SpotLight: Accurate, Explainable and Efficient Anomaly Detection for Open RAN

Chuanhao Sun[†], Ujjwal Pawar[†], Molham Khoja[†], Xenofon Foukas[‡], Mahesh K. Marina[†] and Bozidar Radunovic[‡]
The University of Edinburgh[†] Microsoft[‡]

ABSTRACT

The Open RAN architecture, with disaggregated and virtualized RAN functions communicating over standardized interfaces, promises a diversified and multi-vendor RAN ecosystem. However, these same features contribute to increased operational complexity, making it highly challenging to troubleshoot RAN related performance issues and failures. Tackling this challenge requires a dependable, explainable anomaly detection method that Open RAN is currently lacking. To address this problem, we introduce SpotLight, a tailored system archtecture with a distributed deep generative modeling based method running across the edge and cloud. SpotLight takes in a diverse, fine grained stream of metrics from the RAN and the platform, to continually detect and localize anomalies. It introduces a novel multi-stage generative model to detect potential anomalies at the edge using a light-weight algorithm, followed by anomaly confirmation and an explainability phase at the cloud, that helps identify the minimal set of KPIs that caused the anomaly. We evaluate SpotLight using the metrics collected from an enterprise-scale 5G Open RAN deployment in an indoor office building. Our results show that compared to a range of baseline methods, SpotLight yields significant gains in accuracy (13% higher F1 score), explainability ($2.3-4\times$ reduction in the number of reported KPIs) and efficiency ($4-7\times$ bandwidth reduction).

ACM Reference Format:
Chuanhao Sun[†], Ujjwal Pawar[†], Molham Khoja[†], Xenofon Foukas[‡],, Mahesh K. Marina[†] and Bozidar Radunovic[‡]. 2024. SpotLight: Accurate, Explainable and Efficient Anomaly Detection for Open RAN. In *The 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24), November 18–22, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3636534.3649380

ACM MobiCom '24, November 18–22, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-0489-5/24/11…$15.00
https://doi.org/10.1145/3636534.3649380

1 INTRODUCTION

The Open RAN architecture [16, 30, 49, 54] is fast taking root as the blueprint for future radio access networks (RANs) with the backing from governments, regulators and the industry, including focus groups like the O-RAN alliance [2] specifying the reference standards. Disaggregating the traditional monolithic base stations into Centralized (CU), Distributed Unit (DU) and Radio Unit (RU) components communicating over open interfaces promises to diversify the mobile telecoms ecosystem, and offers flexibility for operators to mix and match solutions from different vendors. Decoupling the RAN function and management software from hardware can reduce CAPEX for operators by leveraging virtualization and the cloud. Moreover, the ability to program RAN behavior via RAN intelligent controllers (RICs) and Apps provides new opportunities for efficient RAN operation driven by AI/ML.

Along with the aforementioned benefits and opportunities to increase competition and accelerate innovation, Open RAN introduces new complexities, arising from its disaggregated multi-vendor nature. In particular, two major obstacles must be confronted to clear the path for its widespread adoption. One barrier that received most attention so far is the issue of interoperability of components from multiple vendors. Significant progress has been made on this front to date with numerous PlugFests, trials and interoperability testing/labs (e.g., NTIA 5G Challenge [1], SONIC Labs [4]).

Our focus in this paper is on the other equally significant roadblock: the much higher operational complexity arising from orchestrating and managing components from multiple vendors. This has been well recognized in the current RAN ecosystem [3, 7–11], but has received little attention till date, partly due to the limited number and scale of real-world Open RAN deployments. There are concerns on the performance and robustness of multi-vendor Open RAN deployments and this in turn delays the move away from single-vendor radio networks, considering that a single outage can cost millions of dollars in lost revenue and reputation for operators [3]. In a multi-vendor Open RAN network, even simple and inadvertent misconfigurations can result in degraded performance or downtime, but such issues are hard to diagnose and identify

the component causing the issue [7, 48]. Without effective ways to cope with the increased complexity, OPEX costs are expected to rise [8, 10, 11], negating the CAPEX cost savings.

Overcoming the operational complexity roadblock hinges on the ability to continually monitor the Open RAN system as a whole, to automatically detect and localize anomalies; a capability that is lacking today. Introducing this capability entails addressing multiple challenges. First, to enable detection of system anomalies, we need rich monitoring data streams covering a wide range of key performance indicators (KPIs) across RAN components as well as the underlying platform, in a way that provides a unified view of the system state at any time instant. Second, we need a method that can not only detect any and all anomalies but also minimizes false alarms. Third, the method should reliably pinpoint the root cause of an anomaly, i.e., it should come with explainability. Fourth, such an explainable anomaly detection method should be efficient and lend itself to easier deployment, operating within computational constraints at the RAN edge sites, with minimal bandwidth consumption. To the best of our knowledge, even the basic Open RAN anomaly detection problem has not been addressed to date, let alone with the above mentioned scope we target.

Motivated by the above, we introduce SpotLight, a system architecture and explainable anomaly detection method for Open RAN. The core of SpotLight is a novel distributed anomaly detection method (illustrated in Figure 3) based on deep generative modeling [52]. For effective anomaly detection, SpotLight uses two custom-designed deep generative models – JVGAN and MRPI – in sequence that respectively perform distribution learning and time-series imputation. The first generator, JVGAN, that runs at the edge learns the 'distribution of the normal KPI time series data' during training and uses that as a reference to reliably infer any 'potential' anomalies in the observed KPI time series data (or equivalently to filter out likely normal cases). The result of JVGAN is further inspected by our second detection model running in the cloud, MRPI, to minimize false alarms while detecting all anomalies. MRPI, also trained with normal data, achieves this by treating the outlier portions of anomalies highlighted by JVGAN as missing data in a time series and relies on the inability to generate them with MRPI, the trained imputation model, as a cue to inferring anomalous data points.

For reliable explaination of anomalies when they are detected, i.e., for fine-grained root cause identification, SpotLight pipeline incorporates two further steps. The first KFilter step takes in anomalous KPIs highlighted by MRPI as input and retains only persistently anomalous KPIs. The final causal discovery step leverages CausalNex [18] to select a subset of anomalous KPIs obtained from KFilter that have a causal effect on the rest of those KPIs. The initial detection with JVGAN is key to achieving efficiency as it is not only very lightweight to fit within the limited computational resources available at RAN edge sites but it also filters out KPI data streams exhibiting normal behavior. As a result, bandwidth and processing requirements for further anomaly detection and identification in the cloud (via MRPI, KFilter and CausalNex) are significantly reduced.

As part of SpotLight's system design, we develop a data collection process, that allows us to collect a detailed set of KPIs (>600 KPIs) spanning both the radio network and platform dimensions, i.e., both 3GPP and OS/network related parameters. We collect the KPIs at a fine time granularity of 100ms that allows us to accurately detect and pinpoint a wide range of anomalies. We leverage a state-of-the-art enterprise-scale 5G Open RAN deployment [17] to create, to our knowledge, the largest and most realistic multi-UE Open RAN dataset till date, comprising more than 100 million datapoints.

We evaluate SpotLight by considering several realistic anomalies (e.g., CPU contention, network contention and radio interference), both separately and in conjunction. We also evaluate with several real-world anomalies we detected while operating our deployment for over a year. Our results demonstrate that SpotLight can detect and localize all introduced anomalies with very high accuracy compared to existing approaches, while also being very efficient in terms of the computational and networking overhead.

In summary, we make the following key contributions:

- For the first time, we draw attention to the problem of anomaly detection and localization in the Open RAN context, highlight its uniqueness and the new challenges (§2).
- We introduce the SpotLight system architecture and method design to resolve the above problem. SpotLight employs a novel distributed multi-stage anomaly detection and identification pipeline. It features a new approach to time series anomaly detection combining distribution learning and time-series imputation, realized through a pair of custom-designed deep generative models distributed across edge and cloud for accurate and efficient anomaly detection in Open RAN (§3).
- We develop a detailed and holistic Open RAN data collection process spanning both the RAN and platform dimensions, and we create, to our knowledge, the largest and most realistic multi-UE Open RAN dataset to date (§4). We make our dataset publicly available[1] to support Open RAN related efforts in the research community.
- We evaluate SpotLight on a realistic 5G RAN deployment and demonstrate its accuracy and explainability benefits compared to existing solutions over synthetic anomalies, as well as its ability to detect and localize real world anomalies during normal RAN operation (§5-§7).

[1] https://github.com/netsys-edinburgh/SpotLight

2 BACKGROUND AND MOTIVATION

2.1 Open RAN Architecture

Traditional RAN deployments are typically developed as embedded systems, where the hardware and software components are built by a single vendor and are tightly integrated. Open RAN is an industry transformation, similar to SDN, that seeks to decouple RAN software (SW) from hardware (HW), further disaggregate the SW, and have open interfaces between all different components. It allows independent evolution of hardware and software, and faster rollout of new services. It also enables operators to mix and match their components from different vendors, thus help diversify the ecosystem.

As shown in Figure 1, an Open RAN base-station consists of several components. One is a radio unit (RU), deployed at a cell tower. A virtualized distributed unit (vDU) serves several RUs and performs latency-critical operations, such as signal processing and radio resource scheduling. It runs on commodity servers, optimized for low latency (e.g. Linux with real-time kernel patches). Due to stringent latency requirements, it is deployed at a far-edge site, within a few kms from the cell towers. Several vDUs connect to a virtualized centralized unit (vCU), which often runs at a near-edge site, further away from the towers, since it has more relaxed latency requirements. Note that for cost and power efficiency reasons, vRAN deployments severely limit tasks other than RAN function processing at edge sites. A typical large telco may have 10,000s of far-edge and 100s of near-edge sites.

In contrast to conventional RANs, Open RAN deployments are composed of several multi-vendor hardware and software components and are bundled into a single solution by third-parties (e.g., system integrators). For example, a vDU and vCU from one vendor can run on server hardware from another vendor and the platform software can come from a third vendor, as illustrated in Figure 1. A key goal of Open RAN is to standardize different interfaces in the architecture to enable such multi-vendor deployments. Entities like a service management and orchestration framework (SMO) and radio intelligent controllers (RICs) allow operators to control various aspects of RAN deployments, such as switching off RUs for power saving and optimizing handovers between cells.

2.2 The Open RAN Management Challenge

The main challenge in Open RAN is how to manage the interaction among different disparate components, together with their ever present software updates. This is a novel challenge that did not exist in conventional, tightly integrated RANs provided by a single vendor. It becomes even harder in the case of a vDU, due to its latency sensitivity. The vDU is characterized by sub-millisecond processing deadlines. Packets are dropped if a deadline is violated, affecting the performance and the reliability of the RAN [27]. Commodity hardware and software has not been designed to enforce such a low latency. Relatively

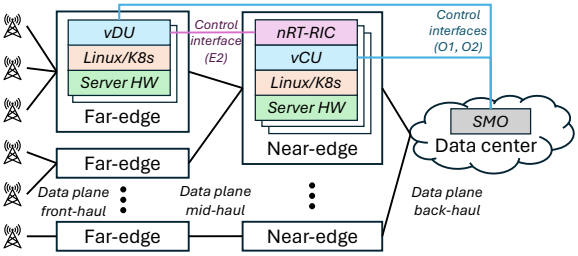


Figure 1: Open RAN defines various open interfaces and consists of hardware and software components provided by diverse vendors (marked in different colors). nRT-RIC: near realtime RIC.

recent Linux developments, such as real-time kernel patches, isolated CPU cores and DPDK, provide the required capability to meet the RAN real-time requirements, but special care needs to be taken to configure it and detect any problems.

There are many potential sources of problems, both at the RAN and platform level. For example, misconfigurations or bugs (introduced by operations people when assigning cores in configuration files) can make some threads to be incorrectly migrated to and run on cores that are meant to run only real-time RAN processing workload, thereby causing CPU contention and violating the RAN runtime deadlines. Similar problems could also arise from regressions of software updates (cf. [13, 38, 51] for OS related regressions and [37] for an orchestration layer regression). Besides, a wrongly configured MAC address can cause unintended contention on the RAN Fronthaul (FH) and performance degradation. External wireless interference can also cause a RAN performance degradation. In many cases, such degradations look similar and it is hard to blame a specific component and its vendor, leading to operators being reluctant to adopt Open RAN due to lack of a "single neck to choke" [40].

We now highlight the difficulty of troubleshooting problems in Open RAN, through an example realized using an enterprise-scale 5G RAN deployment (see §5.1 for details). We consider a cell with a single user receiving TCP traffic. We collect and plot the throughput of the vDU Ethernet port used for the fronthaul traffic, the aggregate runtime of all threads collocated on the same CPU core as the FH traffic thread, as well as the SNR and the downlink (DL) TCP throughput of the user device. In the normal case, shown in Figure 2(a), we can observe that the user receives data at a constant rate of 17Mbps, the SNR value is stable, the fronthaul network link has constant rate traffic of 5.15Gbps, and the runtime of the collocated threads is less than 100ms per second.

Next, we introduce three different types of anomalies: i) wireless interference from an external transmitter (Figure 2(b)), ii) CPU contention on the CPU core of the fronthaul traffic processing thread (Figure 2(c)), and iii) network contention on the vDU FH interface, by introducing an extra flow with 2.2 Gbps of traffic (Figure 2(d)). By using the results of the above

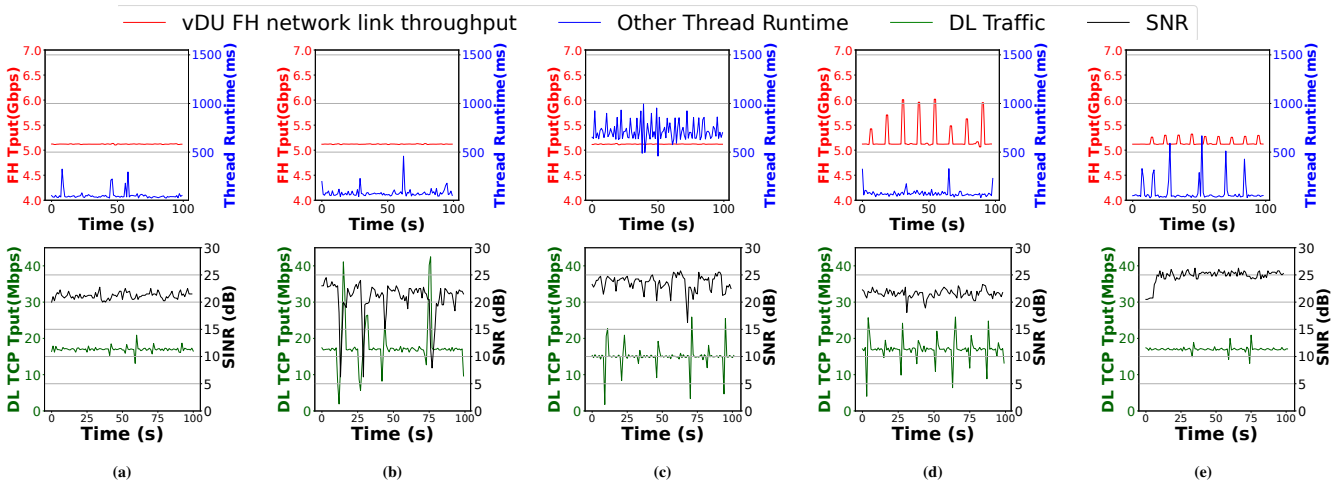


Figure 2: FH link throughput and aggregate runtime of FH related collocated threads (top); DL TCP throughput and SINR at the UE (bottom) for different cases. KPIs for downlink (DL) TCP traffic in (a) normal operation; (b) with external radio interference; (c) with CPU contention on FH thread; (d) with FH network contention; (e) with benign FH network contention.

experiment, we highlight the following key observations about the challenges of anomaly detection in the OpenRAN context.

Different anomalies may have similar effects on the KPIs. Simply by observing individual KPIs can be misleading in identifying the root cause of a problem. For example, in Figure 2(b), we observe that the SNR fluctuates, which one might correctly view as an indicator of radio interference. However, the same observation would be wrong in the case of Figure 2(c). Even though SNR fluctuates in this case too, it is not a root cause, but rather a side-effect of the CPU contention applied to the fronthaul thread, which makes the vDU to delay (and ultimately drop) some of the IQ samples coming from the radio. The difference in the two cases, only becomes clear once more KPIs, like the runtime of the threads collocated with the FH thread are also taken into account.

Telling apart normal from anomalous behavior is not easy. Just because some KPIs might be deviating from a well-known pattern, does not mean that an anomaly is actually present. For example, consider the FH network link contention case of Figure 2(d). The packets of the contending flow end up delaying the fronthaul packets carrying the IQ samples, which miss their deadlines and have to be dropped. This leads to a noticeable degradation of the user TCP thoughput. For this anomalous scenario, and in contrast to the baseline case of Figure 2(a), we can see that the FH network link throughput fluctuates, due to the contending flow. This could lead someone to the conclusion that a deviation from a constant rate in the fronthaul network link is the correct indicator to identify this anomaly. However, that would be wrong, because fluctuations in the fronthaul network link are only anomalous, if they lead to deadline violations. As a counter example, consider the scenario of Figure 2(e), in which we introduce a traffic flow in the fronthaul link, which is configured to have a lower priority than the fronthaul traffic. In this case, while the fronthaul link throughput fluctuates, it does so in a controlled manner, without affecting the RAN performance.

The challenge of scale and granularity. The above example demonstrate that troubleshooting is not straightforward, even for a domain expert. In practice, the problem is even more challenging, due to the large number of KPIs. For instance, in the RAN that we used for our evaluation, a single cell has more than thirty CPU threads and five network interfaces. From each of those sources, we could extract several useful KPIs. If we combine the RAN and platform KPIs, we get several hundred time series that one would have to monitor and correlate, making anomaly detection a very challenging task. To add to this challenge, current standard RAN and O-RAN KPIs [12, 44] define only high-level aggregate metrics that were not designed to troubleshoot subtle RAN and platform interaction issues like the ones discussed here.

2.3 Limitations of Prior Work

Even assuming that the right instrumentation is in place to enable troubleshooting of operational Open RAN problems, existing anomaly detection approaches are insufficient. While, to our knowledge, there is no existing work focusing on anomaly detection for the Open RAN setting, there have been several studies on anomaly detection for traditional RANs. Klaine et al. [36] survey early work on 3G/4G RAN anomaly detection. The focus of these works was on specific use cases of fault detection, fault classification and cell outage management, relying on a small number of relevant KPIs corresponding to each. Recent RAN related anomaly detection works also share this same characteristic of focusing on particular causes (anomalies) and corresponding KPIs. Examples include: radio

interference detection, mainly based on using uplink RSSI, cell-level traffic volumes [34]; traffic spike detection using cellular control channel measurement data [56]; and detecting end-to-end performance drops using TCP loss ratio and round trip time [14]. While the explainability is not an issue with these works due to the focus on one cause, as we show in §6.1, this myopic anomaly detection approach considering only a few KPIs is ineffective to reliably detect the diverse set of anomalies in our target Open RAN setting.

From a method design perspective, the problem we target is essentially (multivariate) time series anomaly detection [19]. In the RAN context, prior work (e.g., [34]) has shown that commonly used non time series anomaly detection methods (e.g., Z-Score, robust covariance, one-class SVM) [6, 36], and supervised binary classification based anomaly detection, as considered in early works (e.g., [33]), are ineffective. Consequently, the state-of-the-art methods for RAN anomaly detection broadly fall under two classes: (i) time series prediction with recurrent neural networks (e.g., LSTM) [22, 34, 56, 59]; (ii) reconstruction based with autoencoders [34, 43, 56]. Both these type of methods are limited by the unwieldy challenge of having to determine a right threshold for prediction/reconstruction errors, especially when there are many KPIs and with highly stochastic nature, as our results in §6.1 demonstrate.

Explainability or root cause analysis has been considered in some prior works on anomaly detection in traditional RANs [22, 50, 59]. A common approach is to augment an anomaly detection method with SHAP (SHapley Additive exPlanations) [42] or similar model-agnostic explainers, for identifying important features/KPIs responsible for the detection of anomalies [22, 59]. Interpretable shallow ML models such as decision trees have also been used [50]. Explainability of AI models is starting to be recognized as an important requirement in the Open RAN context [20] but we are unaware of any existing work on explainable anomaly detection for this context.

More generally speaking, time series anomaly detection is an active area of research in the machine learning domain [19, 23, 53]. As highlighted in a latest survey [23], deep learning based time series anomaly detection methods outperform the traditional alternatives, and deep learning based methods themselves can be classified into forecasting (time-series prediction) based (e.g., GDN [24]), reconstruction based (e.g., TranAD [57]) or a combination of both (e.g., VAE-LSTM [41]). This is in line with what is noted above about most recent methods for anomaly detection in traditional RANs. Also, with the exception of a few methods like GDN [24], most existing time series anomaly detection methods lack explainability. Overall, as we show in §6.1, the existing time series anomaly methods have poor precision (high false alarms) and therefore yield poor detection when applied to our Open RAN setting. This aligns with observations from prior work that most existing time series anomaly detection methods fail to distinguish between normal and anomalous behavior when faced with irregular and stochastic features [53].

2.4 Key Takeaways

- Operational problems (anomalies) in an Open RAN system can occur anywhere across components from different vendors. These problems not only include traditional RAN anomalies such as external interference but also arise from software upgrades and misconfigurations involving complex interactions between RAN and the platform.
- These anomalies are hard to detect and pinpoint because: (1) telling apart normal and anomalous behavior is not easy, unless multiple KPIs are considered and correlated; (2) different anomalies may have similar effect on the KPIs.
- Identifying the whole spectrum of potential problems requires fine-grained measurements of many (hundreds of) KPIs throughout the system along the RAN and platform dimensions, way beyond the aggregate and RAN only KPIs available with standard 3GPP RAN and O-RAN.
- Manually inspecting KPIs by experts to analyze and correlate hundreds of KPIs to infer anomalies and their root causes is simply not scalable, and so an automated process is essential.
- With many potential anomalies and hundreds of KPIs to consider, existing multivariate time series anomaly detection methods are unreliable and result in high number of false alarms when used in the Open RAN context. The fact that they depend on setting thresholds for prediction/reconstruction errors is a key reason why they are ineffective.
- Even from an explainability perspective, existing approaches, which rely either on augmenting anomaly detection methods with model agnostic explainers like SHAP or the few methods that have a builtin explainability feature (e.g., GDN), flag up too many and misleading KPIs to be useful as reliable guides to pinpoint anomalies when they occur.
- From a deployability perspective, reliable anomaly detection and localization alone is insufficient for a method. It should additionally be efficient to stay within the computational and bandwidth resource constraints at the RAN sites (§2.1).

3 SPOTLIGHT DESIGN

3.1 System Architecture

SpotLight is a system for detecting and explaining anomalies across RAN and platform components in Open RAN. It is built on the observations that (i) anomaly detection needs to be automated to the extent possible, and (ii) to build this automation we need to collect detailed metrics from both RAN and platform. SpotLight is powered by a new custom multi-stage deep learning model pipeline, distributed across edge and cloud, to reliably detect anomalies and guide towards root

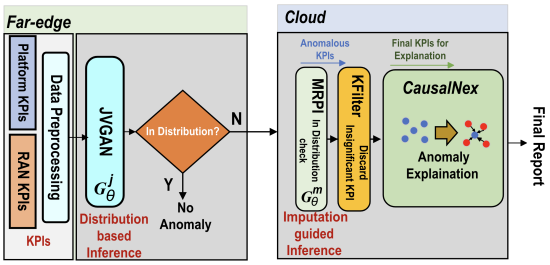


Figure 3: Schematic of SpotLight system architecture. OOD: Out of Distribution.

causes. Its high-level architecture (Figure 3), consists of three parts:

- **Data collection.** Unlike conventional RAN systems, SpotLight introduces detailed instrumentation on both the RAN and platform. We introduce probes that collect fine-grained metrics or KPIs (over 600 in total) that provide required data to enable reliable anomaly detection and root cause identification (see §4).
- **Data processing at the edge.** A far-edge node has limited compute capacity and is not suited for sophisticated ML methods. A centralized location (e.g., cloud) is better equipped for this task, however, sending massive amounts of data from 10,000s of far-edge sites can be prohibitively expensive. So we use the edge only for lightweight yet robust filtering of normal cases (that constitute the majority) to fit within the available compute as well as minimize the uplink bandwidth.
- **Anomaly detection and identification in the cloud.** We deploy the heavy part of our processing in the cloud due to ample availability of compute and storage resources.

We measure samples for each KPI every 100ms. We collect 64 consecutive samples into a window W, as part of a multi-variate time series, and feed it to SpotLight's detection pipeline. As the output, in the event of an anomaly, we receive a filtered set of anomalous KPIs related to specific RAN and platform components that help identify the root cause.

3.2 Method Description

3.2.1 Design Requirements.

- ***Accuracy.*** Our goal is to to maximize the detection of all anomalies (i.e., have a *high recall*) as well as minimize spurious detections or false alarms (i.e., have a *high precision*). To detect any kind of anomaly, we further require our method to be semi-supervised in that it should be trained only using normal data [21, 23], and a limited amount of it. The method should also *generalize well* to new and unseen KPI patterns, including normal cases not in the training data.
- ***Explainability.*** We need an anomaly detection method that reliably directs us to the minimal subset of KPIs that point to the source of an anomaly. For example, an anomaly in a physical (L1) layer of vRAN can be visible through unexpected uplink traffic variations in L1. But this will also cause variations in uplink traffic in MAC, RLC and PDCP layers, and we will potentially mark these metrics anomalous as well. As such, the method should filter KPIs to help find the right root cause through a minimal set of relevant KPIs, by learning and tracking dependencies between them.
- ***Efficiency.*** We seek a method that makes efficient use of limited available local processing resources at the far-edge sites while also minimizing the bandwidth and cost requirements for further processing in the cloud, all while ensuring anomaly detection and identification within the desired timescales.

3.2.2 Design Overview. Formally, the anomaly detection and identification problem we target takes as input the multivariate time series of KPIs = $(x^i(t))_i$, for each KPI i. This includes measured as well as derived KPIs, and span both radio network and platform (§4). Resolving our problem translates to outputting $\emptyset$ if no anomaly is found, and a minimal subset $\mathbb{K} \subseteq \mathbf{K}$ otherwise, where $\mathbf{K}$ is set of *all* KPIs. In the anomaly case, the subset of KPIs $\mathbb{K}$ in the output reflects the likely cause and location of the anomaly, given that each KPI implicitly represents a location in the system. We continually perform the detection and identification for every incoming time window W of KPI streams.

To address this problem in a way that meets the aforementioned requirements, existing methods, that are either time-series prediction or reconstruction based [23], are ineffective when applied to our setting, as comprehensively shown in §6. We therefore introduce a custom-tailored anomaly detection and identification pipeline that takes a fundamentally different approach. The SpotLight pipeline consists of 4 stages (Figure 3).The first two stages are aimed at anomaly detection and filtering at the window level. In the event of an anomaly, the latter two stages help explain the underlying root cause by filtering at the KPI level.

For anomaly detection, our approach uses a combination of distribution learning and time-series imputation, to achieve both accuracy and efficiency. To our knowledge, neither of these ideas has been used previously for time-series anomaly detection, individually or together. In the first stage, JVGAN (Figure 3), runs at the edge to filter out likely normal cases (or equivalently, detects if there is a 'potential' anomaly) using the distribution of normal data learned during training with our new variational autoencoder (VAE) based deep generative model. As we show in §6.4, JVGAN is lightweight (<0.1% of a CPU core) and, as such, well suited for deployment at the edge. It also significantly reduces upstream bandwidth consumption (4×-7×), in line with the observation that normal cases typically make up the majority. The following MRPI stage in the cloud is a deep generative model we developed for time-series imputation, that seeks to minimize false detections, thereby achieve high precision. The key idea is that, if we

treat the outlier points from the JVGAN stage as missing points and cannot generate them with a time-series imputation model (trained on normal data), then the outliers represent true anomalies, otherwise false alarms. This step ensures a high precision very close to 1 (see §6.1).

To address the explainability requirement, when an anomaly is detected by JVGAN and MRPI, the KFilter and CausalNex stages (Figure 3), also running in the cloud, filter down the list of anomalous KPIs to the minimal set, to help identify the root cause. The KFilter stage is a mechanism to trim down the inspected KPIs, by only retaining *persistently* anomalous KPIs in a given time window. Then CausalNex [18] stage leverages a state-of-the-art causal discovery method, DYNOTEARS [47], to learn the causal relationship among the remaining KPIs from the KFilter stage to pinpoint to the root cause KPIs. Note that, although CausalNex is an existing causal reasoning library we incorporate in our pipeline, our contribution lies in strategically using it at the right point to aid in explainability. Only because of whittling down the KPIs through the KFilter stage, the use of CausalNex becomes viable, as its computational cost exponentially grows with the number of variables (KPIs in our case). Running time of DYNOTEARS increases from the order of seconds with 5-10 variables to 10s to 100s of minutes with 50-100 variables [46]. Experimentally, we find that KFilter reduces the number of KPIs passed to CausalNex by 90%, thereby playing a crucial role in enabling real-time anomaly identification. We next elaborate on the different stages of the SpotLight pipeline.

*3.2.3 Distribution based inference with **JVGAN**.* JVGAN, the initial stage of our pipeline, is a generator that first learns the distribution of normal KPI time series and then uses the learned distribution as a reference to infer if a given test KPI time series is anomalous. This distribution learning approach is not only robust to highly diverse and stochastic patterns across many KPIs, as in our setting, but also does not have the threshold setting issue as prior methods. Furthermore, it has the beneficial effect of data filtering at the edge by efficiently distinguishing clearly normal cases from potentially anomalous cases so that only the latter cases are further inspected.

Specifically, for JVGAN we train a generator G^j_θ on KPI data streams $\mathbf{x}(t) \subset \mathbb{X}(t)$, where θ and $\mathbb{X}(t)$ respectively refer to the learned weights of the generator model and the training data. The objective of the generator G^j_θ is to help infer during operational phase if a given measured KPI time series $\mathbf{x}(t) \in \mathbf{X}(t)$, where $\mathbf{X}(t)$ is the true but unknown distribution of the 'normal' KPI time series. For this, the trained generator $G^j_\theta(\mathbf{x}(t))$ is sampled N times (empirically set to 150 in this work) to get a set of samples of a learned distribution $\mathbb{J}(t)$ that approximates $\mathbf{X}(t)$. Then, given a 'test' KPI time series $\mathbf{y}(t)$, we check to see if it falls within the learned distribution $\mathbb{J}(t)$. We use the upper and lower bounds of $\mathbb{J}(t)$, represented respectively as $a(t)$ and $b(t)$, as the envelope of samples drawn from the distribution $\mathbb{J}(t)$, and we declare observed KPI data stream $\mathbf{y}(t)$ as anomalous if

$$\exists y_j \in y(t),\ s.t.\ y_j < b_j \text{ or } y_j > a_j \tag{1}$$

This is illustrated in Figure 4(c).

JVGAN generator $G^j_\theta(\mathbf{x}(t))$ is based on the variational autoencoder (VAE) [35], which is a standard approach for distribution learning. However, as we show in §6.1, vanilla VAE is ineffective for this purpose due to the following three issues:

(1) Classical VAE works well only when dealing with continuous data but there exist many KPIs in our setting that are discrete or categorical (e.g., HARQ outcome).
(2) Our KPI time series is highly bursty, which makes it harder for vanilla VAE to learn the distribution in a way that can reliably separate normal and anomalous cases.
(3) Fitting the learned distribution too closely to training data hurts generalization, as training data does not represent all possible normal cases since $\mathbb{X}(t) \subset \mathbf{X}(t)$.

We address issue (1) in $G^j_\theta(\mathbf{x}(t))$ by considering JointVAE [25] as our basic neural network structure, as it is more robust with categorical and binary data streams. For issue (2), we include adversarial training (à la GANs [31]) for high fidelity distribution learning. For issue (3), we use Monte Carlo (MC) Dropout [29] to have the learned distribution to be not limited by training data, which has not been explored before in the anomaly detection context. In particular, we do this by using MC dropout technique to estimate the inherent uncertainty in the generator for modeling the distribution of normal data. This is then used in setting $a(t)$ and $b(t)$ bounds above to additionally account for model's uncertainty, and not simply use max and min from the samples as the bounds. As shown in §6.1, this approach significantly improves the detection performance with JVGAN. More generally, learning the distribution of low dimensional representation of high dimensional data streams and using that for anomaly detection as we do here is new.

The JVGAN generator architecture ($G^j_\theta(\mathbf{x}(t))$) with the above techniques is illustrated in Fig. 4(a). The input to JVGAN is the set of KPI streams across RAN and platform components every time window. Before inputting them to JVGAN, few preprocessing steps are carried out: (1) any missing data in a given window are filled using nearest-neighbor imputation; (2) values in each stream are normalized to 0-1 scale; (3) all streams are aligned in time. Note that all KPI streams are fed into the SpotLight pipeline, as it is not known a priori if an anomaly is present and of what kind; the set of KPIs get automatically whittled down as a result of processing through the pipeline. Moreover, as the source of anomaly can be anywhere in the system, all KPIs are treated equally.

*3.2.4 Imputation guided Inference with **MRPI**.* JVGAN can reliably detect normal cases when a given test KPI time series

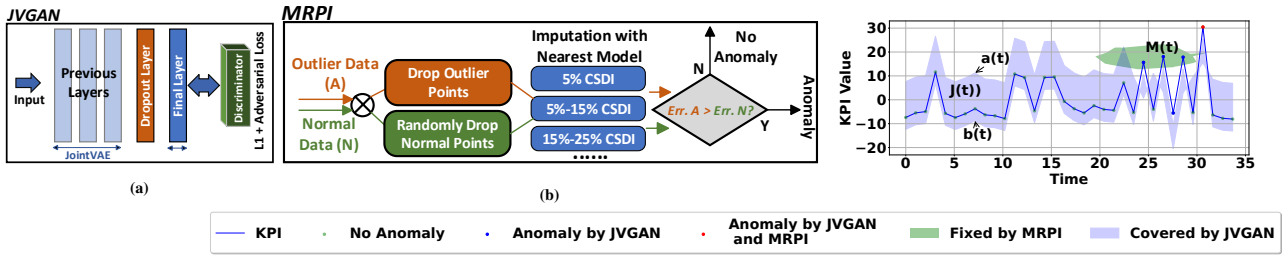


Figure 4: (a) JVGAN architecture; (b) MRPI workflow; (c) Illustration of learned distributions: $\mathbb{J}(t)$ in blue; $\mathbb{M}(t)$ in green.

is fully within $\mathbb{J}(t)$. Considering that $\mathbb{J}(t)$ is only an approximation of the unknown true distribution of the normal KPI time series, $\mathbf{X}(t)$, and that it is limited by the training data $\mathbb{X}(t)$, we have many cases that only fall partially in $\mathbb{J}(t)$. Fig. 4(c) illustrates such a case where some of the points fall outside the distribution $\mathbb{J}(t)$. However, simply inferring all such cases as anomalies will lead to poor precision (i.e., many false alarms). So, we introduce a further vetting step through another generator called multiple rate probabilistic imputation (MRPI) to minimize spurious anomalies. The key idea behind MRPI is as follows. The points that fall outside $\mathbb{J}(t)$ are treated as 'missing points', and then we assess if they can be generated by an imputation model trained using only normal data $\mathbb{X}(t)$. Among those missing points, the ones that cannot be reliably generated through imputation, i.e., fall outside the distribution $\mathbb{M}(t)$ learnt by the imputation model, can be safely inferred as anomalous. The use of time-series imputation as we do with MRPI is new in the anomaly detection context.

Our MRPI design is based on CSDI [55], which is the best existing time series imputation model [15]. However, the original CSDI design is limited to just one setting of missing data rate. On the other hand, it is impractical to have a separate imputation model for each possible missing data rate. We empirically observe that imputation at 10% granularity of rates provides good generalizability for any rate of missing data. Therefore, we train multiple CSDI models for different intervals (ranges of missing data rates) using normal training data $\mathbb{X}(t)$: $\leq 5\%, 5-15\%, 15-25\%$, .. (11 models in total). Fig. 4(b) illustrates MRPI design. During inference, we pick the model that is closest, based on fraction of points corresponding to $\mathbf{x}(t)$ that are not covered by the learned distribution $\mathbb{J}(t)$. For example, if 10% of $\mathbf{x}(t)$ is not in $\mathbb{J}(t)$, then we pick the trained $5-15\%$ CSDI model. We declare a point as an anomaly, if it does not belong to the distribution of the selected CSDI model. We illustrate this in Fig. 4(c) where the CSDI distribution is shown in green.

*3.2.5 Explainability with **KFilter** and **Causal Discovery**.* By applying the combination of JVGAN and MRPI, we can continually detect any KPIs exhibiting anomalous behavior based on the most recent window of KPI data streams. However, the anomalous nature of some of these KPIs may be transient while other anomalous KPIs might be the effect of an anomaly caused elsewhere. So, to better identify the actual root causes of persistent anomalies, we employ two methods – KFilter and Causal Discovery – as elaborated below.

KFilter. The purpose of this method is to filter out insignificant anomalous KPIs. In particular, the aim is to discard those KPIs that are detected as anomalous only for a brief period of time but otherwise show normal behavior. So, we monitor the percentage of time each KPI is detected to be anomalous across each measurement window (which in this work is set to $6.4s = 64 \times 100ms$) and filter out the ones which appear anomalous below a certain threshold period. We empirically set that threshold to 25% in our experiments.

Causal Discovery. Even after applying the KFilter, there may be several anomalous KPIs left for a domain expert to examine to identify the root cause behind a detected anomaly. We observe that KPIs have inherent correlations and causal relationships between them. This suggests that the actual KPIs to inspect in the event of an anomaly are the subset of anomalous KPIs that have a causal relation from them to other anomalous KPIs. So, in the event of an anomaly, we aim to use the directed graph of causal relations among KPIs to reduce the ones reported by SpotLight. To deduce the causal graph among anomalous KPIs after KFilter, we make use of CausalNex [18], the state-of-the-art toolkit for causal reasoning with Bayesian Networks. Specifically, each anomalous KPI is represented as a node in this graph and we report the ones that have directed edges *from* them to other anomalous KPIs.

4 DATA COLLECTION

Here we describe the data collection process behind SpotLight. A brief summary of all the collected KPIs, statistics and the data collection process is listed in Table 1, while a detailed description of the dataset's schema can be found in [5].

Radio Network KPIs: For the collection of RAN KPIs, we leveraged Janus [28], which is a telemetry extraction framework that is integrated in several commercial-grade RAN functions, including the vCU and vDU used for our experiments (see §5). Janus introduces RAN function hooks to expose raw data in real-time. The hooks are used to inject statically verifiable codelets, to programmatically process and extract only relevant data, in a safe and lightweight manner. Using this framework, we developed 16 codelets, each with approximately 300 lines of C code, which allowed us to capture

KPI Category	Description	Type
Signal Quality	UL/DL SINR, CSI reports, UL/DL MCS, etc.	Radio
Packet Size	UL/DL packet size at PDCP, Midhaul, RLC, MAC and FAPI layers	Radio
Buffer Info	UL buffer status report and DL buffer occupancy	Radio
Resource Allocation	UL/DL PRB usage and UL/DL TBS	Radio
Losses	UL BLER and DL HARQ NACK rate	Radio
FH Traffic	FH UL/DL link usage in Gbps	Platform
Thread Scheduling	On and off CPU runtimes of threads	Platform
PTP Logs	PTP frequency, RMS, delay and max offset	Platform

Table 1: Summary of KPIs collected.

data from many vCU and vDU events, such as signal quality reports of UEs, MAC scheduling decisions, queue and packet sizes at the PDCP and RLC layers, HARQ ACKs/NACKs, etc. Using the raw data, we derived several KPIs at the granularity of 100ms, including histograms and other statistics (min, max, distribution skewness, etc.), for a total of 206 radio network KPIs, all agnostic of the number of UEs.

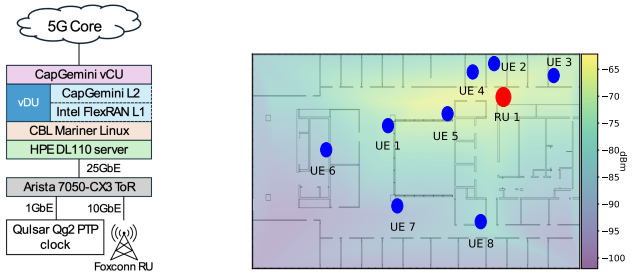
Platform KPIs: We collected network counters (number of packets and throughput) exposed by a gRPC network management interface (gNMI) API of the ToR switch at a granularity of 100ms. We developed an eBPF tool [26], to collect detailed per-CPU runtime information from all the threads running in the system (RAN and others), by capturing all the scheduling events. The tool exposes both on- and off-cpu runtimes of threads, i.e., how long a thread ran before being pre-empted (on-cpu) and how long it waited until it was scheduled again (off-cpu). As these events can reach hundreds of thousands per second, we aggregated the thread runtime statistics at a granularity of 100ms to guarantee the stability of the system, in a similar way as we did for the radio network KPIs (histograms, skewness, etc.). We also scraped OS logs to collect data about PTP synchronization (100ms granularity). This resulted in the collection of 466 platform KPIs overall.

5 EVALUATION METHODOLOGY

5.1 Evaluation setup

We have deployed a state-of-the-art, enterprise-scale 5G Open RAN deployment, covering a five floor office building in Cambridge, UK, with two 5G base stations per floor [17]. All the components are commercial-grade and O-RAN compliant. Table 2 summarizes the hardware and software configuration. The vRAN functions support several 5G features, including 4×4 MIMO and the O-RAN 7.2x FH protocol. To our knowledge, this combination of features in a standards-compliant and end-to-end 5G Open RAN setup is a unique characteristic of our deployment. We operated the deployment for over a year, and in that period we identified a number of real-world cases of anomalies (discussed in Section 7.1).

For the evaluation, we focus on a single cell, using the configuration illustrated in Fig. 5(a). In terms of the end-user devices, we use up to 8 UEs during the network's normal operation, including both commercial 5G smartphones (OnePlus N10, Samsung Galaxy A52s) and Raspberry Pi development kits equipped with Qualcomm 5G modems.



(a) (b)

Figure 5: (a) Cell deployment configuration; (b) Floorplan of cell and UE deployment used for data collection and evaluation.

5.2 Dataset Creation

Using the deployment described in §5.1, we created a rich dataset of fine-grained platform and radio network KPIs described in §4, considering both non-anomalous and anomalous cases. Our training dataset consists of approximately 77 million measurement points, and, to our knowledge, it is the largest and most realistic Open RAN dataset to date.

To generate a realistic dataset, we introduced a diverse set of eight traffic profiles, summarized in Table 3, and considered scenarios with one, five and eight UEs. In experiments involving a single UE, we generated traffic with all profiles for a duration of 10 minutes each. For five and eight UEs, we selected random UE subsets and created two scenario types:

- Constant traffic: Each UE selects a traffic profile and maintains it for 10 minutes.
- Mixed Traffic: Each UE repeats a cycle for 10 minutes, where it randomly selects one of the 8 traffic profiles for a random duration (5 to 10 seconds) and then pauses for a random duration (5 to 10 seconds).

Obtaining reproducible data in a controlled manner is challenging in a multi-UE setting with mobility. Therefore, in this work, we opt to use static UEs, meaning that some KPIs, such as SNR, are more static than in a truly mobile environment. We compensate for this in two ways:

- We consider several UEs with diverse positions and varying distances from the radio (showed in Fig. 5(b)), ensuring that these KPIs differ across different UEs.
- All the KPIs of our dataset are aggregates across all UEs, rather than capturing individual UE metrics.

Servers (8×)	HPE Telco DL110 Gen 10; Xeon 6338N CPU
Accelerator	Intel ACC100 for LDPC coding
Ethernet NIC	Intel E810 4×25GbE NIC
Ethernet switch	Arista 7050-CX3
Radio unit	Foxconn 4x4 RU; 100MHz at 3.5GHz
PTP grandmaster	Qulsar Qg2 multi-sync gateway
Operating system	Real-time CBL-Mariner Linux kernel 5.15
vRAN software	Intel FlexRAN v22.03 (L1) & CapGemini 5G (L2+)

Table 2: 5G RAN hardware and software configuration.

Traffic Type	Description
iperf3 TCP DL	UE sends TCP DL traffic in DL direction
iperf3 TCP UL	UE sends TCP UL traffic in UL direction
iperf3 UDP DL	UE sends UDP traffic at 10 Mbps in DL direction
iperf3 UDP UL	UE sends UDP UL traffic at 10 Mbps in UL direction
file download	UE downloads a file from Internet
file upload	UE uploads a file to server using scp
video stream	UE streams a video
web traffic	UE constantly accesses a random website
random ping	UE constantly pings at different rate and count

Table 3: Different profiles of generated traffic.

Anomaly	Target	Effect	Duration	Frequency
PDCP worker thread contention	CU	Delays in processing user packets, leading to packet drops	between 0.8 to 6 seconds	Every 10 seconds
Radio interference	DU air interface	SNR reduction and packet retransmissions	between 1 to 2 seconds	Every 10 seconds
FH network contention	FH and DU	SNR reduction and packet retransmissions	between 0.3 to 0.6 seconds	Every 15 seconds
MAC scheduler thread contention	DU	Missed scheduling decisions and packet drops	between 0.8 to 1.2 seconds	Every 10 seconds
Mixed anomalies	All	Combination of effects	Same as in single anomaly cases	Same as in single anomaly cases

Table 4: Description of considered anomalies. The end result of all anomalies is UE throughput degradation.

This combination makes the aggregate signal quality related KPIs of the dataset dynamic. In addition, due to random traffic patterns that we introduce, all the other KPIs are also dynamic. Note that we ensured that the data of our normal cases do not contain any anomalous samples, through manual checks.

5.3 Representative Anomalies

During the operation of our testbed for over a year, we often experienced performance degradation, which was related to several anomalies occurring across the stack, with the most common linked to thread and network contention, as well as radio interference. Motivated by this real-world experience, we initially focused our evaluation of SpotLight on generating (synthetic) anomalies linked to the above three issues, which we use for our evaluation in §6. The choice of the exact points in which to introduce anomalies (e.g., PDCP and MAC layer) was made with the goal of covering the whole RAN stack. The generated anomalies allowed us to create a test dataset containing ~33 million datapoints. Once we tuned our method with the synthetic anomalies and traffic, we also deployed it in the real system, which allowed us to detect several further real anomalies, as discussed in Section 7.1.

Below, we provide more details about the synthetic anomalies (also summarized in Table 4):

CPU contention – We emulate a class of anomalies in which thread contentions or misconfigurations throttle threads that are responsible for RAN operations (see Section 7.1 for a real-world example). We introduced CPU contention using stress-ng [32] to a worker thread responsible for relevant processing (e.g. a PDCP worker thread for CU).

Radio interference – This scenario reflects radio interference related issues that could be a result of inter-cell or external (jamming) interference. Here we focus on the latter and realize it by configuring a USRP software-defined radio to transmit an intermittent traffic over 40MHz of spectrum overlapping with our vRAN allocated spectrum and use 70-75 dB gain for the interference signal. We also inspected to ensure such interference did not exist in our training set. Note that for this anomaly we examined different interference patterns (periodic, aperiodic, etc.) but only included results for the pattern described in Table 4, due to lack of space.

Network contention – This scenario is meant to represent anomalies stemming from the sharing of network links between RAN and other functions, without proper isolation and QoS guarantees. For this anomaly, we introduced intermittent traffic on the same link that carries the FH traffic (I/Q samples) between the DU and the RU.

Mixture of anomalies – In this scenario, all six possible combinations of the four anomalies occur together in pairs.

5.4 Baselines

5.4.1 Accuracy baselines. As discussed in §2.3, most state-of-the-art time series anomaly detection methods in both RAN and ML domains are prediction based, reconstruction based or combination of both. To assess accuracy benefit with SpotLight's anomaly detection approach, we pick representative baselines from each of these categories, as outlined in Table 5. Like SpotLight, all these methods are trained on normal data.

Methods based on time series prediction (GDN [24] and LSTM-PRED [34, 56]) rely on the prediction error (i.e., difference between predicted and actual KPIs at each time step) for anomaly inference. On other hand, reconstruction based methods (TranAD [57], MADGAN [39] and LSTM-AE [34, 56]) encode and decode the test time series input using trained models and infer anomalies based on the reconstruction error (i.e., discrepancy between decoded and actual test input). For LSTM-PRED and LSTM-AE methods, we use the standard deviation of prediction/reconstruction errors during training as the threshold for anomaly detection during inference.

As a simple-minded statistical baseline method, we also use a Z-Score based time series anomaly detection [6] in which z-score is continually computed over a sliding window and classify a new point in the time series as an anomaly if its z-score is above a threshold (one standard deviation).

Note that among these baselines, LSTM-PRED, LSTM-AE and Z-Score perform 'univariate' time series anomaly detection separately for each KPI, whereas SpotLight and rest of the baselines are multivariate across all KPIs.

5.4.2 Explanation baselines.

SHAP with TranAD: SHAP [42] is a commonly used model

Category	Baseline Method
Statistical	Z-Score [6]
Prediction based	GDN [24] LSTM-PRED [34, 56]
Reconstruction based	TranAD [57], MADGAN [39] LSTM-AE [34, 56]
Prediction & Reconstruction	VAE-LSTM [41]

Table 5: Baseline anomaly detection methods.

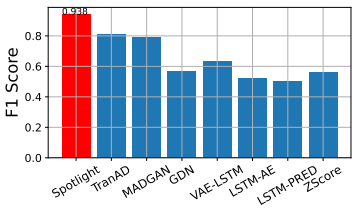


Figure 6: Average F1 score across all scenarios.

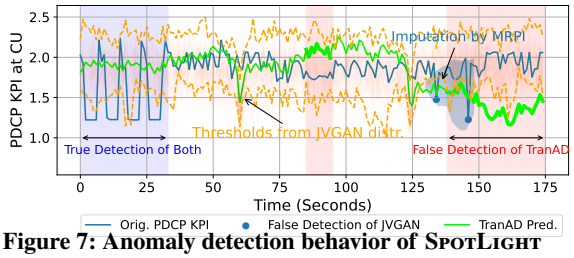


Figure 7: Anomaly detection behavior of SpotLight vs TranAD with PDCP worker thread contention.

agnostic method for explainability and provides a unified measure of feature importance – higher SHAP score for a KPI reflects its higher importance. We augment TranAD, the best performing anomaly detection method among our baselines, with SHAP (using its Omnixai [58] implementation).

GDN [24] has inherent explanation capability by modeling the set of KPIs (variables) as graph nodes and learning the edge weights between them through its attention mechanism. KPI(s) inferred to be anomalous and their highest weight neighbors forms the explanation output from GDN.

Besides the above baselines, all univariate baseline methods we consider – LSTM-PRED, LSTM-AE and Z-Score – provide explainability by default, as they treat KPIs independently and each KPI corresponds to a location in the RAN.

6 EVALUATION RESULTS

6.1 Accuracy

We begin our evaluation by comparing the overall accuracy of SpotLight to the baselines from Section 5.4 for all the configurations identified in Section 5. As an overall accuracy measure, we use F1 score – harmonic mean of precision and recall. As it can be seen from Fig. 6, the average F1 score of SpotLight for all the considered anomalous cases is 0.94, ~13% higher than the second best method (TranAD), demonstrating its high accuracy across all scenarios under study.

To understand the nature of SpotLight's accuracy, we present a detailed view of the results in Table 6, for all single anomalies in the case of 5 UEs with mixed traffic. We obtain similar results for all other scenarios, but we omit them due to lack of space. As we can observe, SpotLight has similar recall to the baselines, but fares significantly better in terms of precision. This can be observed in the example of Fig. 7, in which we show the anomalies detected by SpotLight and the second best method, TranAD, for the PDCP worker thread contention anomaly. In contrast to SpotLight, TranAD identified false anomalies during 85-95s and 130-175s. Instead, the JVGAN component of SpotLight correctly identified the samples during 85-95s as non anomalous, as they fell within the distribution it learned. On the other hand, JVGAN falsely identified the samples between 135s and 147s as anomalous. However, the imputation of MRPI corrected the false detection, maintaining SpotLight's precision at a high level.

The benefits of MRPI on improving the precision of SpotLight can also be seen in the ablation test of Table 7 for one of the 5 UEs scenarios. JVGAN captures most of the true anomalies, leading to a very high recall score for all anomalies under study. However, the precision of the model is fairly low, but is significantly enhanced by the introduction of MRPI at the expense of a marginally negative effect on recall. Finally, it should be noted that if we replace the JointVAE structure with a conventional VAE, then the precision becomes much lower, because the performance on categorical and binary variable is much worse. Introducing model uncertainty based tolerance contributes to the overall precision by reducing false detections. Without adversarial training and MC dropout, the JointVAE does not perform better than other VAEs (*e.g.* VAE-LSTM), and therefore we can see that the adversarial training and MC dropout play critical roles in enhancing the precision.

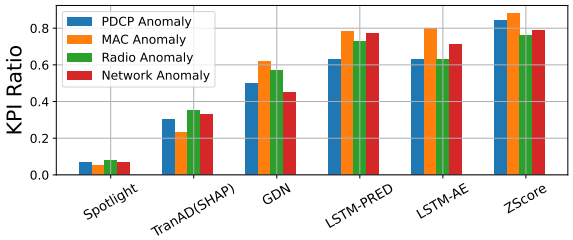


Figure 8: Ratio of KPIs flagged as potential causes for anomaly among all considered KPIs.

6.2 Explainability

Here, we focus on evaluating SpotLight in localizing and explaining anomalies. We begin by comparing the explainability power of SpotLight compared to the baseline methods of Section 5.4. We first consider the ratio of anomalous KPIs that are flagged as potential culprits among all the considered KPIs for each anomaly by both SpotLight and the baselines. A low ratio indicates that a model is more focused and does well in filtering out irrelevant KPIs from the explainability step, effectively simplifying the root cause analysis process. Fig. 8 illustrates the ratios for all single anomalies that we considered. We observe that in all cases, SpotLight has a significantly lower ratio compared to the other methods.

Next, we zoom in the causal analysis results to explain the benefits of SpotLight for explainability over the baselines. We use Anomaly Detection Ratio (**AD Ratio**) as a score for each

Method	PDCP			Radio			MAC			Network			Overall		
	F1↑	Precision↑	Recall↑	F1↑	Precision↑	Recall↑	F1↑	Precision↑	Recall↑	F1↑	Precision↑	Recall↑	F1↑	Precision↑	Recall↑
SpotLight	0.96	1	0.93	0.94	0.95	0.93	0.96	0.93	1	0.93	0.94	0.92	0.95	0.96	0.95
TranAD	0.87	0.79	0.97	0.80	0.71	0.91	0.86	0.76	1	0.68	0.59	0.80	0.80	0.71	0.92
MADGAN	0.78	0.68	0.92	0.76	0.69	0.85	0.85	0.74	1	0.67	0.61	0.75	0.77	0.68	0.88
GDN	0.66	0.54	0.85	0.41	0.27	0.82	0.49	0.35	0.84	0.08	0.06	0.15	0.41	0.30	0.66
VAE-LSTM	0.66	0.52	0.93	0.77	0.63	1	0.74	0.58	1	0.40	0.26	0.81	0.63	0.48	0.93
LSTM-AE	0.36	0.22	0.98	0.34	0.21	0.89	0.10	0.05	0.99	0.05	0.02	0.98	0.21	0.12	0.96
LSTM-PRED	0.37	0.23	0.98	0.34	0.22	0.83	0.10	0.05	1	0.05	0.02	0.98	0.21	0.13	0.94
ZScore	0.33	0.21	0.72	0.35	0.23	0.73	0.13	0.07	0.91	0.06	0.03	0.91	0.21	0.13	0.81

Table 6: F1 score, precision and recall for the 5 UE scenario and for all types of anomalies.

Method	PDCP			Radio			MAC			Network			Overall		
	F1↑	Precision↑	Recall↑	F1↑	Precision↑	Recall↑	F1↑	Precision↑	Recall↑	F1↑	Precision↑	Recall↑	F1↑	Precision↑	Recall↑
SpotLight	0.963	1	0.93	0.939	0.95	0.93	0.963	0.93	1	0.929	0.94	0.92	0.949	0.955	0.945
JVGAN (MC Dropout)	0.924	0.90	0.93	0.89	0.85	0.93	0.90	0.82	1	0.91	0.90	0.93	0.906	0.87	0.95
JVGAN (w/o MC Dropout)	0.73	0.60	0.92	0.72	0.59	0.92	0.73	0.57	1	0.758	0.65	0.91	0.735	0.60	0.938
JVAE	0.70	0.54	1	0.72	0.56	1	0.73	0.57	1	0.65	0.50	0.95	0.70	0.54	0.99
VAE	0.66	0.50	1	0.66	0.50	1	0.66	0.49	1	0.31	0.19	0.90	0.57	0.42	0.97

Table 7: Ablation Test considering the 5 UE scenario.

anomalous KPI, reflecting their relative significance, after the causal discovery step. Fig. 10 illustrates an aggregation of the KPIs that were flagged up by SpotLight and TranAD+SHAP for each anomaly, grouped by the category they belong to. The height of each bar (y-axis) shows the anomaly detection score (or SHAP score for TranAD+SHAP) of the most influential KPI of each category. The number of flagged KPIs of each category are shown at the top of each bar.

	F1↑	Precision↑	Recall↑
Overall	0.97	0.95	1
PDCP + Radio	0.95	0.91	1
PDCP + MAC	0.97	0.95	1
PDCP + Network	0.98	0.96	1
Radio + MAC	0.98	0.97	1
Radio + Network	1	1	1
MAC + Network	0.96	0.92	1

Table 8: Accuracy of SpotLight with pairs of anomalies across all scenarios.

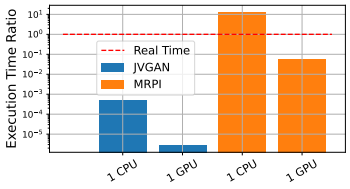


Figure 9: Ratio of processing time vs data collection time.

Taking as a concrete example the anomaly of PDCP contention in Fig. 10(a), only 18 KPIs (out of more than 600) were flagged as potential culprits by SpotLight. Eight were localized to the platform layer and were related to the PDCP threads (the correct cause) and the rest to the radio KPIs of the RLC layer. In contrast, in the case of TranAD+SHAP, 41 KPIs were flagged as significant. In addition, the flagged KPIs belong to six different categories, meaning that a domain expert would have a much harder job in identifying the actual root cause, as they would have to consider a much bigger and less focused set of KPIs. Similar observations can be made for the remaining anomalous scenarios (Fig. 10(c) - 10(d)).

6.3 Results with Multiple Anomalies

As we can observe from Table 8, SpotLight can generalize well to complex anomalous scenarios, achieving similar accuracy to the single anomaly cases, in terms of both precision and recall. It is also successful in localizing combinations of anomalies, as illustrated in Fig. 11, for the case of combined PDCP/network contention and of MAC contention and radio interference. Similar observations about the causal detection capability of SpotLight can be drawn for all the remaining combinations of anomalies (omitted due to lack of space).

6.4 Efficiency

We next show that our algorithm fits the architecture of Fig. 3 in terms of CPU and bandwidth requirements. We measure the time it takes to process metrics of a 5s window on different architectures and we plot the ratio of the processing time over the measurement time (5s). If the ratio is 1 or below, the processing can be done in real-time. We perform measurements on an AMD EPYC 7453 28-Core CPU and NVIDIA RTX A5000 GPU. We show the results in Fig. 9. We see that JVGAN is well suited for running at the edge as it takes less than 0.1% of a CPU core to process the data at line rate. In contrast, MRPI requires almost 10 CPU cores or less than 6% of a GPU to run at line rate. That amount of compute power is not available at the edge, but can be easily accommodated in the cloud. Moreover, sharing 1 GPU (approx $2,000) to manage 17 far-edge servers is a reasonable management overhead.

We next quantify the network overhead. Considering all the KPIs required by SpotLight and their frequency of collection (§4), the upstream bandwidth required from each far-edge site to ship all the KPI data streams to the cloud is nearly 2 Mbps. This would aggregate to 100 Gbps for a large telco network with 50,000 base stations, reflecting very high ingestion costs. Recall that the first JVGAN stage in our SpotLight pipeline running at each far-edge site detects likely normal cases and filters them out from further inspection in the cloud (§3.2). Therefore, the amount of data that ends up being shipped to the cloud depends on i) the rate of anomalies and ii) the detection accuracy of JVGAN. For the test data used in our evaluations (§5.3, Table 4), we find that the above data filtering characteristic of JVGAN reduces upstream bandwidth used for KPI data streams by 4 – 7×.In practice, we expect the bandwidth savings due to JVGAN to be higher, as occurence of anomalies is rarer than our deliberately challenging test scenarios. Finally, JVGAN, MRPI, and KFilter, also make the explanation more efficient by reducing the number of KPIs to process; only 3% ∼ 5% KPIs are used for the explanation step.

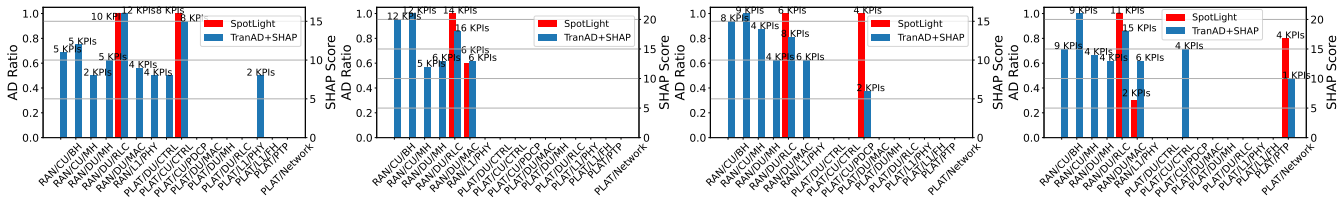


Figure 10.(a) PDCP contention Figure 10.(b) Radio interference Figure 10.(c) MAC contention Figure 10.(d) Network contention

Figure 10: KPIs flagged by SpotLight and TranAD+SHAP grouped by category. AD Ratio: score for each anomalous KPI.

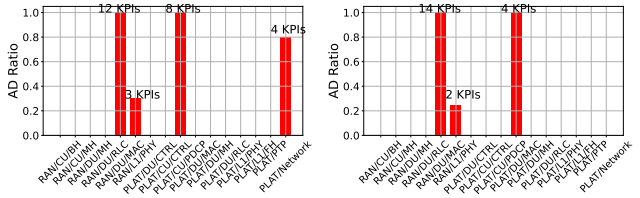


Figure 11.(a) PDCP and network contention Figure 11.(b) MAC contention and radio interference

Figure 11: KPIs flagged by SpotLight for mixture of anomalies.

SpotLight only takes around 30 seconds to run CausalNex for a ten minutes measurement.

7 REAL-WORLD EVALUATION

7.1 Case Studies

In addition to the artificial anomalies that we introduced in Section 5.3, we used SpotLight in our real deployment to evaluate its detection and localization capabilities with real, previously unknown anomalies. By collecting the exact same KPIs that we used for training, SpotLight allowed us to detect and troubleshoot three real anomalies in our network:

NIC driver misbehavior – This issue was linked to a misbehavior of the Intel ICE driver for the E810 NICs. Specifically, the driver version used in our deployment ignored the hints of interrupt thread affinities and placed interrupt threads to isolated CPU cores dedicated to the vRAN. This resulted in CPU contention with vDU worker threads, which degraded the performance of the vRAN. SpotLight successfully identified the thread contentions, which led us to investigate which other processes were running on the same cores, revealing the misbehaving threads. The issue was resolved, after upgrading the ICE driver to a version that introduced relevant patches [45].

Inter-cell interference – Our 5G testbed supports several cells on each floor of the building. Given the limited spectrum covered by our license, the cells are provisioned with a frequency reuse factor of 1. While we have optimized the placement of the radios, ensuring interference-free communication is not always possible. Inter-cell interference manifests when a UE is positioned in cell overlapping areas and is sending or receiving traffic. SpotLight was able to automatically identify inter-cell interference anomalies and to attribute the anomalies to the relevant radio KPIs (e.g., L1 BLER).

PTP synchronization loss – This issue was related to a misconfiguration, where an NTP daemon (systemd-timesyncd) was running alongside the PTP daemon (ptp4l) that was synchronizing our vDU server with the grandmaster clock. Every time that the NTP daemon would run, the server would get de-synchronized from the PTP grandmaster, resulting in UE detachments and/or traffic drops. SpotLight successfully pinpointed the PTP timestamp KPIs as the culprit.

vDU RLC queue overflow bug – This issue was related to a bug of our vDU stack, that caused the RAN to drop all packets arriving at the RLC layer for all attached UEs, if saturated with traffic for a long time. SpotLight detected succesfully the anomaly and reported that the anomalous KPI was related to the RLC queues. After reporting this issue to our vRAN vendor, we were able to identify and rectify the bug.

7.2 Operational model

Here, we demonstrate how SpotLight works in an operational environment. In such environment, it is important to have a very low fraction of false positives to reduce the unnecessary workload of operational teams. Our observation window is 6.4s (Section 3.1), which means that we have about 10 observations per minute. Note that most of the RAN anomalies we had observed are not transient and last for many minutes, so, in order to reduce the false positives, we aggregate the observations over one minute, and we report an anomaly only if we have 10 positive reports out of 10 windows. This means that the precision over 1h interval is $2 \cdot 10^{-4}$ and the recall is 0.91 – 0.98. We verify these results by running the system for 1h without an anomaly, and we note that we do not observe a single false positive during that time.

8 CONCLUSIONS

In this work, we presented SpotLight, a tailored system architecture with a deep generative modeling based method for Open RAN anomaly detection and localization, that is distributed between the edge and the cloud. SpotLight manages to provide highly accurate and explainable anomaly detection results, while remaining computationally efficient. To train and evaluate SpotLight, we developed a large scale and realistic data collection process spanning both the RAN and the platform layer on an enterprise-scale 5G Open RAN deployment.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments and suggestions that greatly improved this paper. This work was supported by a project funded by the UK Department for Science, Innovation and Technology (DSIT).

REFERENCES

[1] NTIA 5G Challenge. https://www.challenge.gov/?challenge=5g-challenge-2022.
[2] O-RAN Alliance. https://www.o-ran.org/.
[3] Overcoming Challenges of Multi-Vendor Open RAN. https://images.samsung.com/is/content/samsung/p5/global/business/networks/insights/white-paper/mvoran-challenges/Samsung-MVoRAN-Challenges-Whitepaper.pdf.
[4] SONIC Labs. https://www.digicatapult.org.uk/expertise/programmes/programme/sonic/.
[5] SpotLight Dataset Schema. https://github.com/netsys-edinburgh/SpotLight/blob/main/dataset-schema.pdf.
[6] How to build robust anomaly detectors with machine learning. https://www.ericsson.com/en/blog/2020/4/anomaly-detection-with-machine-learning, Apr 2020.
[7] Open RAN: Big Opportunity, Big Challenges. https://www.forbes.com/sites/forbestechcouncil/2021/12/02/open-ran-big-opportunity-big-challenges/?sh=6a162f0b63d5, Dec 2021.
[8] Open RAN: ready for prime time? https://www.analysysmason.com/contentassets/bf99e24c249e444fb5c03fdfd125aaa8/analysys_mason_open_ran_reality_apr2021_rdns0_rma18.pdf, Apr 2021.
[9] Overcome the Key Challenges of Open RAN Roll-out. https://www.fiercewireless.com/sponsored/overcome-key-challenges-open-ran-roll-out, Apr 2021.
[10] Recognizing the Challenges of Making Open RAN Work. https://www.spirent.com/blogs/recognizing-the-challenges-of-making-open-ran-work, Jun 2022.
[11] Open RAN: Key Challenges and Opportunities for Network Operators. https://ts2.space/en/open-ran-key-challenges-and-opportunities-for-network-operators/, Jul 2023.
[12] 3GPP. Key Performance Indicators (KPI) for Evolved Universal Terrestrial Radio Access Network (E-UTRAN): Release 17. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2026, 2022.
[13] Harshit Agarwal and Samuel Gaver. Latencies, schedulers, interrupts oh my! the epic story of a linux kernel upgrade. https://www.nutanix.dev/2021/12/09/latencies-schedulers-interrupts-oh-my-the-epic-story-of-a-linux-kernel-upgrade/, 2021.
[14] Faraz Ahmed, Jeffrey Erman, Zihui Ge, Alex X. Liu, Jia Wang, and He Yan. Detecting and Localizing End-to-End Performance Degradation for Cellular Data Services Based on TCP Loss Ratio and Round Trip Time. *IEEE/ACM Transactions on Networking*, 25(6):3709–3722, 2017.
[15] Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *arXiv preprint arXiv:2208.09399*, 2022.
[16] Wilfrid Azariah, Fransiscus Asisi Bimo, Chih-Wei Lin, Ray-Guang Cheng, Rittwik Jana, and Navid Nikaein. A Survey on Open Radio Access Networks: Challenges, Research Directions, and Open Source Approaches. *arXiv preprint arXiv:2208.09125*, 2022.
[17] Paramvir Bahl, Matthew Balkwill, Xenofon Foukas, Anuj Kalia, Daehyeok Kim, Manikanta Kotaru, Zhihua Lai, Sanjeev Mehrotra, Bozidar Radunovic, Stefan Saroiu, et al. Accelerating Open RAN Research Through an Enterprise-scale 5G Testbed. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–3, 2023.
[18] Paul Beaumont, Ben Horsburgh, Philip Pilgerstorfer, Angel Droth, Richard Oentaryo, Steven Ler, Hiep Nguyen, Gabriel Azevedo Ferreira, Zain Patel, and Wesley Leong. CausalNex. https://github.com/quantumblacklabs/causalnex, October 2021.
[19] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3):1–33, 2021.
[20] Bouziane Brik, Hatim Chergui, Lanfranco Zanzi, Francesco Devoti, Adlen Ksentini, Muhammad Shuaib Siddiqui, Xavier Costa-Pérez, and Christos Verikoukis. A Survey on Explainable AI for 6G O-RAN: Architecture, Use Cases, Challenges and Research Directions. *arXiv preprint arXiv:2307.00319*, 2023.
[21] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), Jul 2009.
[22] Ashima Chawla, Paul Jacob, Saman Feghhi, Devashish Rughwani, Sven van der Meer, and Sheila Fallon. Interpretable unsupervised anomaly detection for RAN cell trace analysis. In *2020 16th International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2020.
[23] Zahra Zamanzadeh Darban, Geoffrey I Webb, Shirui Pan, Charu C Aggarwal, and Mahsa Salehi. Deep learning for time series anomaly detection: A survey. *arXiv preprint arXiv:2211.05244*, Dec 2022.
[24] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4027–4035, 2021.
[25] Emilien Dupont. Learning disentangled joint continuous and discrete representations. *Advances in neural information processing systems*, 31, 2018.
[26] eBPF. Dynamically program the kernel for efficient networking, observability, tracing, and security. https://ebpf.io/.
[27] Xenofon Foukas and Bozidar Radunovic. Concordia: Teaching the 5G vRAN to share compute. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 580–596, 2021.
[28] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. Taking 5G RAN analytics and control to a new level. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2023.
[29] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
[30] Andres Garcia-Saavedra and Xavier Costa-Pérez. O-RAN: Disrupting the Virtualized RAN Ecosystem. *IEEE Communications Standards Magazine*, 5(4):96–103, 2021.
[31] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, oct 2020.
[32] Red Hat. Stress testing real-time systems with stress-ng. https://wiki.ubuntu.com/Kernel/Reference/stress-ng.
[33] Anand Padmanabha Iyer, Li Erran Li, and Ion Stoica. Automating diagnosis of cellular radio access network problems. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 79–87, 2017.
[34] Caner Kilinc et al. JADE: Data-Driven Automated Jammer Detection Framework for Operational Mobile Networks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1139–1148. IEEE, 2022.
[35] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
[36] Paulo Valente Klaine, Muhammad Ali Imran, Oluwakayode Onireti, and Richard Demo Souza. A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks. *IEEE Communications*

Surveys Tutorials, 19(4):2392–2431, 2017.

[37] Kubernetes. CPU Throttling on Linux kernel 5.4.0-1029-aws. https://github.com/kubernetes/kubernetes/issues/97445, 2020.

[38] Michael Larabel. The latest on the linux 5.9 kernel regression stemming from page lock fairness. https://www.phoronix.com/review/linux-59-fairness, 2020.

[39] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International conference on artificial neural networks*, pages 703–716. Springer, 2019.

[40] lightreading. T-Mobile's network chief pours cool, but not cold, water on O-RAN. https://www.lightreading.com/open-ran/t-mobiles-network-chief-pours-cool-but-not-cold-water-on-o-ran/d/d-id/765518, 2020.

[41] Shuyu Lin, Ronald Clark, Robert Birke, Sandro Schönborn, Niki Trigoni, and Stephen Roberts. Anomaly detection for time series using vae-lstm hybrid model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4322–4326. Ieee, 2020.

[42] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[43] Rashid Mijumbi, Abhaya Asthana, Markku Koivunen, Fu Haiyong, and Qinjun Zhu. Design, implementation, and evaluation of learning algorithms for dynamic real-time network monitoring. *International Journal of Network Management*, 31(4):e2108, 2021.

[44] O-RAN. Near-real-time RAN intelligent controller E2 service model (E2SM) KPM. https://orandownloadsweb.azurewebsites.net/download?id=393, 2022.

[45] opendev. Kernel-modules: IRQ affinity hint fix-ups. https://opendev.org/starlingx/kernel/commit/7ded00431675bbab05fe254b90efd08eb335f101?style=unified&whitespace=ignore-all.

[46] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Paul Beaumont, Konstantinos Georgatzis, and Bryon Aragam. Dynotears: Structure learning from time-series data. *arXiv preprint arXiv:2002.00498*, 2020.

[47] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. DYNOTEARS: structure learning from time-series data. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 1595–1605. PMLR, 2020.

[48] Georgios Patounas, Xenofon Foukas, Ahmed Elmokashfi, and Mahesh K Marina. Characterization and Identification of Cloudified Mobile Network Performance Bottlenecks. *IEEE Transactions on Network and Service Management*, 17(4):2567–2583, 2020.

[49] 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.

[50] Juan M Ramírez, Fernando Díez, Pablo Rojo, Vincenzo Mancuso, and Antonio Fernández-Anta. Explainable machine learning for performance anomaly detection and classification in mobile networks. *Computer Communications*, 200:113–131, 2023.

[51] RedHat. RHEL8: Latency issue on Kubernetes / k8s / OpenShift. https://bugzilla.redhat.com/show_bug.cgi?id=1795049, 2020.

[52] Lars Ruthotto and Eldad Haber. An introduction to deep generative modeling. *arXiv preprint arXiv:2103.05180*, 2021.

[53] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: a comprehensive evaluation. *Proceedings of the VLDB Endowment*, 15(9):1779–1797, 2022.

[54] Andy Sutton. Open Radio Access Network. *ITP Journal*, 16(2):32–37, 2022.

[55] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

[56] Hoang Duy Trinh, Engin Zeydan, Lorenza Giupponi, and Paolo Dini. Detecting mobile traffic anomalies through physical control channel fingerprinting: A deep semi-supervised approach. *IEEE Access*, 7:152187–152201, 2019.

[57] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. TranAD: deep transformer networks for anomaly detection in multivariate time series data. *Proceedings of the VLDB Endowment*, 15(6):1201–1214, 2022.

[58] Wenzhuo Yang, Hung Le, Silvio Savarese, and Steven Hoi. Omnixai: A library for explainable ai.(2022), 2022.

[59] Yannan Yuan, Jiaolong Yang, Ran Duan, I Chih-Lin, and Jinri Huang. Anomaly detection and root cause analysis enabled by artificial intelligence. In *2020 IEEE Globecom Workshops (GC Wkshps*, pages 1–6. IEEE, 2020.