XAInomaly: Explainable and Interpretable Deep Contractive Autoencoder for O-RAN Traffic Anomaly Detection

Osman Tugay Basaran*, Falko Dressler

School of Electrical Engineering and Computer Science, TU Berlin, Germany

Abstract

Generative Artificial Intelligence (AI) techniques have become integral part in advancing next generation wireless communication systems by enabling sophisticated data modeling and feature extraction for enhanced network performance. In the realm of open radio access networks (O-RAN), characterized by their disaggregated architecture and heterogeneous components from multiple vendors, the deployment of generative models offers significant advantages for network management such as traffic analysis, traffic forecasting and anomaly detection. However, the complex and dynamic nature of O-RAN introduces challenges that necessitate not only accurate detection mechanisms but also reduced complexity, scalability, and most importantly interpretability to facilitate effective network management. In this study, we introduce the XAInomaly framework, an explainable and interpretable Semi-supervised (SS) Deep Contractive Autoencoder (DeepCAE) design for anomaly detection in O-RAN. Our approach leverages the generative modeling capabilities of our SS-DeepCAE model to learn compressed, robust representations of normal network behavior, which captures essential features, enabling the identification of deviations indicative of anomalies. To address the black-box nature of deep learning models, we propose reactive Explainable AI (XAI) technique called fastshap-C, which is providing transparency into the model's decision-making process and highlighting the features contributing to anomaly detection.

Keywords: Explainable and Trustworthy AI, Generative AI, O-RAN, Autoencoder, Anomaly Detection, Network Management

1. Introduction

The rapid proliferation of wireless devices and the exponential growth in data demand have necessitated continuous innovation in cellular network architectures. Traditional Radio Access Networks (RAN) have historically been deployed as monolithic systems, where hardware and software components are tightly integrated and sourced from single vendors. While this approach has ensured reliability and performance, it has also led to significant limitations in terms of scalability, flexibility, and cost-effectiveness. The proprietary nature of traditional RAN hinders interoperability and slows down the adoption of new technologies, making it challenging to meet the evolving requirements of modern wireless communications.

To overcome these challenges, the industry has been transitioning towards O-RAN architectures. O-RAN introduces a paradigm shift by disaggregating the traditional RAN components into modular, interoperable units with open interfaces [1, 2]. This disaggregation allows network operators to mix and match components from different vendors, fostering a competitive ecosystem that drives innovation and reduces costs. O-RAN architectures are also characterized by their support for virtualization and software-defined networking (SDN), enabling dynamic resource allocation and more efficient network management. The complexity and heterogeneity introduced by O-RAN architectures, however, present new challenges in network operation and maintenance. The integration of multi-vendor components and the dynamic reconfiguration of network elements require advanced analytics and automation to ensure optimal performance. AI and Machine Learning (ML) techniques have emerged as key enablers in this context, offering powerful tools for network optimization, self-organization, and intelligent decision-making across different layers of the network [3].

Generative AI (GenAI) models, in particular, have shown great potential in modeling complex network behaviors and generating synthetic data for various applications. Techniques such as Generative Adversarial Networks (GANs) [4] and Deep Autoencoder (DeepAE) can capture intricate patterns in high-dimensional data, making them suitable for tasks like traffic prediction, anomaly detection, and network simulation [5]. By leveraging generative models, network operators can enhance the accuracy of predictive analytics and improve the robustness of network management strategies. In 5G/6G networks as well as O-RAN scenarios [6], AI and generative models can be

*Corresponding author

Email addresses: `basaran@ccs-labs.org` (Osman Tugay Basaran), `dressler@ccs-labs.org` (Falko Dressler) deployed across multiple layers, including:

- *Physical Layer:* Enhancing signal processing algorithms, adaptive beamforming, and channel estimation.
- *MAC and Network Layers:* Optimizing scheduling, resource allocation, and interference management.
- *Application and Service Layers:* Personalizing user experiences, content caching, and service quality prediction.
- *Management and Orchestration Layers:* Automating network configuration, fault detection, and performance optimization.

Anomalies in O-RAN traffic can stem from various sources, including misconfigurations [7], user equipment (UE) failures, software bugs, and security breaches such as cyber-attacks. Early and accurate detection of these anomalies is crucial to prevent service degradation, ensure user satisfaction, and maintain the reliability of critical communications, especially in the context of 5G+/6G networks that support mission-critical applications. Traditional supervised learning approaches for anomaly detection rely heavily on large amounts of labeled data, which is often impractical in real-world network environments due to the rarity and unpredictability of anomalies. Moreover, the manual labeling of network traffic data is labor-intensive and prone to human error. Therefore, there is a pressing need for models that can effectively detect anomalies with minimal reliance on labeled datasets.

Semi-supervised learning models, particularly DeepAE, offer a compelling solution to this challenge. However, primary issue of standard DeepAE is focusing solely on reconstructing input data [8] is their tendency to overfit to the training data, capturing both the underlying patterns and the noise present in the data. Since their objective is purely to minimize reconstruction error, they may not learn robust or meaningful feature representations that generalize well to new, unseen data. This lack of generalization is particularly problematic in anomaly detection tasks within O-RAN environments, where the data is high-dimensional, dynamic, and contains subtle anomalies. Also, DeepAEs are often complex models with large number of layers and parameters, which makes them computationally intensive. This complexity translates to longer training times, higher memory requirements, and increased inference latency, which are challenging for real-time, resource-constrained O-RAN environments in real life deployments. Training these DeepAEs on high-dimensional O-RAN data involves complex optimization, requiring a large amount of labeled data to prevent overfitting and extensive computational resources.

Considering the aforementioned challenges, we designed `SS-DeepCAE` architecture in this study. DeepCAEs address these drawbacks by introducing a contractive penalty into the loss function. This penalty encourages the model to learn smooth and robust feature representations by penalizing large sensitivities of the encoder activations with respect to the input. DeepCAEs tend to have fewer parameters because they do not require complex mechanisms to counter overfitting, such as dropout layers or heavy regularization. This reduces the model's computational demand, making it more scalable and deployable within resource-constrained O-RAN systems. Besides, DeepCAEs are better suited for real-time applications due to their reduced sensitivity to noise and more efficient learning of feature representations. This adaptability is crucial for O-RAN, where anomaly detection may require fast, on-the-fly assessments to maintain network stability and performance. The lack of existing literature on the use of DCAEs for O-RAN traffic anomaly detection underscores the novelty of this approach. By filling this gap, we aim to provide a robust framework that leverages the strengths of DCAEs to address the unique challenges posed by O-RAN networks.

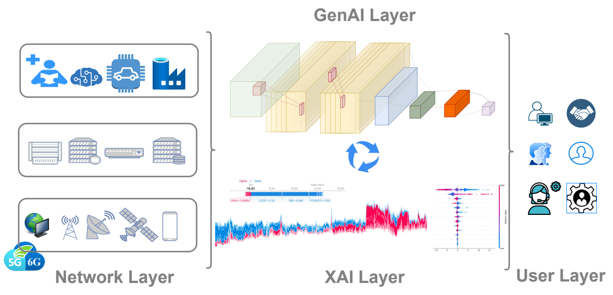


Figure 1: GenAI and XAI interaction between user and network layer

While autoencoders are powerful, they are often criticized for being "black boxes" due to their opaque internal workings [9]. In next generation wireless networks such as 6G [10], this lack of transparency [11] poses significant risks such as unoptimized network management, poor maintenance, biased monitoring, unfair or suboptimal decisions. To address these concerns, integrating XAI [12] techniques into the anomaly detection framework becomes essential [13]. As shown in the Figure 1, XAI aims to make AI models more transparent by providing understandable explanations of their decisions. However, integrating XAI design into O-RAN must be done with understanding performance demands of use cases particularly for Ultra-Reliable Low-Latency Communications (URLLC), impose strict constraints on any deployed solutions. Domain-aware XAI designs can optimize the trade-off between explainability and computational efficiency by focusing explanations on the most relevant features and events within the O-RAN context. In this direction, we introduced novel `fastSHAP-C` XAI algorithm in our previous work [14] for xURLLC use case in O-RAN. However, in this study, we used the DeepAE-based autoencoder that has complexity, scalability and generalizability issues that we mentioned above.

Therefore, there is a clear need for an emerging anomaly detection solution that is performance friendly, effective and trustworthy. In this work, our proposed XAInomaly framework addresses these requirements by combining a novel `SS-DeepCAE` model with `fastSHAP-C` XAI algorithm tailored for the dynamic O-RAN traffic environment.

1.1. Our Contributions

The core outcomes of our study are summarized as new contributions ("C") and new findings ("F") as follows:

C1. We propose the XAInomaly framework, which, for the first time, unites GenAI and XAI design in O-RAN. Our platform provides an interoperable `SS-DeepCAE` model for the O-RAN traffic anomalies.

C2. We introduce novel `fastSHAP-C` XAI method that obtain real-time SHAP values regarding O-RAN intelligence orchestration operations.

C3. We are the first to implement scalable, resource-efficient `SS-DeepCAE` in O-RAN traffic anomaly detection scenario.

F1. The ability of `SS-DeepCAE` to achieve high UAR with minimal labeled data makes it highly scalable for O-RAN environments, where labeled anomalies are rare and costly to obtain.

F2. Our results show that `fastSHAP-C` provides 34% advance over its competitors in terms of runtime performance.

The rest of the paper is organized as follows: In Section 2 building blocks of O-RAN architecture, existing GenAI and XAI implementations are presented (see Table 1). Section 3 describes the proposed `SS-DeepCAE` model in detail, including semi-supervised learning approach and architecture. Section 4 describes proposed XAInomaly framework with selected XAI interpretation methodology, proposed `fastSHAP-C` algorithm and XAInomaly integration on O-RAN architecture. Section 5 includes datasets, performance metrics, experiments with baseline models, and hyper-parameter tuning stages. Section 6 consists of performance results of the `SS-DeepCAE` anomaly detection model and proposed novel XAI algorithm `fastSHAP-C`. Finally, Section 7 concludes the paper and discusses potential directions for future work with challenges. Table 2 contains important acronyms and definitions used throughout this document.

2. Background

This section provides a detailed overview of the key components of the O-RAN architecture, GenAI and XAI literature on O-RAN.

2.1. Principles of O-RAN

O-RAN architecture disaggregates traditional RAN hardware into modular components with standardized interfaces. O-CU is responsible for the higher layers of the protocol stack, specifically the Packet Data Convergence Protocol (PDCP) and Service Data Adaptation Protocol (SDAP) layers. O-RAN Central Unit (O-CU) can be further divided into O-CU-CP (Control Plane) and O-CU-UP (User Plane) for enhanced scalability and flexibility. O-RAN Distributed Unit (O-DU) manages the Radio Link Control (RLC), Medium Access Control (MAC), and parts of the Physical (PHY) layer [22]. It allocates radio resources to UEs based on scheduling algorithms and handles Hybrid Automatic Repeat reQuest (HARQ) for error correction. O-DU is typically located closer to the radio units to meet stringent latency requirements. This allows to perform time-sensitive computations required for real-time operations of radio communications. O-RAN Radio Unit (O-RU) encompasses the lower PHY layer and the Radio Frequency (RF) components [23]. O-RU converts digital signals to analog for transmission over the air and vice versa. It also implements antenna array processing for signal directionality for beamforming.

O-RU interfaces with the O-DU via the Open Fronthaul (OFH) interface, which is standardized to allow interoperability between different vendors' equipment. 3rd Generation Partnership Project (3GPP) and O-RAN Alliance's standardized interfaces are a cornerstone of the O-RAN architecture, ensuring interoperability and flexibility. E2 interface connects the Near-Real-Time RAN Intelligent Controller (Near-RT RIC) with the RAN components (O-CU and O-DU), enabling real-time control and data exchange [24]. A1 interface links the Non-Real-Time RAN Intelligent Controller (Non-RT RIC) with the Near-Real-Time RAN Intelligent Controller (Near-RT RIC), facilitating policy management and model updates [25]. O1 interface connects the Service Management and Orchestration (SMO) with the RAN components for management and orchestration tasks [26]. OFH interface connects the O-DU with the O-RU, supporting high-bandwidth, low-latency data transfer.

SMO framework is a central element in the O-RAN architecture responsible for the overall management and orchestration of network services. It provides end-to-end service lifecycle management, including provisioning, configuration, optimization, and assurance of network functions. It allocates and optimizes physical and virtual resources across the network. Also manages network slices to support diverse service requirements with different performance and resource needs. Hosts AI/ML models and analytics that support intelligent decision-making across the network then implement policies for network operations, security, and compliance.

Non-RT RIC operates within the SMO framework and provides non-real-time control and optimization of RAN functions [27]. It leverages AI/ML algorithms to perform tasks that do not require immediate responsiveness, typically with latency requirements greater than one second. It generates and updates policies for RAN optimization, which are communicated to the Near-RT RIC via the A1 interface. Performs extensive data collection and analysis to understand network performance afterwards it develops and refines AI/ML models based on long-term data analysis. Near-RT RIC is a critical component responsible for real-time and near-real-time control of RAN elements, with latency requirements typically between 10 ms and 1 s. It enables dynamic optimization of RAN resources to meet

Article	Learning Category	Learning Model	Use Case	Year	Data Availability	xApps	Explainability
Fiandrino et al. [15]	Supervised L.	LSTM	Traffic Management	2022	χ	χ	✓
Mahrez et al. [16]	Supervised L.	RFO	Traffic Anomalies	2023	✓	✓	χ
Alves et al. [17]	Supervised L.	MLP	Traffic Anomalies	2023	✓	χ	χ
Fiandrino et al. [18]	Supervised L.	DRL	Resource Allocation	2023	χ	✓	✓
Khan et al. [19]	Supervised L.	DRL	Resource Management	2024	χ	χ	✓
Tassie et al. [20]	Supervised L.	CNN	Traffic Classification	2024	χ	✓	✓
Basaran et al. [21]	Semi-Supervised L.	DeepAE	Traffic Anomalies	2023	✓	✓	χ
Our Work	Semi-Supervised L.	DeepCAE	Traffic Anomalies	2024	✓	✓	✓

Table 1: Literature review on different O-RAN use cases with generative and explainable AI

Acronym	Definition	Acronym	Definition
3GPP	3rd Generation Partnership Project	O-RU	O-RAN Radio Unit
5G	Fifth Generation Mobile Network	PDCP	Packet Data Convergence Protocol
6G	Sixth Generation Mobile Network	PHY	Physical
AI	Artificial Intelligence	QoE	Quality of Experience
AD xApp	Anomaly Detection xApp	QoS	Quality of Service
CNN	Convolutional Neural Network	RAN	Radio Access Networks
$\mathcal{CS}$	Confidence Score	RF	Radio Frequency
DL	Deep Learning	rApps	RAN Applications
DeepAE	Deep Autoencoder	RFO	Random Forest
DeepCAE	Deep Contractive Autoencoder	ReLU	Rectified Linear Unit
DRL	Deep Reinforcement Learning	RLC	Radio Link Control
$\mathcal{EM}$	Error Metric	SDAP	Service Data Adaptation Protocol
GenAI	Generative AI	SGD	Stochastic Gradient Descent
GANs	Generative Adversarial Networks	SHAP	SHapley Additive exPlanations
HARQ	Hybrid Automatic Repeat reQuest	SLA-VAE	Semi-supervised VAE
LSTM	Long Short-term Memory	SMO	Service Management and Orchestration
O-RAN	Open Radio Access Networks	SS	Semi-supervised
MAC	Medium Access Control	SS-DeepCAE	Semi-supervised Deep Contractive Autoen.
MSE	Mean Squared Error	t-SNE	t-Distributed Stochastic Neighbor Embedding
ML	Machine Learning	TS xApp	Traffic Steering xApp
Near-RT RIC	Near-Real-Time RAN Intelligent Controller	UE	User Equipment
Non-RT RIC	Non-Real-Time RAN Intelligent Controller	URLLC	Ultra-Reliable Low-Latency Communications
O-CU	O-RAN Central Unit	VAEs	Variational Autoencoders
O-DU	O-RAN Distributed Unit	XAI	Explainable AI
OFH	Open Fronthaul	xApps	eXtended applications

Table 2: List of important acronyms and their definitions

immediate performance objectives. It adjusts parameters such as scheduling, power control, and beamforming in time sensitive manner.

RAN applications are software modules that provide specialized functionalities within the O-RAN architecture. They are categorized into RAN applications (rApps) and eXtended applications (xApps) based on their operational time scales and hosting environments. rApps are RAN applications that run on the Non-RT RIC within the SMO framework. They perform non-real-time functions such as network planning with long-term optimization of network topology and configurations, forecasting network behavior and traffic patterns, automated adjustment of network settings based on policies and analytics. As shown in Figure 2, our proposed xApps hosted on the Near-RT RIC, designed for real-time and near-real-time operations [28]. They are useful for dynamic distribution of traffic to optimize network utilization, immediate adjustments to

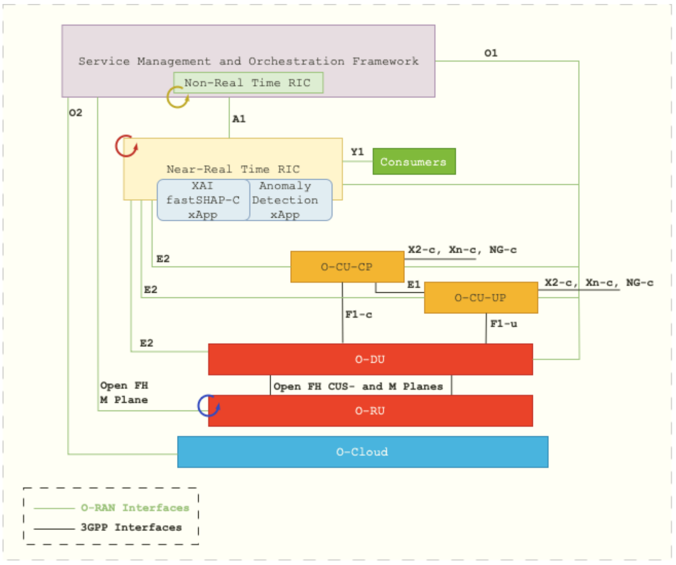


Figure 2: O-RAN reference architecture with XAInomaly

meet Quality of Service (QoS) requirements for different services.

2.2. GenAI and XAI Literature on O-RAN

GenAI studies on wireless communication systems are quite common in the literature [29–31]. GenAI models, such as GANs, Variational Autoencoders (VAEs), and DeepAEs, have demonstrated significant capabilities in modeling complex data distributions and generating synthetic data. Similarly, GenAI studies on O-RAN are quite numerous however, XAI research is still scarce, but gaining momentum recently. In Table 1, we gathered and summarized the current GenAI and XAI research on O-RAN. Studies focused on resource allocation, traffic anomalies and management use cases.

In resource allocation studies, Khan *et al.* [19] explores the application of XAI within distinct 6G use cases, with a focus on a Deep Reinforcement Learning (DRL) solution designed for vehicular network scenarios. Specifically, the authors utilized SHAP values to analyze feature contributions. However, a major challenge lies in the high complexity and dimensionality of DRL models, which often involve many layers and parameters, making interpretation difficult. This complexity can result in explanations that either oversimplify and miss important details or are too intricate for end-users to easily grasp. Moreover, in a vehicular use case, the XAI model must provide real-time evaluations, a requirement that conventional SHAP algorithms struggle to meet in practical settings due to computational demands. In [18], the authors present EXPLORA, an innovative framework designed to improve the explainability of AI/ML models within Open RAN systems. This study highlights the critical need for transparency and interpretability in decentralized networks like Open RAN, where complex decisions are often made at the network edge.

Different models have been designed specifically for traffic management studies. When the studies are examined, it is seen that most of the models are supervised learning based solutions such as Random Forest (RF), Multilayer Perceptron (MLP), Convolutional Neural Network (CNN) [16, 17, 20].However, considering heterogeneous components of O-RAN, especially the specific interfaces and data flows, extensive data labeling is quite costly and challenging. Fiandrino *et al.* [15] further investigate the application of XAI and robustness in 6G networks, acknowledging the challenges that remain, particularly in achieving a balance between explainability and robustness while providing explanations that are both precise and accessible to diverse stakeholders, such as network operators and end-users. A use case involving a 4G network examines the LRP and SHAP algorithms, with shared results on execution time and resource usage. The findings reveal that CPU usage poses a significant limitation, indicating the need for the development of more advanced, domain-specific XAI algorithms. This motivated our research into creating XAI solutions capable of managing the real-time data demands of 6G xURLLC communications.

3. Semi-supervised DeepCAE Design for O-RAN Anomaly Detection

3.1. Overview of Our Semi-Supervised L. Approach

In the context of anomaly detection within O-RAN networks, semi-supervised learning is particularly advantageous due to the scarcity of labeled anomalous data. Semi-supervised learning leverages both labeled and unlabeled data during training, improving the model's ability to generalize and detect unseen anomalies [32]. In our approach, the majority of the training data consists of unlabeled normal traffic patterns, while a small subset is labeled to guide the learning process. Given the labeled dataset

$$\mathcal{D}_l = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_l} \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}^n$ represents the input features, and $y_i \in \{0, 1\}$ is the label indicating normal (0) or anomalous (1) traffic. Given also the unlabeled dataset

$$\mathcal{D}_u = \{\mathbf{x}_i\}_{i=N_l+1}^{N_l+N_u} \tag{2}$$

where $N_u \gg N_l$, reflecting the abundance of unlabeled data. The total dataset is

$$\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u \tag{3}$$

with total samples $N = N_l + N_u$.

Our objective is to train a model that can

① Reconstruct normal traffic patterns accurately.
② Learn robust and contractive latent representations to detect anomalies.
③ Incorporate very few label information to improve anomaly detection performance.
④ Generalize well to detect unseen anomalies.

3.2. Proposed DeepCAE Architecture

Our DeepCAE introduces a regularization term that penalizes the sensitivity of the encoder's output to small perturbations in the input [33]. This encourages the model to learn robust representations that capture essential features. Architecture consists of two main components

1. *Encoder (E)*: Transforms the input data $\mathbf{x} \in \mathbb{R}^n$ into a latent representation $\mathbf{z} \in \mathbb{R}^m$, where $m < n$.
2. *Decoder (D)*: Reconstructs the input data from the latent representation, producing $\hat{\mathbf{x}} \in \mathbb{R}^n$.

DeepCAE is trained to minimize the reconstruction error between $\mathbf{x}$ and $\hat{\mathbf{x}}$, effectively learning the underlying structure of the data. The encoder E maps input data $\mathbf{x}$ to a latent representation $\mathbf{z}$

$$\mathbf{z} = E(\mathbf{x}; \theta_e) = f_e(\mathbf{W}_e \mathbf{x} + \mathbf{b}_e) \tag{4}$$

where $\theta_e = \{\mathbf{W}_e, \mathbf{b}_e\}$ represents the encoder parameters. In layer-wise form, the encoder can be expressed as

$$\mathbf{h}^{(l)} = f^{(l)}(\mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \tag{5}$$

for $l = 1, 2, \ldots, L_e$, where: $\mathbf{h}^{(0)} = \mathbf{x}$, $\mathbf{h}^{(L_e)} = \mathbf{z}$, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are weights and biases for layer l, $f^{(l)}(\cdot)$ is the activation function. [1] Decoder D reconstructs the input from $\mathbf{z}$:

$$\hat{\mathbf{x}} = D(\mathbf{z}; \theta_d) = f_d(\mathbf{W}_d \mathbf{z} + \mathbf{b}_d) \tag{6}$$

where $\theta_d = \{\mathbf{W}_d, \mathbf{b}_d\}$ represents the decoder parameters. Similarly, in layer-wise form:

$$\mathbf{h}^{(l)} = f^{(l)}(\mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \tag{7}$$

for $l = L_e + 1, L_e + 2, \ldots, L$, with: $\mathbf{h}^{(L_e)} = \mathbf{z}$, $\mathbf{h}^{(L)} = \hat{\mathbf{x}}$.

3.2.1. Loss Functions

For all samples (both labeled and unlabeled), we compute the reconstruction loss to train the autoencoder to accurately reconstruct normal input patterns:

$$\ell_{\text{recon}}(\mathbf{x}_i, \hat{\mathbf{x}}_i) = \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 \tag{8}$$

To encourage robustness in the latent representation, we introduce a contractive penalty based on the Frobenius norm of the Jacobian of the encoder activations with respect to the input [34]:

$$\ell_{\text{contractive}}(\mathbf{x}_i) = \lambda_c \left\| \frac{\partial E(\mathbf{x}_i; \theta_e)}{\partial \mathbf{x}_i} \right\|_F^2 \tag{9}$$

where: λ_c is the regularization parameter controlling the strength of the penalty. $\|\cdot\|_F^2$ denotes the squared Frobenius norm. For the deep encoder, the Jacobian $\mathbf{J}_i$ is:

$$\mathbf{J}_i = \frac{\partial \mathbf{z}_i}{\partial \mathbf{x}_i} = \mathbf{W}^{(l)} \text{diag}\left(f'^{(l-1)}(\mathbf{W}^{(l-1)} \mathbf{x}_i + \mathbf{b}^{(l-1)}) \right) \mathbf{W}^{(l-1)} \tag{10}$$

where $f'^{(l-1)}(\cdot)$ is the derivative of the activation function in the first layer. For very few labeled samples, we introduce a cross-entropy loss[2] to incorporate the label information into the training process. We first map the latent representation $\mathbf{z}$ to a prediction $\hat{y}$

$$\ell_{\text{cro}}(y_i, \hat{y}_i) = -[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] \tag{11}$$

Then total loss function combines the reconstruction loss and supervised loss

$$L = \frac{1}{N} \sum_{i=1}^{N} [\ell_{\text{recon}}(\mathbf{x}_i, \hat{\mathbf{x}}_i) + \ell_{\text{contractive}}(\mathbf{x}_i) + \alpha_i \cdot \ell_{\text{cro}}(y_i, \hat{y}_i)] \tag{12}$$

where

$$\alpha_i = \begin{cases} \lambda, & \text{if } i \leq N_l \\ 0, & \text{if } i > N_l \end{cases} \tag{13}$$

and λ is a hyper-parameter controlling the influence of the cross-entropy loss.

3.2.2. Optimization and Batch Training

We optimize the combined parameters $\theta = \{\theta_e, \theta_d\}$ by minimizing the total loss L using gradient-based optimization methods, such as Adam or SGD with momentum. [3] The update rule is:

$$\theta \leftarrow \theta - \eta \nabla_\theta L \tag{14}$$

where η is the learning rate. The gradient of the contractive penalty requires computing the derivative of the Frobenius norm:

$$\nabla_{\theta_e} \ell_{\text{contractive}}(\mathbf{x}_i) = 2\lambda_c \left(\frac{\partial E(\mathbf{x}_i; \theta_e)}{\partial \mathbf{x}_i} \right)^\top \frac{\partial^2 E(\mathbf{x}_i; \theta_e)}{\partial \mathbf{x}_i \partial \theta_e} \tag{15}$$

Efficient computation techniques, automatic differentiation, are employed to compute these gradients. During training, we use mini-batches that contain both labeled and unlabeled samples. Each batch is constructed by sampling B_l labeled and B_u unlabeled samples, ensuring that the model learns from both types of data in each iteration.

[1]Activation function selected as Rectified Linear Unit (ReLU) because ReLU inherently induces sparsity by outputting zero for any negative input, which encourages the model to learn sparse representations. Sparse features are often more effective for distinguishing normal data from anomalies, as they can emphasize essential patterns while suppressing noise. Also, ReLU mitigates the vanishing gradient problem common with saturating activations (e.g., sigmoid or tanh), which allows for deeper networks to learn effectively. In anomaly detection, deeper layers are beneficial for capturing complex patterns and subtle distinctions between normal and anomalous data.

[2]Typically, a sigmoid function is used in the output layer, which can lead to a saturation effect in the loss function, causing it to plateau. This saturation hampers gradient-based learning algorithms, limiting their ability to make progress. To mitigate this issue, incorporating a logarithmic term in the objective function counteracts the exponential nature of the sigmoid function, facilitating effective gradient flow. Consequently, binary cross-entropy loss is preferred over Mean Squared Error (MSE) since it incorporates a logarithmic term, unlike MSE [5].

[3]Although we do not discuss here, we also remark that the Adam optimizer is selected [35].

3.2.3. Anomaly Detection During Inference

After training, we use the model to detect anomalies in new data. For an input $\mathbf{x}$, we compute the reconstruction error:

$$\mathrm{RE}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \tag{16}$$

If $\mathrm{RE}(\mathbf{x})$ exceeds a threshold τ, we classify $\mathbf{x}$ as anomalous:

$$\text{Anomaly Indicator} = \begin{cases} 1, & \text{if } \mathrm{RE}(\mathbf{x}) > \tau \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

Anomalies may result in larger norms of the latent representation due to the contractive penalty:

$$\text{Latent Norm} = \|\mathbf{z}\|_2 \tag{18}$$

Therefore, we can define an anomaly score combining reconstruction error and latent norm:

$$\text{Anomaly Score} = \gamma \cdot \mathrm{RE}(\mathbf{x}) + (1 - \gamma) \cdot \|\mathbf{z}\|_2 \tag{19}$$

where $\gamma \in [0, 1]$ balances the contributions. Model classifies $\mathbf{x}$ as anomalous if the Anomaly Score exceeds a threshold. The latent representation $\mathbf{z}$ provides a compressed embedding:

$$m = \dim(\mathbf{z}) < n = \dim(\mathbf{x}) \tag{20}$$

Our contractive penalty ensures that $\mathbf{z}$ is less sensitive to small input variations, enhancing robustness.

4. Proposed Design: XAInomaly Framework

In this section, we introduce our `fastSHAP-C` algorithm, an efficient and interpretable method designed to explain the predictions of our DeepCAE used for anomaly detection in O-RAN networks. The algorithm extends the existing fastSHAP method by incorporating a Confidence Score $\mathcal{CS}$ and an Error Metric $\mathcal{EM}$ to assess the reliability and accuracy of the explanations, which is critical for 5G+/6G O-RAN applications.

4.1. Selecting XAI Interpretation Methods

In our implementation, we have designed `fastSHAP-C` algorithm with Global Model-Agnostic, and Reactive XAI perspective.

4.1.1. Global Model-Agnostic XAI Interpretation

Global interperation provide explanations that encompass the overall behavior of our model across the entire dataset. This reduces computational overhead, allowing for efficient processing of vast amounts of data and ensuring that the XAI system remains performant as the network complexity grows.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be our DeepCAE model, where d is the number of features. The goal of a global explanation is to compute the expected contribution of each feature i to the model's output over the data distribution $\mathcal{D}(x)$

$$\phi_i^{\text{global}} = \mathbb{E}_{x \sim \mathcal{D}(x)} \left[\phi_i(x)\right] \tag{21}$$

where $\phi_i(x)$ is the Shapley value of feature i for input x, representing the contribution of feature i to the prediction at x. In `fastSHAP-C`, we approximate the global Shapley values by training an explainer function $\phi_{\text{fast}}(x; \theta)$ parameterized by θ to predict $\phi_i(x)$ for any input x

$$\phi_i(x) \approx \phi_{\text{fast},i}(x; \theta) \tag{22}$$

By learning θ over the entire dataset, we ensure that the explainer captures the global behavior of the model. Model-agnostic XAI methods do not rely on the internal structure or parameters of the predictive model. Instead, they use input-output behavior to generate explanations. Our `fastSHAP-C` algorithm treats DeepCAE model $f(x)$ as a black box. The Shapley values are computed based on the model's predictions for different subsets of features without requiring access to the model's internals. The value function $v(S)$ for a subset of features $S \subseteq \{1, 2, \ldots, d\}$ is defined as

$$v(S) = \mathbb{E}_{x' \sim \mathcal{D}(x)} \left[f(x_S \cup x'_{\bar{S}})\right] \tag{23}$$

where

- x_S is the vector containing the values of features in S from x.
- $x'_{\bar{S}}$ is a sample from the background distribution for features not in S.
- $\bar{S}$ denotes the complement of S.

Shapley value $\phi_i(x)$ for feature i is computed as

$$\phi_i(x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(d - |S| - 1)!}{d!} \left(v(S \cup \{i\}) - v(S)\right) \tag{24}$$

This formulation does not depend on the model's architecture, making `fastSHAP-C` model-agnostic.

4.1.2. Reactive XAI Interpretation

Reactive XAI methods generate explanations simultaneously with the model's predictions, providing immediate insights. In the dynamic and time-sensitive context of O-RAN networks, real-time explanations enable prompt understanding and response to anomalies. It supports immediate decision-making processes, such as triggering mitigation strategies or adjusting network parameters. In `fastSHAP-C`, the explainer $\phi_{\text{fast}}(x; \theta)$ is designed to produce Shapley value estimates rapidly for any input x by leveraging the pre-trained parameters θ

$$\phi(x) = \phi_{\text{fast}}(x; \theta) \tag{25}$$

Since ϕ_{fast} is a learned function, computing $\phi(x)$ involves a simple forward pass, allowing for real-time explanations.

Algorithm 1: `fastSHAP-C` Training

```
1  Input: Value function $f_{x,y}$, learning rate $\alpha$
2  Output: fastSHAP explainer $\phi_{fast}(x, y; \theta)$
3  initialize $\phi_{fast}(x, y; \theta)$
4  while not converged do
5      sample $x \sim p(x), y \sim Unif(y), s \sim p(s)$
6      predict $\hat{\phi} \leftarrow \phi_{fast}(x, y; \theta)$
7      if normalize then
8          set $\hat{\phi} \leftarrow \hat{\phi} + d^{-1}(f_{x,y}(1) - f_{x,y}(0) - 1^T \hat{\phi})$
9      calculate
10     $\mathcal{L} \leftarrow (f_{x,y}(s) - f_{x,y}(0) - s^T \hat{\phi})^2$
11     update $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$
12 $\mathcal{CS} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \left| f_{x,y}(s_i) - f_{x,y}(0) - s_i^T \hat{\phi} \right|$
13 $\mathcal{EM} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \left( f_{x,y}(s_i) - s_i^T \hat{\phi} \right)^2$
14 return $\phi_{\text{fast}}(x, y; \theta)$, $\mathcal{CS}$, $\mathcal{EM}$
```

4.2. Proposed fastSHAP-C Algorithm

`fastSHAP-C` algorithm aims to approximate the Shapley values for feature attribution in a computationally efficient manner. Traditional SHAP methods can be computationally intensive, especially for models with high-dimensional input data, as is common in O-RAN traffic analysis. `fastSHAP-C` addresses this challenge by learning a universal explainer that approximates Shapley values for similar data points without needing to optimize separately for each input and incorporating Confidence Score ($\mathcal{CS}$) and Error Metric ($\mathcal{EM}$) to quantify the reliability and accuracy of the explanations. Inputs and steps of `fastSHAP-C` algorithm (cf. Algorithm 1) are as follows:

① *Model* $f_{x,y}$: The value function representing the prediction model (autoencoder) for input x and output y.
② *Learning Rate* α: The step size used in the optimization algorithm.
③ *Data Sample* x: The input data point to be explained.
④ *Background Dataset* X: A set of samples representing the background data distribution.
⑤ *Number of Samples* M: The number of subsets sampled for approximation.

1. *Initialize:*

$$\phi_i(x) = 0 \quad \forall i$$
$$f(x) = f(x)$$
$$\hat{f}(x) = 0$$

2. *Sampling:* For each sample m from 1 to M: Sample a subset $S \subseteq \{1, 2, \ldots, d\}$ uniformly at random.

3. *Marginal Contribution:* For each feature i: Compute the marginal contribution of feature i given the subset S:

$$\Delta f_i(S) = f(S \cup \{i\}) - f(S) \tag{26}$$

Update the Shapley value:

$$\phi_i(x) = \phi_i(x) + \frac{\Delta f_i(S)}{M} \tag{27}$$

4. *Calculate* $\mathcal{CS}$*:*
First compute the reconstructed prediction;

$$\hat{f}(x) = \sum_{i=1}^{d} \phi_i(x) \tag{28}$$

$$\mathcal{CS} = \frac{1}{n} \sum_{k=1}^{n} \left| f(x_k) - \hat{f}(x_k) \right| \tag{29}$$

The $\mathcal{CS}$ quantifies the average absolute discrepancy between the true model output difference and the explainer's approximation over n samples. A lower $\mathcal{CS}$ indicates higher confidence in the explanations.

5. *Calculate* $\mathcal{EM}$*:* Compute the error between the actual and reconstructed predictions:

$$EM = \frac{1}{n} \sum_{k=1}^{n} \left(f(x_k) - \hat{f}(x_k) \right)^2 \tag{30}$$

$\mathcal{EM}$ measures the mean squared error between the true model output and the explainer's approximation. A lower $\mathcal{EM}$ signifies higher accuracy of the explainer.

6. *Output:* Return the Shapley values $\phi_i(x)$, $\mathcal{CS}$, and $\mathcal{EM}$.

The loss function $\mathcal{L}$ measures the discrepancy between the true model output and the approximation provided by the explainer

$$\mathcal{L} = \left(f_{x,y}(s) - f_{x,y}(\mathbf{0}) - s^\top \hat{\phi} \right)^2 \tag{31}$$

where
① $f_{x,y}(s)$: The model output when features in subset s are present.
② $f_{x,y}(\mathbf{0})$: The model output when no features are present (baseline prediction).
③ $s^\top \hat{\phi}$: The linear combination of predicted Shapley values corresponding to the subset s.

Using gradient descent, we update the explainer parameters θ

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L} \tag{32}$$

4.3. XAInomaly Integration on O-RAN

Our XAInomaly framework is integrated into the O-RAN architecture via Anomaly Detection xApp (AD xApp) and Reactive-XAI xApp as seen in the Figure 3. AD xAppis central to the anomaly detection process. It continuously monitors UE data, detects anomalies using our `SS=DeepCAE` model, and identifies UEs exhibiting unusual traffic behavior, degraded performance. Once an anomaly

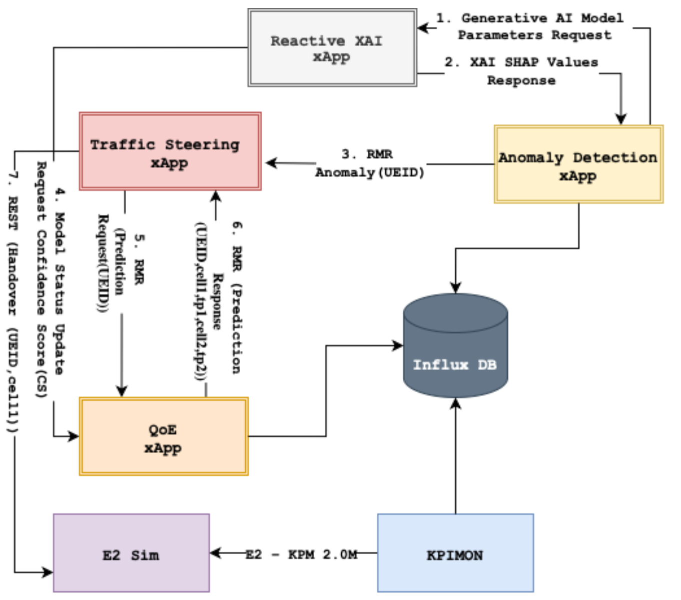

Reactive XAI xApp
1. Generative AI Model Parameters Request
2. XAI SHAP Values Response
Traffic Steering xApp
3. RMR Anomaly (UEID)
Anomaly Detection xApp
4. Model Status Update Request Confidence Score (CS)
5. RMR (Prediction Request(UEID))
6. RMR (Prediction Response (UEID,cell1,tp1,cell2,tp2))
7. REST (Handover (UEID,cell1))
Influx DB
QoE xApp
E2 Sim
E2 - KPM 2.0M
KPIMON

Figure 3: Integration of XAInomaly framework to O-RAN

is detected, the AD xApp logs this information to the InfluxDB database for record-keeping and further analysis. It then sends the anomalous UE information to the Traffic Steering xApp (TS xApp) via Reliable Message Routing (RMR). AD xApp also interacts with the Reactive XAI xApp by responding to model parameter requests and to calculate SHAP values, which enable explainability for the detected anomalies.

TS xApp is responsible for managing and optimizing UE traffic based on detected anomalies. Upon receiving the list of anomalous UEs from the AD xApp, it requests predictions from the Quality of Experience (QoE) Predictor xApp to determine potential throughput for each UE. TS xApp requests $\mathcal{CS}$ for the anomaly detection process from the Reactive XAI xApp. This score, calculated by fastSHAP-C, helps gauge the reliability of the anomaly detection, allowing TS xApp to factor in the confidence level of detected anomalies during handover decisions. Based on the throughput predictions provided by the QoE Predictor xApp, TS xApp decides whether to hand over a UE to a neighboring cell with better performance metrics (higher predicted throughput). TS xApp works collaboratively with both the AD xApp and the QoE xApp to make intelligent traffic management decisions and improve network efficiency.

O-RAN Software Community's E-release of the Near-RT RIC[4] and E2 Agent has been integrated into our XAInomaly framework to manage E2 Service Model (E2SM) functionalities, facilitating communication between the Near-RT RIC and E2 nodes. E2SM-KPM is used to gather and report essential KPIs and measurements from the RAN to the Near-RT RIC. This continuous flow of real-time metrics enables the RIC to maintain a close watch on network performance. Anomaly detection and optimization decisions are then enacted through the E2SM-RC (RAN Control) model, which transmits control policies back to the RAN. By combining E2SM-KPM and E2SM-RC for policy implementation, this setup empowers the Near-RT RIC to make data-driven decisions that optimize network performance in real-time. E2 Simulator provides simulated RAN data through E2 (E2-KPM 2.0M) messages, enabling the testing of the xApps. It supplies critical information on key performance metrics (KPIs) to the xApps.

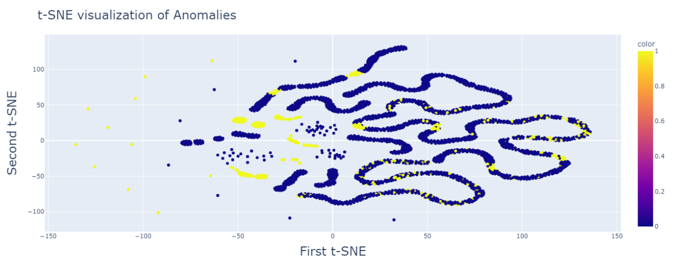


Figure 4: Dimensionality reduction on our high-dimensional unbalanced data (yellow color represent anomalous samples, blue color represents non-anomalous samples)

5. Experimental Analysis

5.1. Dataset

O-RAN dataset[5] was collected from various UE sources, including cars, train passengers, pedestrians, and people waiting. It comprises 10,000 samples that capture important features such as RSRP, RSRQ, RSSINR, Physical Resource Block (PRB) usage, and Throughput. Additionally, it includes contextual features like timestamp, NRCellIdentity, DU-id, and UE-id. The dataset also has a binary label (0 or 1) indicating the presence or absence of an anomaly, with approximately 25% of the samples labeled as anomalies, highlighting an imbalanced dataset. In Figure 4, the dataset's classes are visualized using the t-Distributed Stochastic Neighbor Embedding (t-SNE) technique [36], which reduces high-dimensional data into a two-dimensional view. The figure clearly shows how anomaly samples are scattered and sparsely located, underscoring the need for a robust model to effectively classify this minority class.

5.2. Selected Anomaly Detection Metrics for `SS-DeepCAE`

We employed six standard metrics to assess the efficacy of anomaly detection [37]: Accuracy (ACC), representing the ratio of correctly classified instances; Precision (PR), indicating the ratio of positive instances that are accurately identified as positive; Recall (RE), which reflects the ratio of actual positive instances that are correctly recognized; F1 Score (F_1), defined as the harmonic mean of precision

[4]OSC Near Realtime RIC, `https://wiki.o-ran-sc.org/display/RICP/2022-05-24+Release+E`

[5]Access to the documented data by O-RAN Alliance is provided at `https://github.com/o-ran-sc/ric-app-ad/blob/master/src/ue.csv`

and recall; Area Under Curve (AUC), which gauges the model's ability to differentiate between positive and negative instances and Unweighted Average Recall (UAR) also known as Unweighted Average Accuracy, is the sum of class-wise accuracy (recall) divided by number of classes.

5.3. Selected XAI Metrics for `fastSHAP-C`

To enhance the interpretability and trustworthiness of our `SS-DeepCAE` for anomaly detection in O-RAN networks, we integrate specific XAI metrics into `fastSHAP-C` algorithm. Key XAI metrics incorporated are:

① *Confidence Score $\mathcal{CS}$:* $\mathcal{CS}$ is assessing the reliability and robustness of the explanations generated by `fastSHAP-C`. It measures the consistency between the model's predictions and the explanations provided by the model. A high $\mathcal{CS}$ indicates that the explanations are accurate and align closely with the model's behavior, fostering trust in the model's predictions. Let $f(x)$ denote the model's prediction for input x, and $\phi(x)$ represent the vector of feature importances (Shapley values) provided by `fastSHAP-C` for input x. The reconstructed prediction using the explanations is given by

$$\hat{f}(x) = \sum_{i=1}^{d} \phi_i(x) \tag{33}$$

where d is the number of features. $\mathcal{CS}$ is defined as the average absolute difference between the model's actual prediction and the reconstructed prediction

$$\text{CS} = \frac{1}{n} \sum_{i=1}^{n} \left| f(x_i) - \hat{f}(x_i) \right| \tag{34}$$

where n is the number of samples, x_i is the i-th sample, and $\hat{f}(x_i)$ is the reconstructed prediction for x_i. A lower $\mathcal{CS}$ signifies higher reliability of the explanations, as the reconstructed predictions closely match the actual model outputs.

② *Sensitivity:* It quantifies the degree to which the explanations provided by `fastSHAP-C` are affected by small changes in the input data or model parameters. It assesses the robustness of the model by evaluating how much the explanations vary in response to perturbations in the input features. For input x_i, the sensitivity $\lambda(x_i)$ is defined as

$$\lambda(x_i) = \max_{x_j \in B_\epsilon(x_i)} \frac{\|\phi(x_i) - \phi(x_j)\|_2}{\|x_i - x_j\|_2} \tag{35}$$

where

- $B_\epsilon(x_i)$ is an ϵ-neighborhood around x_i, representing inputs within a small perturbation of x_i.
- $\phi(x_i)$ and $\phi(x_j)$ are the explanations for x_i and x_j, respectively.
- $\|\cdot\|_2$ denotes the Euclidean norm.

A lower sensitivity value indicates that the explanations are stable and robust to small input perturbations, which is essential in dynamic O-RAN environments where network conditions can fluctuate.

③ *Log-odds:* Metric evaluates the importance or relevance of each feature in influencing the model's predictions. It measures the change in the log-odds of the predicted outcome when a particular feature is present or absent, providing insights into the feature's impact on the model's decision-making process. For a feature p, the log-odds is computed as

$$log - odds(p) = -\frac{1}{L} \sum_{i=1}^{L} log \frac{Pr(\hat{y}|x_i^{(p)})}{Pr(\hat{y}|x_i)} \tag{36}$$

where

- L is the total number of samples.
- $x_i^{(p)}$ is the input x_i with feature p removed or altered.
- $\hat{y}_i$ is the model's predicted output for input x_i.
- $\Pr(\hat{y}_i|x_i^{(p)})$ and $\Pr(\hat{y}_i|x_i)$ are the probabilities of the predicted outcome given the modified and original inputs, respectively.

A positive $log - odds$ value indicates that the presence of feature p increases the likelihood of the predicted outcome, whereas a negative value suggests it decreases the likelihood.

5.4. Hyper-parameter Tuning

Hyper-parameter tuning is a critical step in developing effective learning models, particularly for complex architectures like autoencoders. A well-structured tuning process ensures that finding the optimal combination of parameters that yields the best model performance. Robust tuning can significantly improve the performance of the model leading to more accurate anomaly detection and better generalization. Furthermore, optimized hyper-parameters help the model converge faster and more efficiently, saving computational resources for resource-constrained O-RAN networks. It helps prevent overfitting or underfitting by finding hyper-parameters that strike the right balance between model complexity and learning capability. Given the complexity of our problem, systematic method called GridSearchCV [38] is selected in our tuning process. Grid search automates the search for optimal hyper-parameters by exhaustively testing all combinations in a predefined parameter grid. The parameters to be tuned and their respective ranges (parameter space) are seen in Table 3. Selected input size of 20, corresponding to the full set of available features in our dataset.

Considering that each feature may carry significant information relevant to anomaly detection in O-RAN traffic, it was crucial to utilize the complete feature set to capture the complex patterns present in high-dimensional data.

Hyper-param.	Search Space	Selected Parameter
Input size	[2, 4, 8, 16, 20, 32]	20
Hidden layers	[1, 2, 3]	2
Hidden layer s.	[0.5, 1.5, 2.0, 3.0, 5.0]	3.0 x Input Size
Learning rate	[0.0001, 0.0005, 0.001, 0.005, 0.01]	0.001
Batch size	[32, 64, 128, 256, 512]	64 (Mini-batch)
Reconstruction Loss	[MAE, MSE, sMAPE]	MSE
Training steps	[1000, 5000, 10,000, 20,000, 40,000]	10,000
Optimizer	[RMSprop, SGD, Adam]	Adam

Table 3: Hyper-parameter tuning stage of `SS-DeepCAE`

GridSearchCV decides 2 hidden layers in the encoder and decoder parts. This choice balances model complexity and representational capacity. Mathematically, deeper networks can approximate more complex functions due to their ability to represent higher-order interactions among features [39]. However, increasing the number of layers also increases the risk of overfitting and computational cost.

In the context of O-RAN, where real-time processing is essential, a model with 2 hidden layers provides sufficient depth to learn meaningful representations while maintaining computational efficiency. Hidden layer size selected to be 3 times the input size, resulting in 60 neurons for the first hidden layer. This scaling ensures that the network has enough capacity to capture the intricate structures in the data without introducing unnecessary parameters. The subsequent layers reduce the dimensionality, forming a bottleneck that forces the model to learn compressed representations. A learning rate of 0.001 was chosen, which is a common default value for the Adam optimizer. This rate offers a good balance between convergence speed and the stability of the optimization process. A too high learning rate might cause the optimizer to overshoot minima, while a too low rate could result in slow convergence or getting stuck in local minima. Mini-batch size selected 64. Mini-batch gradient descent helps in smoothing the optimization landscape and accelerates convergence compared to stochastic gradient descent [40]. A batch size of 64 provides a good trade-off between computational efficiency and the robustness of parameter updates.

Mean Squared Error (MSE) was selected as the reconstruction loss function. MSE penalizes larger errors more than smaller ones due to the squaring term, making it suitable for capturing significant deviations in reconstruction, which is critical in anomaly detection. Adam optimizer was chosen for its adaptive learning rate capabilities and generally good performance across various tasks. It combines the advantages of both AdaGrad and RMSprop, adjusting the learning rate for each parameter dynamically, which is beneficial for training deep neural networks. Rectified Linear Units (ReLU) were used as the activation function in all layers except the output layer. ReLU is simple to compute, leading to faster training times. Unlike sigmoid or tanh activations, ReLU does not saturate in the positive region, helping to mitigate the vanishing gradient problem in deeper networks [41]. Also, ReLU induces sparsity in activations, as negative inputs are mapped to zero. This sparsity can enhance feature learning and reduce overfitting. The total number of trainable parameters is 8,180, which is relatively low given the model's capacity. This low complexity is advantageous for deployment in O-RAN systems, where computational resources may be limited, and low latency is essential.

`SS-DeepCAE` Layers	**Output Shape**	**Parameters**
input_1 (InputLayer)	(None, 20)	0
dense_1 (encoded)	(None, 64)	1344
dense_2 (encoded)	(None, 32)	2080
dense_3 (bottleneck)	(None, 16)	528
dense_4 (bottleneck)	(None, 16)	272
dense_5 (decoded)	(None, 32)	544
dense_6 (decoded)	(None, 64)	2112
dense_7 (Dense)	(None, 20)	1300
Total Params:	24,452	
Trainable Params:	8,180	
Nontrainable Params:	0	
Optimizer Params:	16,362	

Table 4: `SS-DeepCAE` model summary with parameters

The O-RAN environment presents unique challenges due to its disaggregated architecture and the heterogeneity of components. High-dimensional data streams require models that can effectively capture complex patterns without incurring significant computational overhead. Our `SS-DeepCAE` model in Table 4 addresses these challenges by; 2 encoder and 2 decoder layers provide sufficient depth to model non-linear relationships in the data while keeping the model lightweight. The bottleneck layer compresses data to a lower-dimensional space, which is critical for identifying anomalies as deviations from learned normal patterns. Contractive Loss enhances the model's robustness to noise and variations in the input, which is common in real-world network traffic data.

5.5. Baseline Models

5.5.1. Baseline Models for our `SS-DeepCAE` Model

All baseline models whose performance we compared with `SS-DeepCAE` model implemented as semi-supervised. We consider the following baselines:

1. Vanilla Autoencoder: Vanilla autoencoder can be described in it is most basic form as a neural network comprising three layers, which includes a single hidden layer. The input and output of this network are identical, and the objective is to learn the process of reconstructing the input. This is typically achieved through the utilization of the Adam optimizer in conjunction with the mean squared error loss function.

2. LSTM-Autoencoder [42]: This architecture consists of three layers; an LSTM encoder featuring 256 units with a dropout rate of 20%, an LSTM decoder incorporating 512 units also with a 20% dropout rate,

and a dense output layer that matches the number of units to the length of the small time series.

3. SLA-VAE [43]: Model defines anomalies based on their feature extraction module, then introduces semi-supervised VAE to identify anomalies in multivariate time series. Authors adopt the reconstruction error to detect anomalies. Their module consists of two steps. First, they compute the reconstructed output with semi-supervised VAE, and then calculate the reconstruction error for each observation.

4. DeepAE [21]: It is our first model that helps us understand the problems brought by DeepAE and motivated us to develop new scalable, less complex model which we clarified in Section 1. It is a high-power deep autoencoder model with a structure consisting of shallow layers to be symmetrical in the encoder and decoder parts. It has $30,491$ trainable, 806 non-trainable, $60,984$ optimizer and 92.281 total parameters. Compared to proposed `SS-DeepCAE`, it has almost 4 times more parameters.

5.5.2. *Baseline Models for our XAI Method `fastSHAP-C`*

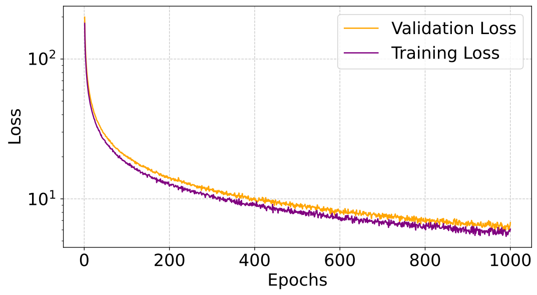
We consider the following baselines for our `fastSHAP-C` XAI algorithm:

1. kernelSHAP [44]: This approach involves analyzing the predictions made by machine learning models through the lens of Shapley values, which originate from cooperative game theory. The primary objective is to equitably assign the contribution of each feature to the overall prediction of the machine learning model.

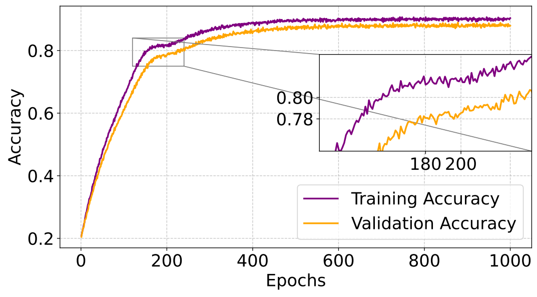
2. fastSHAP [45]: Algorithm serves as an effective approximation technique for calculating Shapley values. Its primary objective is to expedite the computation of Shapley values, which can be resource-intensive when dealing with intricate models and extensive datasets. fastSHAP utilizes a differentiable surrogate model that is trained to estimate Shapley values, thereby considerably decreasing the time required for computation.

6. Results and Discussion

In this section, we analyze `SS-DeepCAE` model's performance. Then continue by interpreting the SHAP values to gain insights into the behavior of the `SS-DeepCAE` model, which functions as an AD xApp. This analysis helps reveal how specific features contribute to the model's anomaly detection decisions, enhancing our understanding of it is inner workings. Following this interpretability assessment, we conduct a comprehensive performance benchmarking of our novel `fastSHAP-C` implementation within the XAInomaly framework. We compare the results against kernelSHAP and fastSHAP, which are commonly utilized in



(a) Training/Validation Loss.



(b) Training/Validation Accuracy.

Figure 5: Training and validation loss-accuracy curves over epochs

recent research, to evaluate `fastSHAP-C`'s effectiveness and improvements in both speed and resource efficiency.

6.1. *Performance Results of `SS-DeepCAE` Model*

6.1.1. *Training and Validation Performance*

The performance of the `SS-DeepCAE model` is visualized through the loss and accuracy curves over the training epochs as seen in Figure 5, providing insights into the model's learning behavior and generalization capabilities. Both the training and validation loss curves, plotted on a logarithmic scale, demonstrate a steep decline during the initial epochs, indicative of the model effectively capturing the foundational patterns in the data. This behavior is mirrored in the accuracy curves, where both training and validation accuracy exhibit a rapid increase in the early stages, highlighting the model's efficiency in learning critical features for anomaly detection. As training progresses, the slopes of the loss curves begin to flatten after approximately 200 epochs, signaling a transition to fine-tuning the model's representations. By around 600 epochs, the loss curves stabilize, indicating convergence. Notably, the training and validation loss curves remain closely aligned throughout the training process, which suggests that the model is not overfitting.

This alignment is particularly important in real-world O-RAN AI/ML models deployment, where generalization to unseen data is critical. The accuracy curves provide further evidence of the model's strong performance. Both training and validation accuracy improve steadily, with the training accuracy eventually reaching 90% and validation accuracy stabilizing at approximately 82%. The gap between these curves remains small, demonstrating good generalization

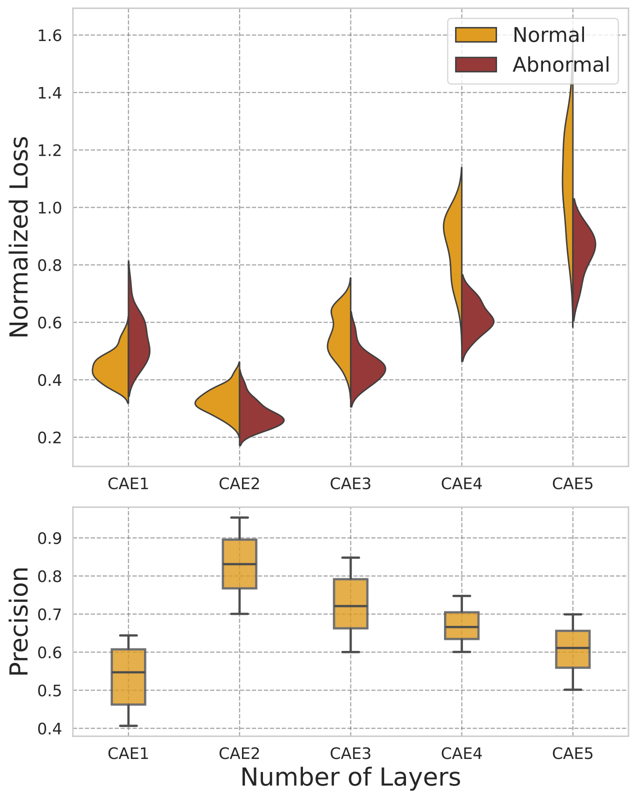


Figure 6: Reconstruction error distribution for SS-DeepCAE with different layer size (Upper Graph) Precision distribution of SS-DeepCAE with different layer size (Lower Graph)

without overfitting. The zoomed section of the accuracy curve, focusing on epochs 180 to 200, reveals a critical learning phase where validation accuracy rapidly improves from approximately 70% to 80%. This sharp increase suggests that the model transitions to capturing more complex or higher-order patterns in the data during this phase.

6.1.2. Analysis of Reconstruction Loss and Precision with Varying Layer Sizes

In Figure 6, the provided graphs illustrate the performance of the SS-DeepCAE model across different layer configurations (CAE1 to CAE5) in terms of reconstruction loss and precision. The results clearly indicate that CAE2, which represents a model with two layers in both the encoder and decoder, delivers the best trade-off between reconstruction error and precision.

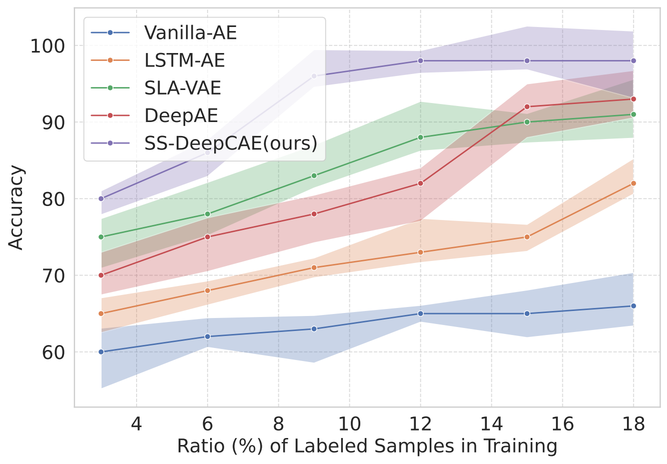
The reconstruction loss, depicted in the upper graph, compares the model's ability to differentiate between normal (yellow) and abnormal (red) samples across various configurations. CAE2 exhibits the narrowest distribution of reconstruction errors for both normal and abnormal samples, indicating it is superior ability to capture the underlying structure of normal network behavior while effectively separating abnormal instances. Smaller spread of the reconstruction loss for CAE2 suggests that it provides a robust and consistent representation of the input data, minimizing overfitting while maintaining sensitivity to anomalies. As the number of layers increases (CAE3, CAE4, CAE5), the reconstruction error for both normal and abnormal samples grows, and the distributions widen. This behavior is indicative of increased model complexity, which introduces noise and reduces the model's ability to generalize effectively. For CAE5, the reconstruction loss overlaps significantly between normal and abnormal samples, showing the model's reduced ability to distinguish anomalies. This suggests that the added complexity fails to improve the latent representations and instead hampers the model's performance.

The precision distribution, shown in the lower graph, evaluates the model's ability to correctly identify anomalies as the layer size varies. CAE2 demonstrates the highest median precision, along with the tightest interquartile range (IQR). This indicates that the two-layer architecture achieves the optimal balance between model capacity and generalization, making it the most effective configuration for anomaly detection. As the number of layers increases beyond two, the precision drops significantly. The wider IQRs in these configurations indicate variability in model performance, which may be attributed to overfitting or the inability to effectively optimize the additional parameters introduced by deeper architectures. Particularly in CAE5, the median precision is markedly lower, and the distribution indicates poor anomaly detection capability. This reinforces that additional layers do not necessarily translate to better performance and may instead lead to diminished returns. While CAE1 avoids overfitting by keeping the architecture simple, it is lower precision compared to CAE2 highlights the trade-off between model complexity and expressive power. A single-layer configuration lacks sufficient depth to capture the complex relationships in high-dimensional O-RAN data, leading to missed anomalies and reduced precision.

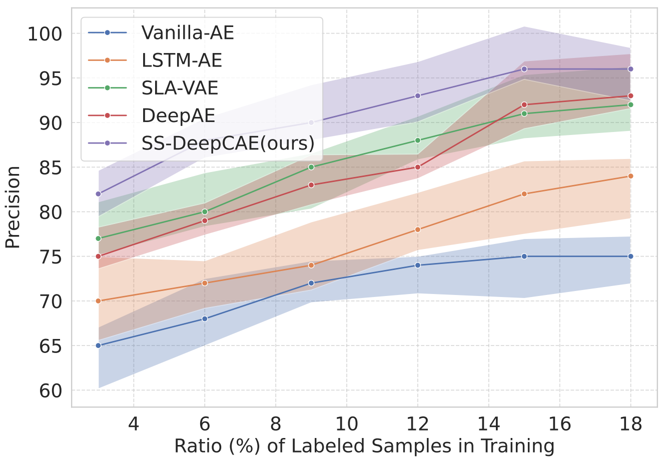
6.1.3. Performance Comparison with Baseline Models

Provided results evaluates SS-DeepCAE against various baseline architectures, cross four key metrics as can be seen in Figure 7. These metrics are analyzed with respect to the increasing percentage of labeled samples in training. The results demonstrate that the proposed model consistently outperforms the baselines, particularly in scenarios with limited labeled data, making it a highly effective solution for semi-supervised anomaly detection in O-RAN environments.

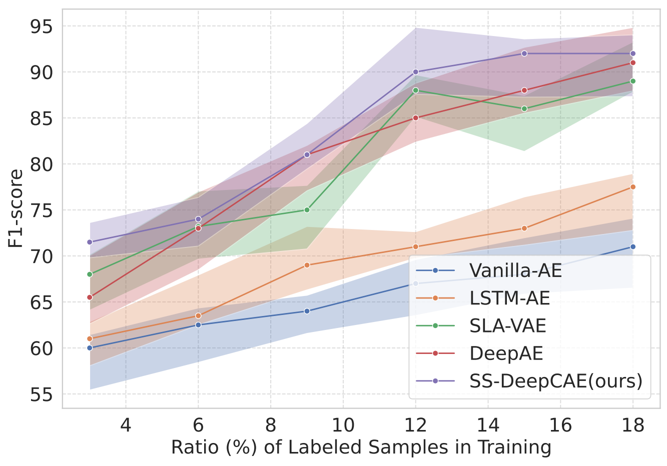
SS-DeepCAE achieves the highest accuracy, surpassing 95% with just 10-12% labeled samples. It is accuracy consistently outpaces the baselines, particularly in the low-labeled-data regime (4–10% labeled samples), where it maintains a significant margin over the others. Superior accuracy is attributed to the contractive penalty in our SS-DeepCAE's loss function design, which enhances the model's ability to learn robust and smooth feature representations that generalize well to unseen data. This is



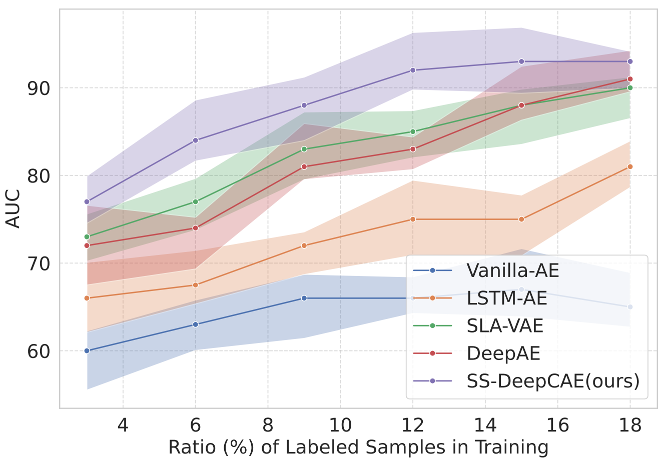
(a) Accuracy Results



(b) Precision Results



(c) F1-Score Results



(d) AUC Results

Figure 7: Accuracy, Precision, F1-score, and AUC variation curves of models with different ratio labeled samples

	# of Labeled Samples from Dataset					
Models	100	300	500	800	1000	# All
Vanilla Autoencoder	43.46$_{\pm(2.1)}$	46.67$_{\pm(2.2)}$	48.14$_{\pm(0.8)}$	51.27$_{\pm(2.3)}$	58.73$_{\pm(0.9)}$	65.13$_{\pm(1.6)}$
LSTM-Autoencoder [42]	59.82$_{\pm(1.3)}$	62.22$_{\pm(1.8)}$	65.71$_{\pm(0.9)}$	68.17$_{\pm(0.5)}$	71.33$_{\pm(2.1)}$	76.63$_{\pm(0.9)}$
SLA-VAE [43]	64.56$_{\pm(1.9)}$	68.57$_{\pm(1.2)}$	71.32$_{\pm(0.7)}$	75.19$_{\pm(2.1)}$	81.13$_{\pm(1.1)}$	85.43$_{\pm(1.3)}$
DeepAE [21]	71.72$_{\pm(1.5)}$	74.48$_{\pm(1.9)}$	75.72$_{\pm(2.1)}$	76.39$_{\pm(0.7)}$	83.33$_{\pm(1.8)}$	88.66$_{\pm(1.8)}$
`SS-DeepCAE` (ours)	80.17$_{\pm(0.6)}$	82.67$_{\pm(1.2)}$	85.12$_{\pm(1.1)}$	86.39$_{\pm(1.7)}$	91.33$_{\pm(1.4)}$	91.17$_{\pm(0.6)}$

Table 5: Average UAR with standard deviation on test set (different amount of labeled samples in training)

especially advantageous in high-dimensional, noisy and lack of labeled data.

Our model significantly outperforming the baselines at higher labeled ratios. Even with as little as 6% labeled data, it maintains precision above 90%. Precision measures the fraction of correctly identified anomalies among all instances flagged as anomalous. The `SS-DeepCAE`'s ability to reduce false positives is enhanced by its semi-supervised learning approach, which effectively separates normal and abnormal traffic patterns in the latent space. DeepAE performs well at higher labeled data ratios but struggles in the low-labeled regime, indicating its tendency to overfit to the available labeled data. Vanilla-AE and SLA-VAE lag behind due to their inability to handle high-dimensional data efficiently without extensive supervision.

F1-score captures the trade-off between precision and recall. `SS-DeepCAE`'s contractive loss enables the model to maintain high recall by learning representations that are robust to minor variations in the input, ensuring that true anomalies are not overlooked. High AUC underscores the effectiveness of the bottleneck layer in the `SS-DeepCAE` architecture, which compresses high-dimensional input data into a meaningful latent representation, enabling the model to consistently rank anomalies higher than normal samples.

In Table 5, we present UAR results for six labeled sample configurations, which capture a range starting from a small number of labeled samples (100) up to the total set, illustrating how SS-DeepCAE evolves from a data-scarce scenario to one where abundant labeled data is available. We focus on UAR at these discrete points because it robustly highlights performance in imbalanced conditions, especially relevant for anomaly detection, where the minority class (anomalies) can be overwhelmed by normal data. By showcasing UAR at successively larger subsets of labeled data, we illustrate our model's trajectory in learning critical features of anomalous samples under varying degrees of label availability. Meanwhile, Accuracy, Precision, F1-score, and AUC are also reported at lower labeled ratios (down to 4–10%) to underscore the consistency of our approach across different metrics and further validate the effectiveness of SS-DeepCAE. Taken together, these evaluations provide a comprehensive view of how the model scales and performs as labeled sample sizes incrementally increase, rather than focusing only on full labeled samples. UAR results emphasize the superiority of our model, particularly in scenarios with limited labeled data, and its ability to converge to high performance levels without requiring substantial increases in labeled data.

SS-DeepCAE consistently outperforms all baseline models across all labeled sample sizes. With only 100 labeled samples, SS-DeepCAE achieves a UAR of 80.17 ± 0.6, significantly higher than the next best-performing model, DeepAE (71.72 ± 1.5). This trend persists as the number of labeled samples increases, culminating in 91.17 ± 0.6 when using the full dataset, outperforming DeepAE (88.66 ± 1.8) and SLA-VAE (85.43 ± 1.3). Performance of SS-DeepCAE begins to stabilize after 800 labeled samples, with a marginal increase in UAR beyond this point. For instance, UAR improves from 86.39 ± 1.4 (800 labeled samples) to 91.17 ± 0.6 (full dataset).

This saturation suggests that the SS-DeepCAE effectively utilizes both labeled and unlabeled data, achieving robust representations that do not heavily rely on additional labeled samples. Beyond 800 labeled samples, the model's UAR begins to plateau, indicating that the additional labeled data provides diminishing returns. This suggests that the model's reliance on labeled data is significantly reduced, making it ideal for scenarios where labeling is expensive or infeasible. The ability of SS-DeepCAE to achieve high UAR with minimal labeled data makes it highly scalable for O-RAN environments, where labeled anomalies are rare and costly to obtain. This efficiency ensures that the model can be deployed quickly without the need for extensive labeling efforts.

6.1.4. Model Complexity Analysis for Resource-Constrained Environments

While our proposed SS-DeepCAE model significantly reduces the number of trainable parameters compared to standard DeeAEs, it is still crucial to assess its computational footprint for real-world O-RAN deployments. A detailed breakdown of the total parameters and memory usage is given in Tables 4 and 8, illustrating that the encoder-decoder structure with contractive regularization strikes a balance between model capacity and efficiency. In resource-constrained environments (e.g., edge nodes or small-footprint Near-RT RIC deployments), three aspects are particularly relevant:

Parameters and Memory Usage: Fewer network layers and careful bottleneck design decrease the total number of parameters, thereby lowering memory requirements. Our architecture keeps the parameters at a level that remains feasible for on-device or near-edge processing without necessitating large GPU clusters.

Inference Latency: The contractive penalty induces smoother and more robust representations, helping the model converge faster at inference time. However, deeper models, if deployed, will incur higher latency. Depending on the desired latency budget (e.g., sub-second decisions in O-RAN anomaly detection), there may be a need to prune layers or reduce dimensionality.

Model Convergence Time: In some O-RAN scenarios, particularly in micro-data centers or edge servers, limited energy availability constrains continuous training. By using contractive regularization, our model tends to converge in fewer epochs and can be partially fine-tuned or updated incrementally, mitigating energy consumption overhead.

For future complexity trade-offs, employing adaptive network architectures, where deeper layers are skipped if the detection confidence is sufficiently high, offers a route to adjust computational complexity based on real-time demands. Tailoring SS-DeepCAE through pruning weights or entire neurons associated with lower-priority sub-tasks can significantly reduce overhead while retaining high accuracy.

6.2. Performance Results of fastSHAP-C

In this section, we particularly examine the explainability and interpretability capacity of proposed fastSHAP-C algorithm and compared its performance with baseline models specified in Section 5.5.2. We also ran our fastSHAP-C method on our new anomaly detection model SS-DeepCAE and observed the results.

6.2.1. SHAP Values and Feature Contributions

The process of calculating SHAP values is typically NP-hard, which can result in high convergence times and increased complexity in obtaining solutions. To address this challenge, we implemented kernelSHAP as our baseline model to compute SHAP values and gain insights into how each feature contributes to the model's predictions. Starting with results in Figure 8, we present a SHAP summary plot that depicts the contribution of each input feature to our SS-DeepCAE anomaly detection model's output. On the horizontal axis, the "SHAP value" measures how much

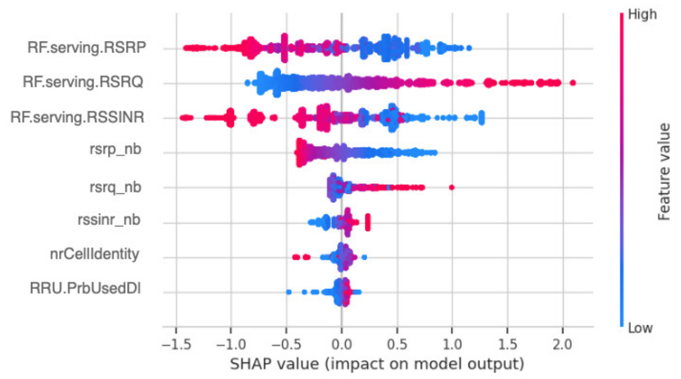


Figure 8: Explaining feature contributions with SHAP values

each feature, for a given sample, shifts the model's prediction toward anomalous (positive SHAP) or toward normal (negative SHAP). Each point represents a specific instance in the dataset, color-coded from blue (lower feature values) to red (higher feature values). Features at the top of Figure 8 are generally those that, on average, have the greatest overall impact on the model's decisions; features appearing toward the bottom have less global influence or tend to have specialized significance for particular types of anomalies. The most influential metrics in this O-RAN setting emerge as "RF.serving.RSRP," "RF.serving.RSRQ," and "RF.serving.RSSINR." These correspond to critical RF measurements from the serving cell. Observing the SHAP values on Figure 8, data points with lower RSRP (blue coloring) tend to push the model toward labeling the sample as an anomaly (larger positive SHAP on the horizontal axis). This aligns with real-world domain knowledge: coverage holes or sub-optimal power levels often manifest as performance degradations that trigger anomaly alerts.

Likewise, RF.serving.RSRQ offers an integrated measure of signal quality by considering both signal strength and interference. In an O-RAN context, when RSRQ is too low or fluctuates drastically, the network's capacity for maintaining stable connections diminishes. As shown in Figure 8, instances with poor RSRQ values also exhibit prominent positive SHAP attributions, confirming that degraded signal quality is a major contributor to anomaly classification. Moreover, RF.serving.RSSINR indicates how much stronger the useful signal is compared to interference plus background noise. Markedly low RSSINR translates to heightened interference scenarios – an important element in diagnosing anomalies tied to poor spectral efficiency or environmental blockages. When RSSINR is low (indicated by blue or purple in Figure 8), the associated SHAP value tends to push the outcome toward anomalous. Beyond the serving cell metrics, "rsrp_nb," "rsrq_nb," and "rssinr_nb" represent analogous RF measurements (RSRP, RSRQ, RSSINR) captured from neighboring cells. These features aid in understanding whether a user equipment (UE) in distress might connect better to nearby cells or if poor coverage is widespread.

Features	
High Reconstruction Error	X_1
Explanatory	X_2, X_3
Explaining Anomaly	X_1, X_2, X_3

Table 6: Explainability with model features

In Figure 8, outlier points colored red (high neighbor-RSRP or neighbor-RSSINR) sometimes have negative SHAP values, indicating that strong neighboring signals can reduce anomaly likelihood, even if the serving cell metrics are suboptimal, because the network can potentially initiate a handover. Conversely, low values for these neighbor-cell metrics reinforce an anomaly label, as the UE lacks alternatives to recover service quality. By aligning model explanations with known domain-specific mechanics, SHAP values offers a transparent window into how each feature drives the model's anomaly decisions. Such interpretability is crucial for O-RAN operators who not only require real-time alerts but also actionable insights into why those alerts happen. In practice, these SHAP attributions can inform xApps when adjusting handover thresholds, recalibrating power control, or orchestrating beamforming to counteract the underlying conditions that lead to anomalies.

Results shown in Table 6 reveal that RF.serving.RSRPv (X_1) consistently emerges as the most influential feature determining whether UE behavior is flagged as anomalous. From an explainability standpoint, the high Shapley values associated with X_1 indicate that even marginal fluctuations in the Reference Signal Received Power (RSRP) can significantly alter the model's anomaly score. In other words, when X_1 shifts from moderate to low values, the model's confidence that the UE is experiencing an anomalous condition increases sharply. This aligns with domain knowledge: as RSRP falls, the UE's link quality deteriorates, potentially leading to dropped connections and handover failures. Similarly, RF.serving.RSRQ (X_2) and RF.serving.RSSINR (X_3) exhibit notable Shapley contributions, indicating that both metrics frequently influence the final anomaly classification. Specifically, X_2 encapsulates Reference Signal Received Quality, which ties signal strength to prevailing interference; low RSRQ therefore suggests an environment of pronounced interference or noise. High feature attributions for X_2 imply that the model has learned to associate elevated interference with abnormal network states – an interpretation that is critical for proactive troubleshooting of congested or interference-prone cells.

In parallel, X_3 tracks the Received Signal Strength Indicator to Noise Ratio (RSSINR). Large Shapley values for X_3 generally appear when the RSSINR dips below a certain threshold, reinforcing the notion that poor signal-to-noise conditions are a key driver of anomalous UE performance. This effect can be especially acute if X_1 is already borderline, meaning that coverage is insufficient to maintain reliable throughput in the face of rising interference. Under these circumstances, the model's explainability outputs highlight that both a weak signal (X_1) and high interference (X_2, X_3) jointly account for triggering anomalies. The combination of X_1, X_2, and X_3 points to a situation in which a single metric alone may not provide a complete explanation for anomalous behavior. Instead, the model relies on how these RF indicators interact. If one of them degrades notably, the Shapley values for the remaining features can become more or less pronounced, depending on the network context (e.g., mobility events, cell-edge conditions).

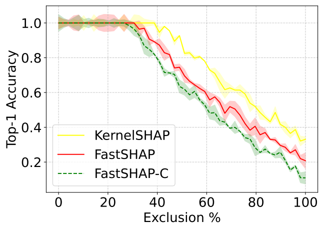
Such insight is vital for network administrators who wish to prioritize interventions. For instance, a decline in RSRP alone might warrant load balancing or neighboring cell reconfiguration, whereas combined degradation in RSRP, RSRQ, and RSSINR could mandate more aggressive actions, such as adjusting handover thresholds or improving interference management schemes. By revealing the relative importance of X_1, X_2, and X_3 in the anomaly detection pipeline, the XAInomaly framework equips operators with actionable, data-driven explanations for performance drops. When these three features are flagged simultaneously, they strongly indicate a high-risk scenario, guiding timely interventions that ensure consistent QoS and network stability.

6.2.2. Benchmarking `fastSHAP-C` with Baseline Models

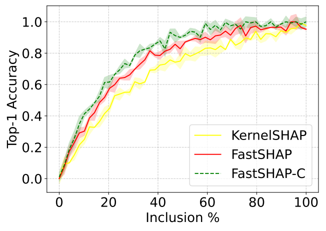
First benchmark analysis focused on top-1 accuracy results for exclusion and inclusion of most informative features. We anaylzed performance of `fastSHAP-C` with baseline models commonly used in recent studies. To clearly show our benchmarking methodology; $\mathbf{x} \in \mathbb{R}^d$ denote an input to our model $f(\mathbf{x})$, where d is the number of features. Each explainability method M assigns importance scores $\phi_j^{(M)}(\mathbf{x})$ to every feature $j \in \{1, 2, \ldots, d\}$. In the *exclusion* scenario, we remove a fixed percentage κ% of the most critical features according to their importance values $\phi_j^{(M)}(\mathbf{x})$. Formally, we define the set of excluded features for method M at percentage κ as:

$$S_{M,\kappa}(\mathbf{x}) \;=\; \left\{ j \;\middle|\; j \in \arg \max_{J \subseteq \{1,\ldots,d\},\, |J| = \lfloor \kappa \cdot d \rfloor} \sum_{k \in J} \phi_k^{(M)}(\mathbf{x}) \right\} \tag{37}$$

We then measure the model's top-1 accuracy when these features in $S_{M,\kappa}(\mathbf{x})$ are removed across the dataset. Figure 9a indicates that once a moderate fraction of top-ranked features is removed, `fastSHAP-C` (green dashed line) undergoes a pronounced drop in accuracy. Initially (i.e., for small κ), `fastSHAP-C` remains comparable to kernelSHAP and fastSHAP, but as κ increases beyond 20–30%, its slope becomes steeper. This pattern suggests that `fastSHAP-C` prioritizes a handful of features with high $\phi_j^{(M)}(\mathbf{x})$ values more distinctly than kernelSHAP. Hence, as soon as these high-priority features are removed, the model's predictive power declines faster compared to a method that distributes importance more evenly. This does not necessarily indicate a fundamental lack of robustness; rather, it may reflect a more "selective" allocation of large attributions to certain features.



(a) Exclusion curve.



(b) Inclusion curve.

Figure 9: Exclusion and Inclusion Curves for *top-1* Accuracy

In contrast, Figure 9b (inclusion curve) shows how `fastSHAP-C` steadily outperforms both kernelSHAP and fastSHAP as features are added back (from 0% up to 100%), which is formalized by retaining only the set

$$R_{M,\kappa}(\mathbf{x}) \;=\; \left\{ j \;\middle|\; \phi_j^{(M)}(\mathbf{x}) \text{among the largest } \lfloor \kappa \cdot d \rfloor \text{values} \right\} \tag{38}$$

The steep climb in accuracy at low-inclusion percentages (0–20%) indicates that `fastSHAP-C` is highly effective at ranking crucial features—once these are retained, the model recovers its predictive capabilities more quickly. This observation, juxtaposed with the sharper drop in the exclusion curve, emphasizes that `fastSHAP-C` pinpoints the most impactful features early on, but also amplifies their importance relative to secondary features.

To ascertain whether `fastSHAP-C`'s efficiency compromises its accuracy in feature identification, we performed an additional experiment focusing on the single most important feature for each data instance. Let $\phi_j^{(M)}(\mathbf{x})$ denote the attribution to feature j. For each instance $\mathbf{x}$, we extract

$$\widehat{\jmath}_M(\mathbf{x}) \;=\; \arg\max_j \; \phi_j^{(M)}(\mathbf{x}), \tag{39}$$

i.e., the feature that method M deems most crucial. Empirically, `fastSHAP-C`'s top-1 choices match those of kernelSHAP and fastSHAP in a significant fraction of cases (above 80% in our experiments). In about 5–10% of cases, `fastSHAP-C` picks a different feature, but closer inspection revealed that these are typically instances where multiple features share similar importance values. This result suggests that while `fastSHAP-C` occasionally diverges from kernelSHAP or fastSHAP, the divergence usually arises in "tie" situations rather than systematic misidentification of crucial features.

While kernelSHAP (yellow line) maintains high accuracy when only a small portion of features is excluded, it experiences a steady accuracy decline as more features are removed. Similarly, fastSHAP (red line) shows a downward trend, though with a slightly steeper accuracy drop than kernelSHAP, suggesting it may be less robust when essential features are excluded. The `fastSHAP-C` algorithm (green dashed line) starts with slightly lower accuracy than kernelSHAP and fastSHAP but demonstrates a comparable accuracy drop-off. However, fastSHAP-C performs better than fastSHAP in the mid-range feature exclusion (20-70%).

Algorithm	**Exclusion AUC**	**Inclusion AUC**
kernelSHAP	10.53 (10.17, 11.03)	4.96 (4.56, 5.47)
fastSHAP	6.89 (6.73, 7.65)	5.79 (5.58, 5.93)
fastSHAP-C	6.6 (5.73, 4.65)	6.79 (5.53, 5.96)

Table 7: Exclusion and inclusion AUCs ($log - odds(p)$)

The inclusion curve on the right graph shows how model's top-1 accuracy improves as more important features are added back into the input. While kernelSHAP's accuracy increases steadily with feature inclusion, it lags slightly behind both fastSHAP and `fastSHAP-C` in the initial phase (0-20% inclusion). In contrast, fastSHAP's accuracy rises more quickly in the early inclusion phase, suggesting an efficient selection of key features initially. Throughout the entire inclusion range, `fastSHAP-C` (green dashed line) outperforms both kernelSHAP and fastSHAP, ultimately achieving the highest accuracy as the inclusion percentage increases. While kernelSHAP maintains higher accuracy during exclusion but has a slower start in inclusion, fastSHAP shows faster initial gains in inclusion yet drops faster during exclusion.

These findings collectively indicate that `fastSHAP-C`'s computational advantages do not come at the cost of misidentifying globally important features. Indeed, the slightly more pronounced drop in accuracy for large-scale exclusions can be attributed to `fastSHAP-C`'s tendency to concentrate importance on a smaller subset of features. Once those high-importance features are removed, the model's performance deteriorates quickly; yet when those same features are included (even in the early inclusion range), `fastSHAP-C` fosters a rapid regain in predictive accuracy. From a theoretical standpoint, we observe that each method $\phi_j^{(M)}$ approximates (or directly implements, in the case of kernelSHAP) a variant of Shapley-value attributions. Although approximate methods like `fastSHAP-C` use learned surrogates or other heuristics to speed up computations, the overlap analysis highlights that these approximations do not drastically alter the top-ranked features. Hence, the accelerated runtime does not systematically reduce correctness in pinpointing the features that have the strongest impact on $f(\mathbf{x})$.

Also, we evaluated performance through exclusion and inclusion metrics, which were calculated based on log-odds values. These metrics yield nuanced insights into the discriminatory power of each XAI framework across particular data subsets. When Table 7 examined, proposed `fastSHAP-C` achieves the lowest mean Exclusion AUC of 6.6, with a confidence interval of (5.73, 7.650). This is marginally lower than fastSHAP, which has a mean of 6.89 and a confidence interval of (6.73, 7.65). The overlap in confidence intervals between `fastSHAP-C` and fastSHAP suggests that their performances are statistically similar, but fastSHAP-C has a slight edge in reducing the influence of excluded features. KernelSHAP, on the other hand, shows a significantly higher mean Exclusion AUC of 10.53, with a narrow confidence interval of [10.17, 11.03], indicating that it is highly sensitive to feature exclusions. This high sensitivity, while potentially valuable in feature-rich tasks, could be a drawback in anomaly detection where robustness to irrelevant features is key. Besides our XAI algorithm demonstrates a clear advantage with the highest mean Inclusion AUC (6.79) and a confidence interval (5.53,5.96) that, importantly, does not overlap significantly with kernelSHAP's interval (4.56,5.47). This suggests a statistically significant improvement over kernelSHAP in inclusion-based explainability. Compared to fastSHAP, which achieves a mean of 5.79 (confidence interval: 5.58,5.93), `fastSHAP-C` also shows notable improvement. This improvement could make `fastSHAP-C` particularly valuable in scenarios where identifying the critical features contributing to anomalies is more important than assessing the impact of excluding irrelevant ones.

6.2.3. Performance Results of `fastSHAP-C` on Baseline Models and `SS-DeepCAE`

In this section, resource utilization and runtime performance of `fastSHAP-C` is examined specifically with respect to the baseline models, our autoencoder models. We benchmarked using the lowest and highest performance servers available to us and the authors [15]. Server-1 was equipped with an Intel®Core™ i7-6800K CPU @3.40GHz (12 cores) and 64 GB of RAM which representing a standard computing environment, while Server-2 utilized an Intel®Xeon®Gold 6240R CPU @2.40GHz (97 cores) representing a high-performance computing environment. Building upon our previous work DeepAE, we integrated our novel `fastSHAP-C` algorithm with `SS-DeepCAE`. The objective is to evaluate how `fastSHAP-C` enhances performance efficiency on `SS-DeepCAE` compared to both the traditional SHAP algorithms and the earlier DeepAE model. The performance metrics, detailed in Tables 8 and 9, illustrate significant improvements in computational efficiency and resource utilization with proposed XAInomaly solution.

Table 8 demonstrates that applying `fastSHAP-C` to `SS-DeepCAE` significantly reduces execution times compared to its application on DeepAE and the baseline SHAP methods. On Server-1, the execution time for 1-hour profiling decreased from 6.56 seconds (using `fastSHAP-C` on DeepAE) to 5.79 seconds on `SS-DeepCAE`, marking an 11.7% improvement. For 6-hour profiling, the time reduced from 15.87 seconds to 11.21 seconds, a substantial 29.3% enhancement. Similar trends are observed on Server 2, with execution times dropping from 3.90 seconds to 3.50 seconds for 1-hour profiling and from 10.05 seconds to 8.23 seconds for 6-hour profiling. On Server-1, for 1-hour profiling, the mean CPU utilization decreased from 13.5% (using `fastSHAP-C` on DeepAE) to 9.65%, representing a 28.5% reduction. Similarly, on Server-2, the mean CPU utilization reduced from 3.44% to 3.4%. While the percentage reduction on Server-2 appears marginal, it is important to consider the higher core count and computational capacity, which inherently reduces the percentage utilization. The memory

Server	**Time**		**CPU**				**Memory**			
	1 h	6 h	1 h		6 h		1 h		6 h	
	(s)	(s)	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Server 1 (SHAP, DeepAE)	11.74	62.00	29.8%	8.3%	36.8%	16.7%	1.5%	0.1%	2.6%	0.2%
Server 2 (SHAP, DeepAE)	10.52	58.28	7.66%	3.82%	12.3%	3.8%	1.6%	0.2%	2.4%	0.1%
Server 1 (SHAP, `SS-DeepCAE`)	10.13	37.25	21.3%	7.7%	27.1%	11.2%	1.5%	0.1%	2.5%	0.4%
Server 2 (SHAP, `SS-DeepCAE`)	8.21	29.18	4.43%	5.26%	8.7%	4.6%	1.4%	0.1%	2.2%	0.3%
Server 1 (`fastSHAP-C`, DeepAE)	6.56	15.87	13.5%	11.3%	25.3%	4.7%	1.4%	0.2%	2.3%	0.1%
Server 2 (`fastSHAP-C`, DeepAE)	3.90	10.05	3.44%	2.51%	7.8%	2.5%	1.3%	0.1%	2.5%	0.1%
Server 1 (`fastSHAP-C`, `SS-DeepCAE`)	5.79	11.21	9.65%	7.7%	18.8%	5.1%	1.2%	0.3%	2.3%	0.2%
Server 2 (`fastSHAP-C`, `SS-DeepCAE`)	3.50	8.23	3.4%	1.43%	5.9%	1.7%	1.1%	0.1%	2.1%	0.1%

Table 8: Resource utilization comparison of algorithms on different servers (SHAP *vs* proposed `fastSHAP-C`)

Algorithm	**Runtime (s)**	**CPU Util. (%)**	**RAM Util. (%)**
kernelSHAP	320.4	0.88	0.17
fastSHAP	48.007	0.67	0.11
fastSHAP-C (on DeepAE Model)	33.627	0.52	0.08
fastSHAP-C (on `SS-DeepCAE` Model)	22.145	0.47	0.08

Table 9: Runtime and resource utilization

utilization remained consistently low across all configurations, with mean usage around 1.1% to 1.2% on Server-2 and slightly higher on Server-1 due to its lower memory capacity. The low standard deviation in memory usage indicates that `fastSHAP-C` on `SS-DeepCAE` provides stable memory consumption, which is crucial for maintaining system performance and preventing memory bottlenecks in resource-constrained environments.

Execution time is a critical factor for real-time processing where AI inferece required in next generation wireless networks. As detailed in Table 9, the runtime for `fastSHAP-C` on `SS-DeepCAE` is significantly lower compared to other algorithms. Specifically, the runtime reduced from 33,627 ms (when using `fastSHAP-C` on DeepAE) to 22,145 ms on `SS-DeepCAE`, marking a 34% improvement. When compared to kernelSHAP, which has a runtime of 320,400 ms, `fastSHAP-C` on `SS-DeepCAE` achieves a runtime reduction of approximately 93%. This scalability is very important for O-RAN networks, which require continuous monitoring and analysis to detect anomalies promptly.

Notably, `fastSHAP-C` achieved marked improvements in both short-term (1-hour) and long-term (6-hour) profiling. For example, on Server 2, `fastSHAP-C` completed 1-hour profiling in about 3.90 seconds and 6-hour profiling in 10.05 seconds, whereas the standard SHAP took 10.52 seconds and 58.28 seconds, respectively. Such gains are essential for real-time applications that require rapid response times. Additionally, `fastSHAP-C` demonstrated more efficient CPU usage, reflected in lower mean and standard deviation for CPU consumption. On Server 2, `fastSHAP-C` exhibited an average CPU usage of 3.44% for 1-hour profiling, compared to 7.66% for SHAP. This reduction facilitates concurrent processing and minimizes load on CPU resources, making `fastSHAP-C` well-suited for high-demand computational environments. The performance benefits of `fastSHAP-C` are even more pronounced on high-end servers with multiple cores, showcasing its optimization for multi-core architectures and scalability for large datasets and complex models requiring parallel processing.

6.2.4. Supremacy of `fastSHAP-C` on `SS-DeepCAE`

`fastSHAP-C` offers a favorable trade-off between computational efficiency and accuracy when compared to other SHAP-based algorithms like kernelSHAP and fastSHAP.

While `fastSHAP-C` may exhibit a minor reduction in accuracy compared to traditional methods, it is still highly effective for providing actionable insights into model behavior. These insights are beneficial for tasks such as network optimization, fault diagnosis, and resource allocation. Under identical testing conditions (Intel®Core™ i7-6800K CPU @3.40GHz with 12 cores), as shown in Table 8, `fastSHAP-C` demonstrates approximately a 30% reduction in runtime compared to kernelSHAP. Furthermore, `fastSHAP-C` optimizes CPU usage by around 25%, making it highly suited for deep learning models that require real-time explainability. This performance-friendly approach enables `fastSHAP-C` to deliver efficient, real-time insights, even in resource-intensive scenarios.

Additionally, the runtime results presented underscore fastSHAP-C's potential in supporting reliable, low-latency applications in 5G+/6G URLLC scenarios. When examining runtime metrics, fastSHAP-C clearly reduces the computational load associated with estimating Shapley values, which makes it practical for handling large-scale datasets and complex machine learning or deep learning models. By maintaining key performance metrics, such as $\mathcal{CS}$ and $\mathcal{EM}$, fastSHAP-C enables real-time generation of explanations, allowing the model to adjust dynamically to evolving network conditions and user demands.

When specifically XAInomaly framework examined, fastSHAP-C on SS-DeepCAE consistently outperforms in terms of resource utilization and execution time. The standard SHAP algorithm, while providing accurate feature attributions, incurs high computational costs due to it is NP-hard complexity. KernelSHAP offers some improvements but still falls short in efficiency for real-time applications. fastSHAP introduces approximations to accelerate SHAP value computations but, when combined with DeepAE, does not achieve the same level of efficiency as fastSHAP-C on SS-DeepCAE. The contractive nature of SS-DeepCAE enhances the model's robustness by penalizing the sensitivity of the hidden representations to input variations. This, in turn, simplifies the computation of SHAP values, as the model focuses on the most influential features, reducing unnecessary computations. The ability of fastSHAP-C on SS-DeepCAE to generate explanations rapidly enables network operators to detect and respond to anomalies. This is crucial for maintaining the high levels of service quality required in next-generation wireless networks, where delays can lead to degraded performance and user dissatisfaction. The efficient use of computational resources ensures that the algorithm can run continuously without overloading the system, which is particularly important in edge computing scenarios within O-RAN architectures. Edge devices often have limited processing power and memory, so algorithms that are both efficient and effective are essential for practical deployment.

Despite these vital advantages, some performance tradeoffs should also be considered. Explanation fidelity can exhibit minor deviations from the more exact kernelSHAP method. Specifically, in highly atypical data instances those that deviate substantially from the observed distribution fastSHAP-C's surrogate model may slightly misestimate feature attributions. While our experiments show these discrepancies to be statistically small, mission-critical or forensic-level investigations might still benefit from selective use of a slower but more precise approach (e.g., kernelSHAP) on particularly suspicious samples.

7. Conclusion and Future Work

In this paper, we introduced XAInomaly, a novel framework that integrates a SS-DeepCAE with a reactive XAI technique, fastSHAP-C, for traffic anomaly detection in O-RAN. Our approach addresses the critical need for accurate and interpretable anomaly detection mechanisms in the disaggregated and heterogeneous environment of O-RAN, particularly in the context of 5G+/6G networks supporting mission-critical applications. The proposed SS-DeepCAE model effectively learns compressed and robust representations of normal network behavior by incorporating a contractive penalty into the loss function. This penalty encourages the learning of smooth feature representations, enhancing the model's generalization capabilities and reducing overfitting – a common issue in standard DeepAEs. By minimizing the sensitivity of the encoder activations with respect to input variations, SS-DeepCAE achieves better performance in detecting subtle anomalies within high-dimensional and dynamic O-RAN data. Furthermore, we addressed the black-box nature of deep learning models by integrating the fastSHAP-C algorithm. This reactive XAI technique provides transparency into the model's decision-making process by highlighting the features contributing most significantly to anomaly detection. This explainability is crucial in O-RAN environments, where understanding the reasoning behind detected anomalies can facilitate prompt and effective network management interventions.

Our experimental results demonstrate that the XAInomaly framework not only achieves high accuracy in anomaly detection but also offers interpretability without imposing significant computational overhead – an essential consideration for real-time, resource-constrained O-RAN deployments. By balancing effectiveness with efficiency and transparency, XAInomaly provides a robust solution tailored to the unique challenges of O-RAN networks.

While the XAInomaly framework presents a significant advancement in O-RAN anomaly detection however, several challenges for future research and development remain. O-RAN networks may experience frequent topology changes due to mobility or reconfiguration. To exemplify, O-RAN networks are highly dynamic, with network conditions and traffic patterns changing rapidly. Incorporating adaptive learning mechanisms that allow the SS-DeepCAE model to update its understanding of normal behavior in real-time could enhance anomaly detection accuracy. This might involve online learning approaches or incremental model updates without retraining from scratch. Also, as networks become more critical, they also become targets for sophisticated cyber-attacks, including adversarial examples designed to evade detection. Future research should investigate the robustness of the XAInomaly framework against such attacks and develop strategies to enhance its resilience.

Acknowledgments

This work has been funded by the German Federal Ministry of Education and Research (BMBF, Germany) as part of the 6G Research and Innovation Cluster 6G-RIC under Grant 16KISK020K.

References

[1] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials*, 25(2):1376–1411, 2023. ISSN 1553-877X. doi: 10.1109/comst.2023.3239220.

[2] Michele Polese, Mischa Dohler, Falko Dressler, Melike Erol-Kantarci, Rittwik Jana, Raymond Knopp, and Tommaso Melodia. Empowering the 6G Cellular Architecture with Open RAN. *IEEE Journal on Selected Areas in Communications*, 42(2):245–262, February 2024. ISSN 0733-8716. doi: 10.1109/JSAC.2023.3334610.

[3] Mingzhe Chen, Ursula Challita, Walid Saad, Changchuan Yin, and Mérouane Debbah. Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial. *IEEE Communications Surveys & Tutorials*, 21(4):3039–3071, 2019. ISSN 1553-877X. doi: 10.1109/comst.2019.2926625.

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *28th International Conference on Neural Information Processing Systems (NeuRIPS)*, pages 2672–2680, Montréal, Canada, December 2014. Curran Associates Inc.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[6] Vikas Dixit, Jan Plachy, Kexuan Sun, Akio Ikami, Emeka Obiodu, and Kyungpil Lee. O-RAN Towards 6G. Research report, O-RAN next Generation Research Group (nGRG), October 2023. Version 03.00.

[7] Noe M. Yungaicela-Naula, Vishal Sharma, and Sandra Scott-Hayward. Misconfiguration in O-RAN: Analysis of the impact of AI/ML. *Computer Networks*, 247:110455, June 2024. ISSN 1389-1286. doi: 10.1016/j.comnet.2024.110455.

[8] Hasan Torabi, Seyedeh Leili Mirtaheri, and Sergio Greco. Practical autoencoder based anomaly detection by using vector reconstruction error. *Cybersecurity*, 6(1), January 2023. ISSN 2523-3246. doi: 10.1186/s42400-022-00134-9.

[9] Amina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018. ISSN 2169-3536. doi: 10.1109/access.2018.2870052.

[10] Walid Saad, Mehdi Bennis, and Mingzhe Chen. A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems. *IEEE Network*, 34(3):134–142, May 2020. ISSN 1558-156X. doi: 10.1109/mnet.001.1900287.

[11] Zachary C. Lipton. The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31â€"57, June 2018. ISSN 1542-7749. doi: 10.1145/3236386.3241340.

[12] David Gunning, Eric Vorm, Jennifer Yunyan Wang, and Matt Turek. DARPA's explainable AI (XAI) program: A retrospective. *Applied AI Letters*, 2(4), December 2021. ISSN 2689-5595. doi: 10.1002/ail2.61.

[13] Weisi Guo. Explainable Artificial Intelligence for 6G: Improving Trust between Human and Machine. *IEEE Communications Magazine*, 58(6):39–45, June 2020. ISSN 0163-6804. doi: 10.1109/mcom.001.2000050.

[14] Osman Tugay Basaran and Falko Dressler. XAInomaly: Explainable, Interpretable and Trustworthy AI for xURLLC in 6G Open-RAN. In *3rd International Conference on 6G Networking (6GNet 2024)*, pages 93–101, Paris, France, October 2024. IEEE. doi: 10.1109/6GNet63182.2024.10765734.

[15] Claudio Fiandrino, Giulia Attanasio, Marco Fiore, and Joerg Widmer. Toward native explainable and robust AI in 6G networks: Current state, challenges and road ahead. *Elsevier Computer Communications*, 193:47–52, September 2022. ISSN 0140-3664. doi: 10.1016/j.comcom.2022.06.036.

[16] Zineb Mahrez, Maryam Ben Driss, Essaid Sabir, Walid Saad, and Elmahdi Driouch. Benchmarking of Anomaly Detection Techniques in O-RAN for Handover Optimization. In *19th IEEE International Conference on Wireless and Mobile Computing (IWCMC 2023)*, pages 119–125, Marrakesh, Morocco, June 2023. IEEE. doi: 10.1109/IWCMC58020.2023.10183347.

[17] Pedro V.A. Alves, Mateus A.S.S. Goldbarg, Wysterlânya K.P. Barros, Iago D. Rego, Vinícius J.M.T. Filho, Allan M. Martins, Vicente A. de Sousa Jr., Ramon dos R. Fontes, Eduardo H. da S. Aranha, Augusto V. Neto, and Marcelo A.C. Fernandes. Machine Learning Applied to Anomaly Detection on 5G O-RAN Architecture. *Procedia Computer Science*, 222:81–93, 2023. ISSN 1877-0509. doi: 10.1016/j.procs.2023.08.146.

[18] Claudio Fiandrino, Leonardo Bonati, Salvatore D'Oro, Michele Polese, Tommaso Melodia, and Joerg Widmer. EXPLORA: AI/ML EXPLainability for the Open RAN. *Proceedings of the ACM on Networking*, 1:1–26, November 2023. ISSN 2834-5509. doi: 10.1145/3629141.

[19] Nasir Khan, Sinem Coleri, Asmaa Abdallah, Abdulkadir Celik, and Ahmed M. Eltawil. Explainable and Robust Artificial Intelligence for Trustworthy Resource Management in 6G Networks. *IEEE Communications Magazine*, 62(4):50–56, April 2024. ISSN 0163-6804. doi: 10.1109/mcom.001.2300172.

[20] Chinenye Tassie, Brian Kim, Joshua Groen, Mauro Belgiovine, and Kaushik R. Chowdhury. Leveraging Explainable AI for Reducing Queries of Performance Indicators in Open RAN. In *IEEE International Conference on Communications (ICC 2024)*, pages 5413–5418, Denver, CO, June 2024. IEEE. doi: 10.1109/icc51166.2024.10622827.

[21] Osman Tugay Basaran, Mehmet Basaran, Derya Turan, Hamide Gul Bayrak, and Yagmur Sabucu Sandal. Deep Autoencoder Design for RF Anomaly Detection in 5G O-RAN Near-RT RIC via xApps. In *IEEE International Conference on Communications (ICC 2023), 2nd Workshop on Industrial Private 5G-and-beyond Wireless Networks*, pages 549–555, Rome, Italy, May 2023. IEEE. doi: 10.1109/ICCWorkshops57953.2023.10283501.

[22] O-RAN Alliance. O-RAN Architecture Description. Technical specification, O-RAN Alliance (O-RAN), 06 2024. Version 12.00.

[23] ETSI. O-RAN Fronthaul Control, User and Synchronization Plane Specification. Technical specification (ts), European Telecommunications Standards Institute, 09 2022. Version 07.02.

[24] O-RAN Alliance. O-RAN Use Cases and Requirements. Technical specification, O-RAN Alliance (O-RAN), October 2024. Version 07.00.

[25] O-RAN Alliance. O-RAN A1 interface: General Aspects and Principles. Technical specification, O-RAN Alliance (O-RAN), October 2024. Version 04.00.

[26] O-RAN Alliance. O-RAN O1 Interface Specification. Technical specification, O-RAN Alliance (O-RAN), October 2024. Version 14.00.

[27] O-RAN Alliance. O-RAN SMO Intents-driven Management. Technical specification, O-RAN Alliance (O-RAN), October 2024. Version 03.00.

[28] O-RAN Alliance. O-RAN Near-RT RIC Architecture. Technical specification, O-RAN Alliance (O-RAN), 06 2024. Version 6.00.

[29] Liang Fang, Ruiyuan Song, Zhi Lu, Dongheng Zhang, Yang Hu, Qibin Sun, and Yan Chen. PRISM: Pre-training RF Signals in Sparsity-aware Masked Autoencoders. In *43rd IEEE International Conference on Computer Communications (INFOCOM 2024)*, pages 2109–2118, Vancouver, Canada, May 2024. IEEE. doi: 10.1109/infocom52122.2024.10621246.

[30] Xinyuan Zeng, Chao Wang, Cheng-Cai Wang, and Zan Li. CVCA: A Complex-Valued Classifiable Autoencoder for MmWave Massive MIMO Physical Layer Authentication. In *42nd IEEE International Conference on Computer Communications (INFOCOM 2023), Infocom Workshops 2023 (Workshops)*, pages 1–6, New York City, NY, May 2023. IEEE. ISBN 978-1-66549-427-4. doi: 10.1109/infocomwkshps57453.2023.10225831.

[31] Ye Zeng, Li Qiao, Zhen Gao, Tong Qin, Zhonghuai Wu, Emad Khalaf, Sheng Chen, and Mohsen Guizani. CSI-GPT: Integrating Generative Pre-Trained Transformer With Federated-Tuning to Acquire Downlink Massive MIMO Channels. *IEEE Transactions on Vehicular Technology*, November 2024. ISSN 1939-9359. doi: 10.1109/tvt.2024.3493463. to appear.

[32] Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-*

Supervised Learning. Springer, 2009. ISBN 978-3-031-01548-9. doi: 10.1007/978-3-031-01548-9.

[33] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006. ISSN 1095-9203. doi: 10.1126/science.1127647.

[34] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: explicit invariance during feature extraction. In *28th International Conference on International Conference on Machine Learning (ICML 2011)*, pages 833–840, Bellevue, WA, June 2011. Omnipress. ISBN 978-1-4503-0619-5.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, May 2015.

[36] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86): 2579–2605, 2008.

[37] Md Abul Bashar, Richi Nayak, and Nicolas Suzor. Regularising LSTM classifier by transfer learning for detecting misogynistic tweets with small training set. *Knowledge and Information Systems*, 62(10):4029–4054, June 2020. ISSN 0219-3116. doi: 10.1007/s10115-020-01481-0.

[38] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12 (85):2825–2830, 2011.

[39] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8.

[40] Léon Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In *19th International Conference on Computational Statistics (COMPSTAT 2010)*, pages 177–186, Paris, France, August 2010. Physica-Verlag. ISBN 978-3-7908-2604-3. doi: 10.1007/978-3-7908-2604-3_16.

[41] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *14th International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, pages 315–323, Fort Lauderdale, FL, November 2011. JMLR.org.

[42] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. Anomaly Detection Using Autoencoders in High Performance Computing Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9428–9433, July 2019. ISSN 2159-5399. doi: 10.1609/aaai.v33i01.33019428.

[43] Tao Huang, Pengfei Chen, and Ruipeng Li. A Semi-Supervised VAE Based Active Anomaly Detection Framework in Multivariate Time Series for Online Systems. In *ACM Web Conference 2022*, Lyon, France, April 2022. ACM. doi: 10.1145/3485447.3511984.

[44] Scott M. Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *31st International Conference on Neural Information Processing Systems (NIPS 2017)*, pages 4768–4777, Long Beach, CA, December 2017. Curran Associates Inc.

[45] Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: Real-Time Shapley Value Estimation. In *10th International Conference on Learning Representations (ICLR 2022)*, volume V1, Virtual Conference, April 2022. ICLR.