 ± 13.0	51.7 ± 18.3	50.1 ± 18.3	65.2 ± 19.5	50.8 ± 8.5
	Network Stress	81.5 ± 8.6	82.3 ± 8.1	83.4 ± 7.5	78.2 ± 16.9	75.8 ± 9.2	50.9 ± 15.3	50.2 ± 23.1	65.7 ± 17.4	51.1 ± 14.7
Dataset		**CPU Utilization**			**Memory Utilization**			**Outgoing Packet Rate**		
IMS	**Fault Type**	bono	homestead	sprout	bono	homestead	sprout	bono	homestead	sprout
	Normal	90.4 ± 2.8	91.0 ± 2.5	92.2 ± 3.0	88.0 ± 4.6	85.9 ± 3.4	51.5 ± 12.7	60.9 ± 8.0	75.5 ± 8.5	61.7 ± 9.8
	CPU Fault	92.0 ± 5.5	92.6 ± 5.2	93.8 ± 4.1	89.5 ± 7.2	86.7 ± 6.1	51.0 ± 15.2	61.3 ± 16.2	76.4 ± 12.5	62.0 ± 12.8
	Memory Fault	91.2 ± 4.1	91.8 ± 3.5	93.0 ± 3.2	88.6 ± 9.1	86.4 ± 6.6	50.8 ± 18.2	60.7 ± 11.7	75.9 ± 9.2	61.5 ± 10.5
	I/O Fault	90.9 ± 3.5	91.4 ± 4.1	92.6 ± 2.5	88.3 ± 6.6	85.8 ± 4.3	50.6 ± 16.2	60.6 ± 8.3	75.6 ± 11.8	61.2 ± 9.8

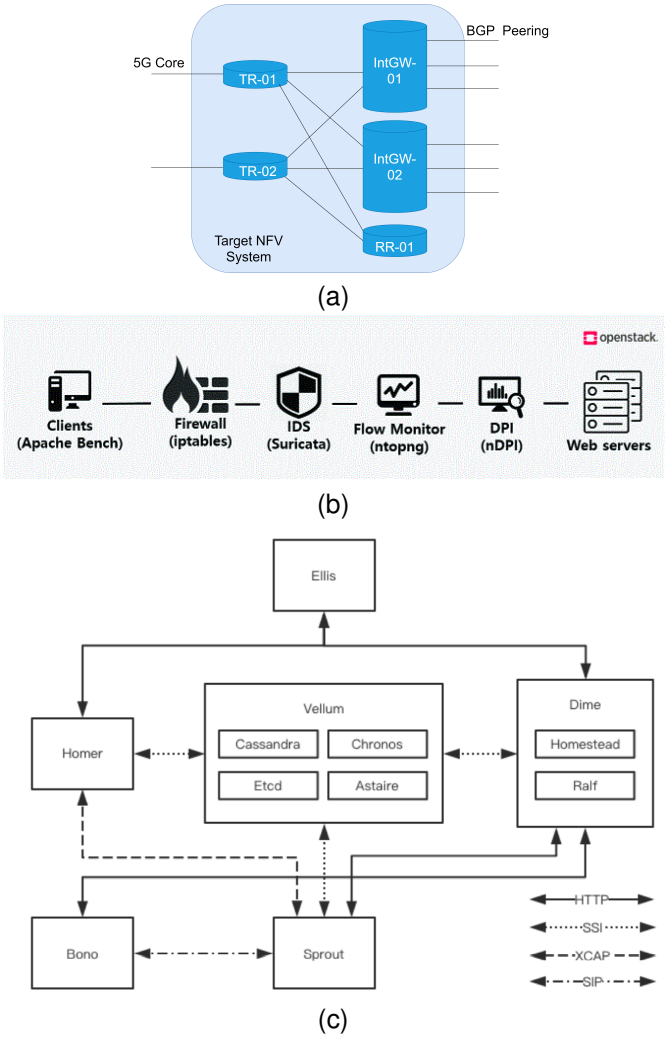

Fig. 2. NFV system of (a) ITU Dataset, (b) MEC Dataset [27], and (c) IMS Dataset [6]

from very shallow to very deep, to reach the best possible outcome for the Baseline. Moreover, similar to [10], we added L2-regularization to the Autoencoder network to improve the model's robustness against contamination in the training data.

For the baseline localization approach, we implemented the conventional method where the VNF whose features have been reconstructed more poorly is chosen as the location of the fault (Let $x'_t = \{vnf'^1_t, vnf'^2_t, ..., vnf'^k_t\}$ be the reconstructed output of a vanilla Autoencoder for the input sample $x_t = \{vnf^1_t, vnf^2_t, ..., vnf^k_t\}$. We calculate the reconstruction error $||vnf'^j_t - vnf^j_t||_2$ for each vnf^j, and choose the VNF with the highest reconstruction error as the fault location).

DAGMM: Our teacher model, DAGMM [12], 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.

MSCRED: Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) was proposed in [54] 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.

LOE: This method from [40] applies the concept of Latent Outlier Exposure (LOE) to detect anomalies on datasets contaminated with unlabeled anomalies. This method jointly infers binary labels to each unlabeled sample while updating the model parameters by incorporating two loss functions: a normal loss, which maximizes mutual information between complementary feature subsets, and an anomaly loss, which minimizes mutual information to emphasize anomalies. Both losses share the same model parameters, allowing the anomaly loss to inform the normal loss about regions in feature space where anomalies are likely to occur. By using an assumed contamination ratio (as input) to guide the optimization, LOE jointly refines the anomaly labels and model parameters, making it robust to noisy datasets. Although the contamination

ratio is treated as a hyperparameter in LOE and can be adjusted by the method, we provide the true contamination ratio of the dataset in our experiments to ensure the best possible performance. Additionally, the paper recommends specific loss functions based on the type of data. Following these guidelines, we use Internal Contrastive Learning (ICL) [31] as the loss function for tabular data. For anomaly localization in this approach, we adopt the localization procedure outlined in ICL [31], adapting it to our context. Specifically, we calculate the mutual information between each VNF's features and the features of all other VNFs, and identifying the VNF with the largest deviation from normal mutual information patterns as the most anomalous.

SemanticMask: SemanticMask [32] is a semantic-aware data augmentation method designed for contrastive learning-based anomaly detection in tabular data. It incorporates three key steps to enhance the quality of representations. First, SemanticMask groups features with similar semantics into clusters. In our problem, we consider features of one specific VNF as a cluster. Second, the method uses a data augmentation module that creates positive pairs for contrastive learning by dividing the clusters into two disjoint subsets and applying a cluster-wise masking strategy. Finally, the augmented views are passed through an encoder, and a contrastive loss is calculated to encourage the representations of similar instances to be close. The resulting structured feature space, with tightly clustered normal samples and well-preserved feature correlations, allows anomalies to be identified as samples with high Mahalanobis distance from the training distribution. Since the paper does not propose an anomaly localization method, we only evaluated it for anomaly detection.

Our Proposed Methods: MNSUAD is our proposed anomaly-detection solution described in Section IV. SHAP (Algorithm 1), ClusterPer (Algorithm 2), and MaskPer (Section V.A) are our proposed anomaly-localization approaches that utilize MNSUAD 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 IV. For training our MNSUAD model, we used the Adam optimizer with dataset-specific learning rates. Specifically, for the ITU dataset, we trained the model for 150 epochs with a learning rate of 1e-4 and a batch size of 128. For the MEC and IMS datasets, a slightly smaller learning rate of 5e-5 and 100 training epochs were used, as these datasets exhibited faster convergence. The reconstruction loss was computed over the unmasked tokens only, and dropout with a rate of 0.1 was applied after each encoder and decoder layer to prevent overfitting. In the first row of TABLE IV, 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. In the next rows of TABLE IV, 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 all existing methods, and this degradation becomes more severe as δ increases. Also, in all datasets, DAGMM and LOE have a better average performance than Baseline, MSCRED and SemanticMask (especially in the ITU dataset), and their performance degrades less significantly compared to the Baseline, MSCRED and SemanticMask as δ increases. This is due to the fact that DAGMM and LOE explicitly account for robustness against contamination in the training data. Even though we provided the LOE method with the true contamination ratio in each experiment, contamination in the training data still degraded its performance. This was because a considerable portion of anomalies in the training data were incorrectly assigned to the normal class during the method's latent label inference steps. As a result, the representations learned by the normal loss function were partially corrupted, reducing the model's ability to effectively distinguish between normal and anomalous samples.

Moreover, our detection approach MNSUAD outperforms all existing methods when δ is 1%, and, unlike the other approaches, its detection performance improves as δ increases up to 5%. This is due to MNSUAD's ability to leverage the extracted pseudo-labeled anomalies in the contaminated training data rather than being negatively affected by them. More specifically, increasing δ (percentage of anomalous samples) in the training data has both a negative and a positive effect on MNSUAD's performance. The negative effect is that as δ increases, the performance of MNSUAD's teacher model (DAGMM) degrades. The positive effect is that MNSUAD extracts a greater number of pseudo-labeled anomalies at higher values of δ, enabling the student model to learn better anomaly representations. Our experimental results show that up to $\delta = 5\%$, the positive effect dominates, leading to a performance improvement compared to lower contamination levels. Unlike prior methods that are significantly impacted by contamination, MNSUAD takes advantage of the anomalous samples in the training data. This allows it to generalize better under real-world conditions where training sets are rarely clean.

However, beyond a certain contamination threshold, the degradation in the teacher model's performance outweighs the benefits of pseudo-labeling. For instance, when we increased δ to 8%, the teacher model's F1-score declined to 65.4%, 64.1%, and 67.5% in the ITU, MEC, and IMS datasets, respectively. Consequently, MNSUAD's F1-score dropped to 81.1%, 88.5%, and 91.4%, which is notably lower than its performance at $\delta = 5\%$. This indicates that while MNSUAD is robust to moderate levels of contamination, excessive contamination reduces its effectiveness. For very high contamination levels, Active Learning [66] approaches that incorporate expert labeling might be more suitable.

D. Comparison of Different Weakly-supervised Learning Anomaly Detection Methods

In our experiments so far, our proposed MNSUAD method has been the only approach that is trained on X_{new}, and all the other compared approaches were trained on the original contaminated training dataset . To show that the performance

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

δ	ITU Dataset																	
	Baseline			DAGMM			MSCRED			LOE			SemanticMask			MNSUAD (ours)		
	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	84.2	83.7	83.9	82.3	80.7	81.5	-	-	-
1	77.6	31.4	44.8	80.1	80.2	80.1	71.5	55.9	62.3	81.6	81.2	81.4	77.0	75.3	76.1	**84.8**	**84.3**	**84.5**
2	76.9	31.1	44.5	79.3	78.9	79.1	68.6	49.3	57.3	77.1	77.3	77.2	63.4	58.6	61.0	**86.3**	**85.0**	**85.6**
3	76.1	30.0	43.0	75.8	74.8	75.3	57.7	42.1	48.6	75.9	75.0	75.4	58.0	61.3	59.6	**86.7**	**85.3**	**86.0**
4	75.8	29.5	42.6	73.5	72.6	72.9	41.5	32.6	36.5	76.3	76.6	76.5	54.4	53.9	54.2	**87.7**	**85.9**	**86.8**
5	74.5	29.2	42.0	71.8	71.7	71.7	27.8	24.3	25.9	73.5	73.6	73.5	48.7	49.5	49.1	**87.1**	**85.6**	**86.3**
δ	MEC Dataset																	
	Baseline			DAGMM			MSCRED			LOE			SemanticMask			MNSUAD (ours)		
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1
0	83.4	78.5	80.9	85.8	83.7	84.7	76.2	73.3	74.7	85.2	83.0	84.1	85.1	82.8	83.9	-	-	-
1	80.3	71.2	75.5	82.1	78.8	80.4	65.6	62.7	64.0	84.6	83.3	83.9	74.6	75.1	74.7	**93.0**	**89.7**	**91.3**
2	78.9	64.1	70.7	77.3	75.2	76.2	49.3	48.8	49.1	79.6	79.1	79.3	68.5	69.3	69.0	**94.2**	**93.5**	**93.8**
3	77.2	60.9	68.1	74.5	73.6	74.1	41.2	38.7	39.9	75.8	74.6	75.2	54.3	57.6	55.9	**94.8**	**93.2**	**94.0**
4	70.8	59.5	64.6	73.6	71.8	72.7	33.9	30.5	32.1	75.1	73.9	74.5	50.9	53.7	52.2	**95.3**	**93.5**	**94.4**
5	66.7	52.6	58.8	70.3	69.9	70.1	20.3	19.7	20.0	75.4	74.3	74.9	41.6	42.0	41.8	**95.2**	**94.0**	**94.6**
δ	IMS Dataset																	
	Baseline			DAGMM			MSCRED			LOE			SemanticMask			MNSUAD (ours)		
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1
0	98.8	97.5	98.1	98.1	96.9	97.5	100	100	100	95.5	95.9	95.7	92.7	92.5	92.6	-	-	-
1	74.8	74.1	74.4	84.8	81.4	83.1	65.7	64.3	65.0	81.6	81.1	81.3	90.3	88.6	89.4	**95.1**	**93.4**	**94.2**
2	68.6	67.1	67.8	80.1	78.3	79.2	59.1	58.2	58.6	77.9	78.4	78.1	71.6	70.3	71.0	**96.4**	**95.2**	**95.8**
3	64.2	62.3	63.2	78.2	75.4	76.8	52.4	51.0	51.7	73.2	73.6	73.5	64.1	66.0	64.9	**96.6**	**95.3**	**95.9**
4	54.9	50.1	52.4	73.5	71.6	72.5	47.1	47.0	47.1	70.4	70.0	70.2	55.9	55.8	55.9	**96.6**	**95.5**	**96.1**
5	50.8	47.3	49.0	72.7	70.9	71.8	42.4	42.2	42.3	70.9	70.6	70.7	48.4	47.9	48.2	**96.5**	**95.3**	**95.9**

improvement achieved by MNSUAD is not just due to cleaning the contaminated dataset, in this section, we compare our student model in MNSUAD with other state-of-the-art weakly-supervised anomaly detection methods when they are also trained on the newly obtained dataset X_{new}. We implemented the following weakly-supervised anomaly detection methods for comparison with our student model:

DevNet [15]: As we described earlier, our student model integrates DevNet with MAE for a better feature encoding. In this algorithm, we directly apply the DevNet on the raw input samples. Comparing the result of this algorithm with our student model can better illustrate the effectiveness of using an MAE for learning a richer feature representation in MNSUAD.

V-DevNet [42]: V-DevNet is a modified version of DevNet, where the reference scores used in the deviation loss of DevNet are calculated by a Variational Auto-encoder (VAE), instead of generating them according to a prior probability.

D-SAD [43]: In this algorithm, a neural network is trained to transform the input space to a lower-dimensional output space with the goal that the output of unlabeled samples should be as close as possible to a predetermined center point, while maximizing the quadratic distance of known anomaly samples from this multi-dimensional center point. This algorithm can be considered as an extension of the Deep one-class classification method proposed in [67].

PRO [44]: In this algorithm, anomaly detection is reformulated as a pairwise relation prediction task in order to take advantage of a few available labeled anomaly samples. More specifically, the algorithm tries to train a supervised learning method on unordered random instance pairs labeled as anomaly-anomaly, anomaly-unlabeled, or unlabeled-unlabeled.

ContLeaEncod: In this approach, the SCARF [68] contrastive learning encoder (ContLeaEncod) is trained to map samples into a latent space where augmented views of the same sample are close, and normal and anomalous samples are farther apart. We then fit two Gaussian Mixture Models (GMMs) in this latent space: GMM-Normal, trained exclusively on embeddings of normal samples, and GMM-Anomaly, trained on embeddings of the known anomalies. For each test sample, its latent embedding is passed through both GMMs to compute the normal likelihood $p_{\text{normal}}(z)$ and anomaly likelihood $p_{\text{anomaly}}(z)$. The anomaly score is calculated as:

$$S(z) = \log p_{\text{anomaly}}(z) - \log p_{\text{normal}}(z),$$

and samples are classified as anomalies if $S(z) > \tau$, where τ is a predefined threshold.

NSUAD [18]: NSUAD is the proposed algorithm in our previous work, where DevNet is integrated with a vanilla auto-encoder to perform the anomaly detection task.

The anomaly detection performance of the mentioned weakly-supervised methods (namely, Precision, Recall, and F1-score) are presented in TABLE V. We can see that DevNet and V-DevNet have a very similar performance, so we can conclude that calculating the reference scores through a data-driven approach (e.g., VAE) instead of a prior distribution has no significant effect on the anomaly detection performance. Among the compared approaches, the contrastive learning methods (PRO and ContLeaEncod) have the lowest average F1-score. This is expected as we only have access to pseudo-labeled anomalies rather than accurately labeled data samples. Thus, generating different instance pairs from the pseudo-

TABLE V
Anomaly-detection results of different weakly-supervised methods trained on X_{new} with the contamination percentage (δ) varying from 1% to 5%

δ	ITU Dataset																				
	DevNet			V-DevNet			D-SAD			PRO			ContLeaEncod			NSUAD (ours)			MNSUAD (ours)		
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1
1	80.4	78.5	79.4	80.5	78.7	79.6	81.5	79.2	80.3	73.4	75.5	74.4	78.3	78.5	78.4	82.7	81.6	82.1	**84.8**	**84.3**	**84.5**
2	82.2	81.1	81.6	82.2	81.3	81.7	82.9	81.6	82.2	77.9	78.6	78.2	80.9	81.3	81.1	84.1	83.5	83.8	**86.3**	**85.0**	**85.6**
3	83.7	82.0	82.8	83.6	82.0	82.8	84.4	82.5	83.4	78.4	79.1	78.7	82.4	81.8	81.6	85.4	83.8	84.6	**86.7**	**85.3**	**86.0**
4	85.2	83.6	84.4	85.1	83.5	84.3	85.9	84.0	84.9	79.3	80.0	79.6	82.9	81.8	82.3	86.1	84.4	85.2	**87.7**	**85.9**	**86.8**
5	85.8	84.1	85.0	85.8	84.2	85.0	86.4	84.8	85.6	79.2	79.8	79.5	83.6	84.0	83.8	86.5	84.4	85.4	**87.1**	**85.6**	**86.3**
δ	MEC Dataset																				
	DevNet			V-DevNet			D-SAD			PRO			ContLeaEncod			NSUAD (ours)			MNSUAD (ours)		
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1
1	82.8	82.1	82.5	82.7	82.1	82.4	83.0	82.6	82.8	66.5	66.3	66.4	80.4	79.7	80.0	86.1	83.5	84.8	**93.0**	**89.7**	**91.3**
2	85.2	84.3	84.7	85.1	84.3	84.7	87.3	86.5	86.9	72.1	72.2	72.2	82.3	82.0	82.1	90.4	87.1	88.7	**94.2**	**93.5**	**93.8**
3	89.6	87.8	88.7	89.6	87.8	88.7	90.8	90.0	90.4	78.4	78.2	78.3	84.0	83.6	83.8	92.7	90.5	91.6	**94.8**	**93.2**	**94.0**
4	91.2	90.4	90.8	91.3	90.4	90.9	92.6	91.3	91.9	79.2	79.5	79.3	87.1	86.8	86.9	93.2	91.1	92.1	**95.3**	**93.5**	**94.4**
5	91.6	91.0	91.3	91.7	91.2	91.4	92.5	91.4	91.9	78.5	78.4	78.5	87.9	87.8	87.8	93.1	91.4	92.2	**95.2**	**94.0**	**94.6**
δ	IMS Dataset																				
	DevNet			V-DevNet			D-SAD			PRO			ContLeaEncod			NSUAD (ours)			MNSUAD (ours)		
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1
1	87.8	87.3	87.5	87.8	87.4	87.6	88.2	87.6	87.9	81.3	81.0	81.2	80.5	80.1	80.3	90.3	89.1	89.7	**95.1**	**93.4**	**94.2**
2	89.4	88.7	89.0	89.5	88.7	89.1	89.6	88.5	89.0	88.5	88.6	88.5	83.1	83.2	83.1	91.6	90.7	91.1	**96.4**	**95.2**	**95.8**
3	90.7	89.6	90.1	90.6	89.5	90.0	91.5	90.4	91.0	89.2	89.2	89.2	87.4	86.7	87.0	92.1	91.0	91.5	**96.6**	**95.3**	**95.9**
4	91.6	90.7	91.1	91.5	90.6	90.9	92.0	91.1	91.6	90.2	90.1	90.2	90.6	88.9	89.7	93.0	91.5	92.2	**96.6**	**95.5**	**96.1**
5	91.6	90.5	91.0	91.6	90.4	91.0	92.2	91.2	91.7	89.8	89.5	89.7	90.3	89.6	89.9	92.7	91.5	92.1	**96.5**	**95.3**	**95.9**

labeled pairs (anomaly-anomaly, anomaly-unlabeled) in PRO and creating augmented views of pseudo-labeled anomalies as positive pairs in the contrastive learning process of ContLeaEncod would intensify the inaccuracies of pseudo-labels. This would lead to a poorly structured latent space that fails to effectively distinguish between normal and anomalous samples, ultimately resulting in poor performance, as observed in our experiments. Moreover, results of TABLE V show that D-SAD has a slightly better performance than DevNet and V-DevNet in all the datasets since it projects the input space into a multi-dimensional outer space close to a center point as opposed to DevNet and V-DevNet that transform the input space into a single anomaly score value as the output space. Finally, we can see that the MNSUAD and NSUAD methods, which learn the anomalous behavior from a latent space pre-processed by an auto-encoder architecture instead of learning it directly from the input samples, have the best performance compared to the other methods. The pre-processing in NSUAD is done by a vanilla auto-encoder while MSUAD performs the pre-processing through an MAE that as discussed earlier leads to learning a more generalizable feature representation. Our experiments confirm this intuition as the results show that on average, MNSUAD has a 1.7%, 3.7%, 4.3% better F1-score than NSUAD in the ITU dataset, MEC dataset, and IMS dataset, respectively. It is important to note that for lower contamination rates (and consequently fewer pseudo-labeled anomalies in the dataset), the difference between the performance of MNSUAD and NSUAD is more significant compared to when the contamination rate is higher. This likely stems from the MAE in MNSUAD learning more generalizable features than the vanilla Autoencoder in NSUAD, allowing MNSUAD to perform well even with few labeled anomalies, while NSUAD requires more to reach peak performance.

E. Ablation Study of the Loss Function in MNSUAD

In this subsection, we conduct an ablation study to evaluate the impact of different loss functions on the performance of the student model in our MNSUAD approach. Specifically, we compare the original loss function $L(x_t)$ (defined in Section IV.C) with two alternative loss functions:

a) Alternative $L_1(x_t)$: This loss function trains the student model exclusively on the cleaned dataset X_c. It retains only the normal component of the original loss function $L(x_t)$ and is defined as:

$$L_1(x_t) = \alpha_1 \times L_{MAE} + |\frac{\Phi(x'(x_t, m)) - \mu_R}{\sigma_R}|, \text{ for } x_t \in X_c \tag{33}$$

b) Alternative $L_2(x_t)$: In this alternative, the reconstruction error of the MAE is removed from the original loss function. The student model is trained on the combined dataset $X_{\text{new}} = \{X_c, \tilde{X}_{\text{anom}}\}$ using the following formulation:

$$L_2(x_t) = \begin{cases} |\frac{\Phi(x'(x_t,m)) - \mu_R}{\sigma_R}|, \text{ if } x_t \in X_c \\ \\ max(0, a - \frac{\Phi(x'(x_t,m)) - \mu_R}{\sigma_R}), \text{ if } x_t \in \tilde{X}_{anom} \end{cases} \tag{34}$$

The performance of these two alternative loss functions is reported in TABLE VI. The student model trained with $L_1(x_t)$ exhibits significantly lower performance compared to the original loss function. Furthermore, its performance deteriorates as the contamination rate (δ) increases, as it fails to leverage the extracted pseudo-anomalies that enhance the model's ability to generalize. Similarly, $L_2(x_t)$ consistently underperforms the original loss function, highlighting the critical role of the reconstruction error term in the loss function. These findings emphasize the necessity of the reconstruction error term and the utilization of pseudo-anomalies in the original loss function to achieve superior performance in MNSUAD.

TABLE VI
Anomaly-detection results of two different alternative loss functions for MNSUAD, trained on X_{new} with the contamination percentage (δ) varying from 0% to 5%

δ	ITU Dataset						MEC Dataset						IMS Dataset					
	L1			L2			L1			L2			L1			L2		
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1
1	80.3	81.1	80.7	81.4	80.6	81.0	81.6	80.9	81.2	82.3	81.8	82.0	86.7	86.2	86.4	87.4	87.0	87.2
2	78.7	76.3	77.5	82.9	82.1	82.6	79.2	79.6	79.4	84.9	83.7	84.3	82.9	81.5	82.2	88.9	87.6	88.2
3	77.0	75.4	76.2	84.0	83.3	83.6	78.5	78.8	78.6	89.1	87.3	88.2	81.6	80.3	80.9	90.3	89.4	89.8
4	75.2	74.8	75.0	85.6	84.7	85.1	74.1	73.7	73.9	90.8	90.0	90.4	78.4	78.5	78.4	92.1	90.8	91.4
5	74.9	74.8	74.8	85.5	84.7	85.0	74.0	72.3	73.1	91.4	90.7	91.1	78.1	75.6	76.8	91.9	90.5	91.2

F. Comparison with Supervised Methods in terms of Generalization Capability

As we mentioned, the main drawback of supervised anomaly detection methods is that they require abundant labeled faulty data to achieve good performance. However, such data is a scarce resource. Additionally, obtaining labeled data requires domain experts to annotate numerous logs of anomalous scenarios, which is both time-consuming and labor-intensive. In this section, through a set of experiments, we illustrate another important disadvantage of supervised methods, which is the problem that they are restricted to the specific failure scenarios that exist in their training data and do not generalize to failure scenarios unseen during training. However, in traditional unsupervised anomaly detection methods, where training is only performed on normal samples, the model is naturally not specialized for some specific failure cases and can detect any anomalous behavior that deviates from the learned normal patterns. Our goal in this section is to show that unlike supervised methods, our proposed weakly-supervised student model for anomaly detection (that takes advantage of a few pseudo-labeled anomaly samples) is generalizable to fault scenarios unseen during training and has a similar behavior to traditional unsupervised approaches in this regard.

To compare the generalization capability of our proposed method with supervised learning approaches, we have conducted the following experiments on the ITU dataset described in Section VI.A. Let X_U denote the contaminated unlabeled training dataset (from the ITU dataset) that has 3870 normal samples, 50 samples of node down (ND) failure, 50 samples of interface down (ID) failure, 50 samples of packet loss (PL) failure, and 50 samples of packet delay (PD) failure. So, X_U has a total of 200 failure samples, and its contamination rate is around 5%. On the other hand, let X_L denote the fully labeled training dataset that includes 3870 normal samples, 54 samples of ND failure, 233 samples of ID failure, 762 samples of PL failure, and 763 samples of PD failure. We define $X_L^{(ND-\nu)}$ as a subset of dataset X_L, where ν percent of ND failure samples are removed from X_L. For example, $X_L^{(ND-50)}$ would only have 27 samples of ND failure, and $X_L^{(ND-100)}$ would have no samples of ND failure. $X_L^{(ID-\nu)}$, $X_L^{(PL-\nu)}$, and $X_L^{(PD-\nu)}$ are defined in a similar way. Moreover, $X_U^{(ND-100)}$, $X_U^{(ID-100)}$, $X_U^{(PL-100)}$, and $X_U^{(PD-100)}$ would be the same as X_U but without any samples of ND, ID, PL, and PD failure, respectively. For each of these training datasets, we create a corresponding test dataset that only consists of anomalous samples of the missing failure scenario. For example, the test data for the training datasets $X_L^{(ND-\nu)}$ and $X_U^{(ND-100)}$ would be denoted as $T^{(ND)}$ that only consists of samples of ND failure. In our experiments, $T^{(ND)}$ has 100 samples of ND fault samples, $T^{(ID)}$ has 100 samples of ID fault samples, $T^{(PL)}$ has 100 samples of PL fault samples, and $T^{(PD)}$ has 100 samples of PD fault samples. We created the test datasets in this way because we are interested to see whether different methods are capable of detecting failure scenarios for which only very few samples (or even no samples) of them exist in the training data.

We trained Random Forest (RF) models as a supervised method on different subsets of the labeled dataset X_L (e.g., $X_L^{(ND-\nu)}$, $X_L^{(ID-\nu)}$, etc.) for different values of ν. Here, the RF model performs anomaly detection as a binary classification problem, similar to [27]. In Fig. 3(a), we have reported the obtained Recall score (the percentage of correctly detected anomalies to the total number of anomalies) of each RF model on its corresponding test dataset when the value of ν changes from 0 to 100. In Fig. 3(b), we have reported the Recall score of our proposed MNSUAD method when it is trained on different subsets of X_U. It is important to clarify that the performance of an RF model/MNSUAD trained on $X_L^{(ND-\nu)}$/$X_U^{(ND-100)}$ is reported on its corresponding test dataset $T^{(ND)}$ (this is true for other failure scenarios, too).

Fig. 3(a) shows that as we decrease the number of samples of a specific scenario in the labeled training data (increasing the value of ν), the ability of the supervised RF model to detect the samples of that failure case degrades significantly. For example, the Recall score of RF when trained on X_L is 87.0% on the $T^{(PL)}$ test set, but its Recall decreases to 75.6% and 68.8% when it's trained on $X_L^{(PL-90)}$ and $X_L^{(PL-100)}$, respectively. We can observe a similar decrease in the Recall score of RF for the other three failure scenarios, too. However, from Fig. 3(b), we can see that excluding all samples of a particular failure scenario from the unlabeled training dataset X_U has an insignificant effect on the performance of our MNSUAD method (for example, Recall score of MNSUAD on test data $T^{(PL)}$ when it is trained on X_U and $X_U^{(PL-100)}$ is almost the same value). This is due to the fact that the student model of MNSUAD only utilizes the pseudo-labeled samples to boost the performance, and does not need samples from all the considered failure scenarios. These results show that our MNSUAD is much more generalizable to unseen fault cases compared to supervised learning approaches.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3582195

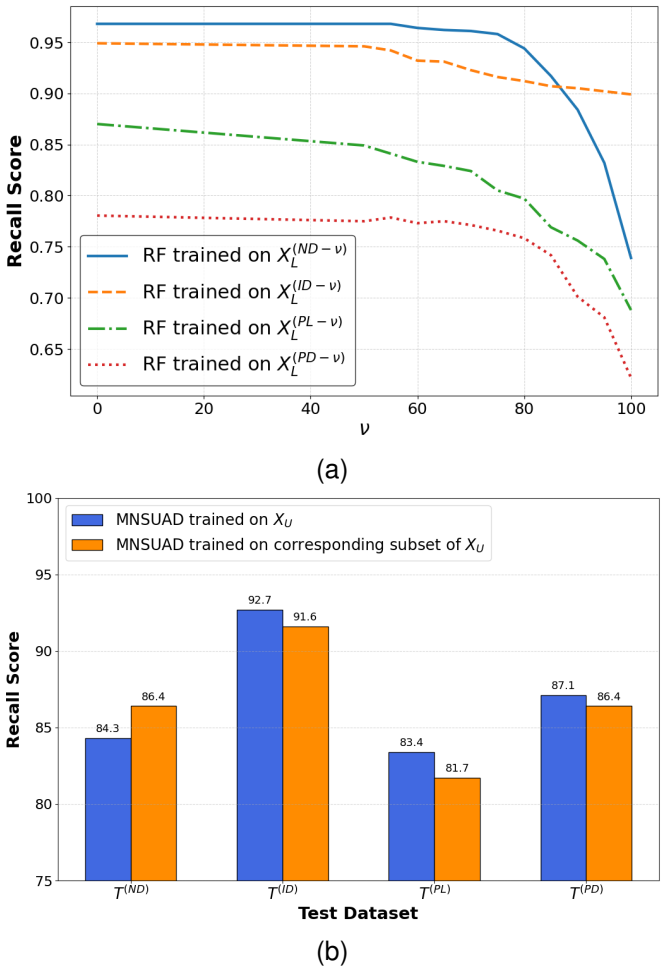

Fig. 3. The Recall scores of RF models trained on different subsets of X_L are shown in (a) when the value of ν changes from 0 to 100. (b) shows the Recall score of our MNSUAD method when it's trained on X_U compared to when its trained on one of the following datasets: $X_U^{(ND-100)}$, $X_U^{(ID-100)}$, $X_U^{(PL-100)}$, or $X_U^{(PD-100)}$.

G. 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 the three datasets are shown in TABLE VII. As discussed, the localization methods use a detection model for their process and localize the anomalies that are detected by that detection model. MNSUAD is the detection model for SHAP, ClusterPer, and MaskPer localization methods (with a training data that is 5% contaminated), and the detection models for the other localization methods are the same as their names according to Section VI.B. Therefore, the number of truly detected anomalies is different for different approaches. According to the results of TABLE VII, our SHAP and ClusterPer methods outperforms Baseline, MSCRED, DAGMM, and LOE in all datasets despite having more detected anomalies. Moreover, the MaskPerm method, which leverages outputs from the detection model across all masking scenarios of the detected anomaly, achieves the highest F1-score on all three datasets. We can also observe from these experiments that Baseline has the lowest average execution time in the datasets. This

TABLE VII
Anomaly localization results of the compared approaches

ITU Dataset					
Method	Pr.	Re.	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
LOE	68.9	69.4	69.1	931	32
SHAP (ours)	87.6	87.5	87.4	951	108
ClusterPer (ours)	92.5	92.5	92.5	951	37
MaskPer (ours)	**96.2**	**96.0**	**96.2**	951	22
MEC Dataset					
Method	Pr.	Re.	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
LOE	70.8	71.1	70.9	166	19
SHAP (ours)	84.5	84.3	84.4	188	97
ClusterPer (ours)	93.2	93.1	93.1	188	22
MaskPer (ours)	**94.4**	**93.0**	**93.7**	188	20
IMS Dataset					
Method	Pr.	Re.	F1	#Anom.	Time(ms)
Baseline	61.4	61.7	61.6	682	**12**
DAGMM	57.3	57.5	57.5	678	**12**
MSCRED	74.6	73.5	74.2	700	15
LOE	66.4	66.7	66.6	671	29
SHAP (ours)	79.3	79.2	79.2	667	114
ClusterPer (ours)	84.3	84.2	84.2	667	32
MaskPer (ours)	**87.7**	**87.8**	**87.8**	667	25

is because its procedure involves a simple analysis of the reconstruction error of different features. On the other hand, SHAP has the highest execution time due to its relatively complex calculations. Since ClusterPer only needs to perform the SHAP calulation if its initial result is "undecided", its average execution time is much lower than the SHAP method. As MaskPer's procedure simply includes calculating the output of the detection model a few times for the detected anomaly sample, its average execution time is even lower than ClusterPer. So, we can conclude that the significant improvement in F1-score obtained by ClusterPer and MaskPer comes at the price of only a slight increase in the execution time.

VII. Conclusion

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 the information provided by the MAE in our detection model can help to achieve a high accuracy in the anomaly localization task. Through a comprehensive experimental analysis on three datasets from 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 20% and 22% in anomaly-detection and localization tasks, respectively.

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] S. Zehra, U. Faseeha, H. J. Syed, F. Samad, A. O. Ibrahim, A. W. Abulfaraj, and W. Nagmeldin, "Machine learning-based anomaly detection in nfv: A comprehensive survey," *Sensors*, vol. 23, no. 11, p. 5340, 2023.

[9] J. Li, X. Qi, J. Li, Z. Su, Y. Su, and L. Liu, "Fault diagnosis in the network function virtualization: A survey, taxonomy and future directions," *IEEE Internet of Things Journal*, 2024.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] Q. Du, Y. He, T. Xie, K. Yin, and J. Qiu, "An approach of collecting performance anomaly dataset for nfv infrastructure," in *Algorithms and Architectures for Parallel Processing: 18th International Conference, ICA3PP 2018, Guangzhou, China, November 15-17, 2018, Proceedings, Part III 18*. Springer, 2018, pp. 59–71.

[18] S. S. Johari, N. Shahriar, M. Tornatore, R. Boutaba, and A. Saleh, "Anomaly detection and localization in nfv systems: an unsupervised learning approach," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–9.

[19] H. U. Adoga and D. P. Pezaros, "Network function virtualization and service function chaining frameworks: A comprehensive review of requirements, objectives, implementations, and open research challenges," *Future Internet*, vol. 14, no. 2, p. 59, 2022.

[20] K. Kaur, V. Mangat, and K. Kumar, "A review on virtualized infrastructure managers with management and orchestration features in nfv architecture," *Computer Networks*, vol. 217, p. 109281, 2022.

[21] S. Mostafavi, V. Hakami, and M. Sanaei, "Quality of service provisioning in network function virtualization: a survey," *Computing*, vol. 103, pp. 917–991, 2021.

[22] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[23] K. Guo, J. Chen, P. Dong, S. Liu, and D. Gao, "Fullsight: A feasible intelligent and collaborative framework for service function chains failure detection," *IEEE Transactions on Network and Service Management*, 2022.

[24] 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.

[25] L. Girish and S. K. Rao, "Anomaly detection in cloud environment using artificial intelligence techniques," *Computing*, pp. 1–14, 2021.

[26] C. Lee, J. Hong, D. Heo, and H. Choi, "Sequential deep learning architectures for anomaly detection in virtual network function chains," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2021, pp. 1163–1168.

[27] 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.

[28] Y. Cheng, H. Yao, Y. Wang, Y. Xiang, and H. Li, "Protecting vnf services with smart online behavior anomaly detection method," *Future Generation Computer Systems*, vol. 95, pp. 265–276, 2019.

[29] D. Kushnir and M. Goldstein, "Causality inference for failures in nfv," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2016, pp. 929–934.

[30] S. Cherrared, S. Imadali, E. Fabre, and G. Gössler, "Sfc self-modeling and active diagnosis," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2515–2530, 2021.

[31] T. Shenkar and L. Wolf, "Anomaly detection for tabular data with internal contrastive learning," in *International conference on learning representations*, 2022.

[32] S. Tao, T. Zhu, H. Wang, and X. Meng, "Semanticmask: a contrastive view design for anomaly detection in tabular data," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024, pp. 2370–2378.

[33] H. Darvishi, D. Ciuonzo, and P. S. Rossi, "Deep recurrent graph convolutional architecture for sensor fault detection, isolation and accommodation in digital twins," *IEEE Sensors Journal*, 2023.

[34] H. Darvishi, D. Ciuonzo, E. R. Eide, and P. S. Rossi, "Sensor-fault detection, isolation and accommodation for digital twins via modular data-driven architecture," *IEEE Sensors Journal*, vol. 21, no. 4, pp. 4827–4838, 2020.

[35] 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.

[36] 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.

[37] F. Michelinakis, J. S. Pujol-Roig, S. Malacarne, M. Xie, T. Dreibholz, S. Majumdar, W. Y. Poe, G. Patounas, C. Guerrero, A. Elmokashfi *et al.*, "Ai anomaly detection for cloudified mobile core architectures," *IEEE Transactions on Network and Service Management*, 2022.

[38] L. Tang, C. Xue, Y. Zhao, and Q. Chen, "Anomaly detection of service function chain based on distributed knowledge distillation framework in cloud-edge industrial internet of things scenarios," *IEEE Internet of Things Journal*, 2023.

[39] A. Chawla, A.-M. Bosneag, and A. Dalgkitsis, "Graph-based interpretable anomaly detection framework for network slice management in beyond 5g networks," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2023, pp. 1–6.

[40] C. Qiu, A. Li, M. Kloft, M. Rudolph, and S. Mandt, "Latent outlier exposure for anomaly detection with contaminated data," in *International conference on machine learning*. PMLR, 2022, pp. 18 153–18 167.

[41] Z. Li, Y. Zhao, Y. Geng, Z. Zhao, H. Wang, W. Chen, H. Jiang, A. Vaidya, L. Su, and D. Pei, "Situation-aware multivariate time series anomaly detection through active learning and contrast vae-based models in large distributed systems," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 9, pp. 2746–2765, 2022.

[42] J. Lu, J. Wang, X. Wei, K. Wu, and G. Liu, "Deep anomaly detection based on variational deviation network," *Future Internet*, vol. 14, no. 3, p. 80, 2022.

[43] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," *arXiv preprint arXiv:1906.02694*, 2019.

[44] G. Pang, C. Shen, H. Jin, and A. v. d. Hengel, "Deep weakly-supervised anomaly detection," *arXiv preprint arXiv:1910.13601*, 2019.

[45] 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.

[46] 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.

[47] 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.

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

[49] Y. Zhang, Z. Guan, H. Qian, L. Xu, H. Liu, Q. Wen, L. Sun, J. Jiang, L. Fan, and M. Ke, "Cloudrca: a root cause analysis framework for cloud computing platforms," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4373–4382.

[50] C. Zhang, Z. Zhou, Y. Zhang, L. Yang, K. He, Q. Wen, and L. Sun, "Netrca: an effective network fault cause localization algorithm," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 9316–9320.

[51] C.-C. Yen, W. Sun, H. Purmehdi, W. Park, K. R. Deshmukh, N. Thakrar, O. Nassef, and A. Jacobs, "Graph neural network based root cause analysis using multivariate time-series kpis for wireless networks," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–7.

[52] 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.

[53] K. Guo, J. Chen, P. Dong, T. Zou, J. Zhu, X. Huang, S. Liu, and C. Liao, "Dtfl: A digital twin-assisted graph neural network approach for service function chains failure localization," *IEEE Transactions on Cloud Computing*, 2023.

[54] 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.

[55] 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.

[56] 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.

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

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

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

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

[61] 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.

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

[63] 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.

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

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

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

[67] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*. PMLR, 2018, pp. 4393–4402.

[68] D. Bahri, H. Jiang, Y. Tay, and D. Metzler, "Scarf: Self-supervised contrastive learning using random feature corruption," *arXiv preprint arXiv:2106.15147*, 2021.

Seyed Soheil Johari is a graduate researcher at the David R. Cheriton School of Computer Science, University of Waterloo. He received his B.Sc. in electrical engineering from Sharif University of Technology in 2020. He was a recipient of the best student paper award at IEEE/IFIP NOMS 2022. His research focuses on the application of machine learning techniques for data-driven management and orchestration of 5G network slices.

Nashid Shahriar is an assistant professor in the Department of Computer Science at the University of Regina. He received his PhD from the School of Computer Science, University of Waterloo in 2020. He was a recipient of 2020 PhD Alumni Gold Medal, 2021 Mathematics Doctoral prize, Ontario Graduate Scholarship, President's Graduate Scholarship, and David R. Cheriton Graduate Scholarship with the University of Waterloo.

Massimo Tornatore (Fellow, IEEE) is a Professor at Politecnico di Milano, Italy. He has also held a appointments as Adjunct Professor at University of California, Davis, USA and as visiting professor at University of Waterloo, Canada. His research interests include performance evaluation and design of communication networks (with an emphasis on optical networking), and machine learning application for network management. He co-authored more than 500 conference and journal papers (with 22 best-paper awards) and the recent Springer "Handbook of Optical Networks". He is member of the Editorial Board of IEEE Communication Surveys and Tutorials, IEEE Transactions on Network and Service Management and IEEE Transactions on Networking, among others.

Raouf Boutaba (Fellow, IEEE) is currently a University Chair Professor and the Director of the David R. Cheriton School of Computer science at the University of Waterloo (Canada). He also holds an INRIA International Chair in France. He is the founding Editor-in-Chief of the IEEE Transactions on Network and Service Management (2007- 2010). He is a fellow of the IEEE, the Engineering Institute of Canada, the Canadian Academy of Engineering, and the Royal Society of Canada. His research interests include resource and service management in networks and distributed systems.

Aladdin Saleh received the Ph.D. degree in electrical and electronic engineering and the M.B.A. degree in international management from the University of London, U.K. He is currently an Adjunct Professor with the Cheriton School of Computer Science, University of Waterloo. He is currently priming research and innovation activities with Rogers Communications, among them the joint research partnership with the University of Waterloo on 5G and emerging technologies.