# Fault And Performance Management In Multi-Cloud Virtual Network Services Using AI: A Tutorial And A Case Study

Lav Gupta[a*], Tara Salman[a], Maede Zolanvari[a], Aiman Erbad[b], Raj Jain[a]

[a]Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, USA

[b]Department of Computer Science and Engineering, Qatar University, Doha, Qatar

***Abstract*— Carriers find Network Function Virtualization (NFV) and multi-cloud computing a potent combination for deploying their network services. The resulting virtual network services (VNS) offer great flexibility and cost advantages to them. However, vesting such services with a level of performance and availability akin to traditional networks has proved to be a difficult problem for academics and practitioners alike. There are a number of reasons for this complexity. The challenging nature of management of fault and performance issues of NFV and multi-cloud based VNSs is an important reason. Rule-based techniques that are used in the traditional physical networks do not work well in the virtual environments. Fortunately, machine and deep learning techniques of Artificial Intelligence (AI) are proving to be effective in this scenario. The main objective of this tutorial is to understand how AI-based techniques can help in fault detection and localization to take such services closer to the performance and availability of the traditional networks. A case study, based on our work in this area, has been included for a better understanding of the concepts.**

***Key Words*— Network Function Virtualization, Virtual Network Services, Service Function Chains, Virtual Network Functions, multi-cloud, fault management, performance management, machine learning, deep learning**

## 1. Introduction

Network Function Virtualization (NFV) is being regarded as one of the most important developments of the last decade for communication networks. The Gartner Hype Cycle 2018 describes NFV and network performance as the key technologies, alongside the Internet of Things (IoT) and 5G [1]. NFV allows telecommunications carriers[1] to instantiate software-based network functions on commercial, off-the-shelf hardware. Using these virtual network functions (VNFs) as the building blocks for creating Virtual Network Services (VNSs), carriers can change the way the network services are provided. They are prepared to bear the pains of making this major change in order to reap the benefits of the reduced cost of deployment, agility in introducing new services, ease of scaling, independence from proprietary equipment and vendor lock-in [2]. The virtual resources (e.g., virtual machines and virtual networking) for building these services can be obtained from the in-house datacenter, carrier-cloud owned by carriers themselves or public clouds owned by Cloud Service Providers (CSPs). Use of multiple clouds gives additional advantages like more competitive prices, larger resource pool, better points of presence and avoidance of single point of failure because of a cloud blackout [3] [4].

Corresponding Author Address: Dept. of CSE, Washington University in St Louis, St Louis, MO 63130, USA, Tel: +1 314-825-0063
E-mails: lavgupta@wustl.edu (Lav Gupta), tara.salman@wustl.edu (Tara Salman), maede.zolanvari@wustl.edu (Maede Zolanvari), jain@wustl.edu (Raj Jain), aerbad@qu.edu.qa (Airman Erbad)

[1] The term carrier refers to all communications service providers including Internet Service Providers (ISPs)

### 1.1 Challenges of VNS Deployments

Despite many advantages, there are several challenges in providing large-scale deployments of NFV-based VNSs. It is important to identify these challenges so that they can be met and this promising technology does not disappear into oblivion. Some of the main challenges are listed here:

a) Performance and availability of VNSs are nowhere close to the traditional networks. The traditional networks have five nines availability (99.999%), which translates to 5.25 minutes of downtime in a year. Cloud information technology applications have been working more on three nines (99.9%) availability, which go up to 8.76 hours of downtime in a year.

b) Traditional networks are built to the stringent quality of service (QoS) norms defined by Fault, Configuration, Accounting, Performance and Security (FCAPS) standards like ISO Common Management Information Protocol (CMIP) and ITU Telecommunications Management Network (TMN) M.3010 and M.3400 recommendations [5] [6] [7] [8]. Such norms are still to be fully defined and met for the VNS deployments.

c) In NFV-based VNSs, faults may occur for many more reasons compared to traditional physical networks. The cloud infrastructure consists of virtual resources such as virtual machines, virtual storage, and virtual network links. These virtual resources are created on shared physical resources like server hardware, system software or network links, using virtualization software (e.g., Hypervisors). One reason why virtual resources may fail is because of the failure of physical resources. Even if the physical resources are operational, the virtual resources may themselves fail [9]. Taking this argument a little further, even if both physical and virtual resources are healthy, the VNFs, like routers, instantiated on these virtual resources can develop algorithmic problems causing VNSs to malfunction or totally break down. The myriad levels of dysfunctions make handling of fault and performance (FP) issues in NFV over clouds more abstract and complex.

d) Internet Engineering Task Force (IETF) has recently identified ensuring performance and guaranteeing the QoS as open research areas and technology gaps in NFV [10]. Without a robust mechanism for handling these issues, carriers would find it very difficult to meet the quality, reliability and availability norms. This calls for vigorous research efforts so that NFV deployments acquire carrier-grade performance and availability [10] [11] [12]. ITU has included the fault management of the cloud-based NFV in their standardization agenda [13]. The National Science Foundation (NSF) is supporting research work in this important area, which can potentially change the way the telecommunication services are delivered [14].

### 1.2 Objective and Goals of this Tutorial and the Related Case Study

The primary objective of this tutorial is to take a deep and incisive look at the complexities of detection and localization of fault and performance issues in an NFV multi-cloud environment and to examine how machine and deep learning techniques can help to tackle them. We divide this objective into the following goals:

a) Discuss the architecture, creation and management of VNSs with a real-life example.

b) Elucidate clearly the fault and performance (FP) management problem and its complexities.

c) Explain why the traditional methods do not perform well in the cloud-based NFV environments and how AI techniques like machine learning and deep learning can help.

d) Describe the AI based FP management framework that we have evolved.

e) Give these discussions a more concrete and practical footing, with a case study that describes in detail the use of a hybrid shallow and a deep learning model to detect and localize some important aspects of fault and performance issues.

We believe that this tutorial would provide background and motivation for other researchers to contribute to this important area.

### 1.3 Structure of the Tutorial

The remaining paper has been organized as follows: Section 2 provides the background information about VNSs and their management. In Section 3, the fault and performance management problem, its complexities, markers with details of the proposed work are presented. Section 4 discusses different methods for fault and performance management. Applicability of AI-based approach to FP management is described in Section 5. The concepts discussed so far in the tutorial have been used to evolve a framework for VNS over multi-cloud is described in detail, which is discussed in Section 6. To demonstrate that such a frameworks provides a viable solution to the complex FP problem in the cloud and NVF environment, Section 7 presents the evaluation of the framework in the form of a case study. Section 8 gives a summary and possible research directions emanating from this work. A list of acronyms used in this paper has been included in the annexure.

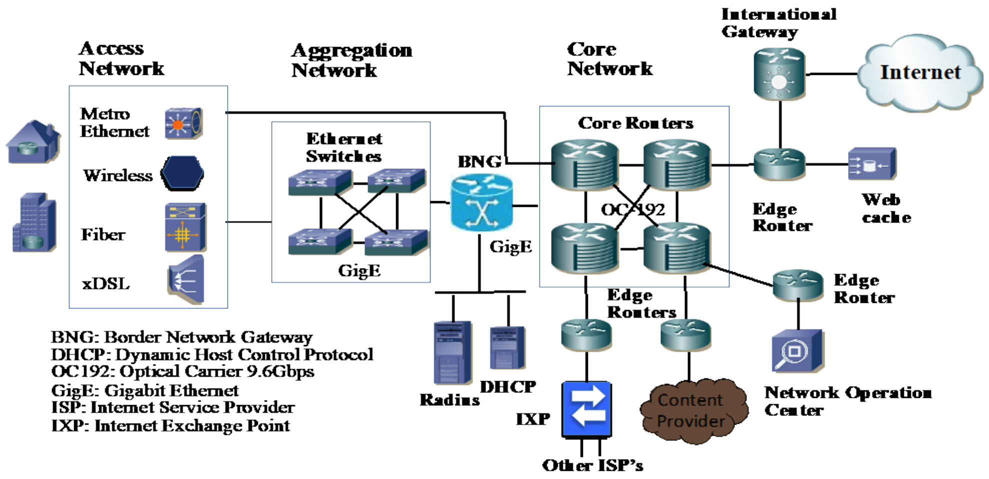


**Fig. 1.** A carrier's broadband service network

## 2. Background - VNSs and their Management

In this section, we explore the structure of a VNS, using NFV over a multi-cloud system, with the help of an example of a carrier network service. Additionally, we shall see the complexities and deficiencies in its management setup, which make a strong case for a predictive fault and performance management framework. Fig. 1 illustrates a carrier's implementation of the broadband Internet service. The *Access Network* consists of various technologies through which home and business customers access the Internet and the related services. The *Aggregation Network* collects various streams of traffic and concentrates them on higher capacity links to the core network. The Border Network Gateway (BNG) is situated at the border of the core and provides Layer-2 and Layer-3 connectivity, policy injection, QoS and accounting of user sessions and traffic flows. The *Core Network* contains core routers that transport traffic and connect to the Internet and other services like content delivery through the edge routers. The core also connects to the Internet Exchange Points (IXPs) for exchanging traffic with other local ISPs without using the expensive international bandwidth.

### 2.1 Structure and Components of VNS

The broadband services, like other carrier services, are currently provided through networks constructed from physical appliances like routers, aggregation switches and Digital Subscriber Line Access Multiplexers (DSLAMs) from various Original Equipment Manufacturers (OEMs). In discussing the virtualization of this network, we will take a top-down approach, starting from VNS and go down to the infrastructure level as shown in Fig. 2.

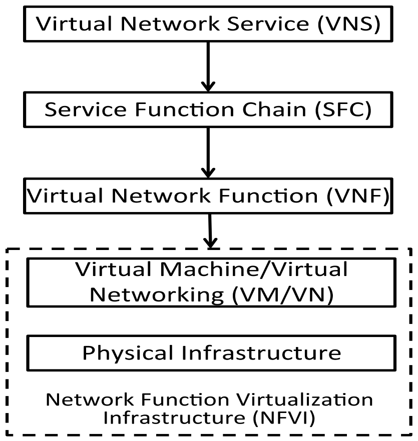

Virtual Network Service (VNS)
Service Function Chain (SFC)
Virtual Network Function (VNF)
Virtual Machine/Virtual Networking (VM/VN)
Physical Infrastructure
Network Function Virtualization Infrastructure (NFVI)

**Fig. 2.** The virtualization hierarchy

1) Virtual Network Service (VNS)

From the illustration of a complex network service in Fig. 1, we abstract a subset to represent a VNS that we can use as an example. Fig. 3 shows this VNS being composed of virtual network functions (VNFs) realized as VNF1 to VNF8. The figure also shows that the carrier has retained DSLAMs as Physical Network Function (PNF) from their legacy network, as these functions might not have reached their end-of-life. VNFs of a service may belong to different vendors, owned by different operators, managed by different platforms and even unaware of each other. In such a case, the service is a multi-domain VNS [16].

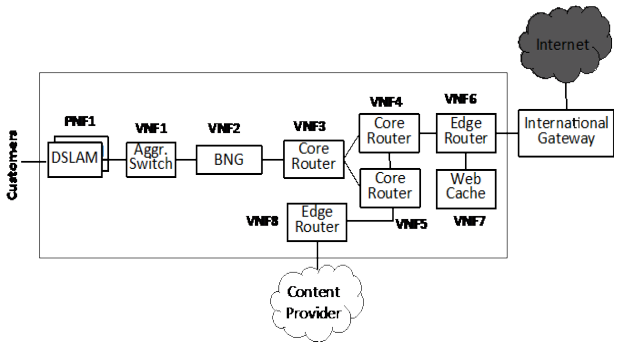


**Fig. 3.** Virtual broadband service

2) Service Function Chains (SFCs)

An SFC consists of an ordered set of interconnected VNFs (and possibly PNFs), which perform pre-programmed operations on the traffic routed through them. A carrier may obtain resources from multiple cloud service providers to avoid problems like vendor lock-in (when the carrier is forced to buy resources from a particular cloud service provider) or a service failure because of a single cloud blackout. VNFs are instantiated on these cloud resources and linked using virtual networking resources to form one or more SFCs [17] [18].

Fig. 4 shows an SFC with two paths, i.e., PNF1-VNF1-VNF2-VNF3-VNF4-VNF6 for the Internet access and PNF1-VNF1-VNF2-VNF3-VNF5-VNF8 for content services. As can be seen in this figure, there are multiple paths through meshed core routers through which traffic can be routed if the selected link fails or if there is congestion on the selected link. VNFs of an SFC are connected in the same manner as the physical

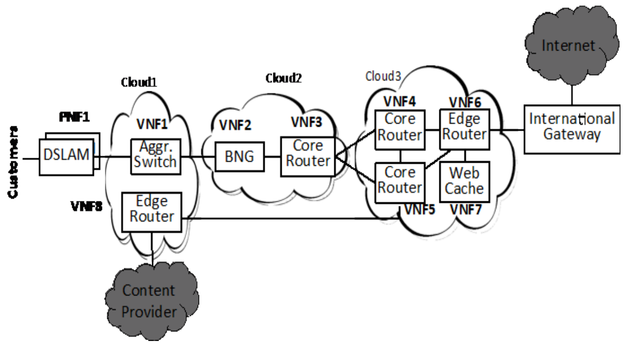


**Fig. 4.** SFC created on multiple clouds

appliances are connected in a traditional network [19]. Some VNFs are dimensioned with multiple instances to handle the volume of the expected traffic.

### 3) Virtual Network Functions (VNFs)

A VNF is the virtual counterpart of a network appliance or a middlebox implemented by running software over commercial off-the-shelf general purpose servers. The software to implement a network function may run over a general-purpose physical machine or over the virtual machine(s) created on physical machines using virtualization software. VNFs can also be instantiated on virtual resources obtained from one or more cloud service providers. Each VNF has a well-defined functional behavior and interfaces for interconnection with other VNFs or PNFs. Fig. 5 shows a VNF with its Element Management System (EMS) and interfaces to the rest of the network [19] [20].

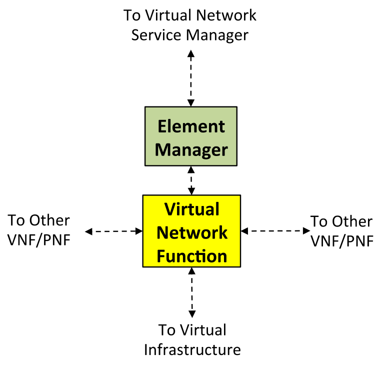


**Fig. 5.** A VNF implementation

Some examples of pre-programmed VNFs are given in Table 1.

**TABLE 1**
EXAMPLES OF COMMERCIALLY AVAILABLE VNFS

| VNF | Function | OEM |
|---|---|---|
| **ISRv** | Integrated Services Router | Cisco Systems |
| **vSphere** | Distributed Switch | VMware |
| **SRX** | Firewall | Juniper Networks |
| **440Vx** | Load Balancer | Barracuda |
| **SBC SWe** | Session Border Controller | Ribbon Communications |
| **Vyatta** | vRouter | Brocade |
| **Steelhead CX 555V** | WAN Accelerator | Riverbed Technology |
| **SSR 800** | Smart Service Router | Ericsson |
| **Liquid Core** | Mobile core virtualization | Nokia Siemens Networks |

Even though the concept of VNF is just about 7 years old, there have been some major innovations proposed. For instance, a VNF could be implemented using a set of predefined and reusable microservices. Microservices are easy to replace in case of a fault. However, management of microservices-based VNS becomes complex. Similarly, the concept of cloud-native VNFs (or CNFs) is also being currently discussed [83]. CNFs are created on clouds using containers. They are purported to be lightweight and more agile compared to the traditional VNFs. However, these new concepts are beyond the scope of this tutorial. We refer the interested readers to [21] and other references on the subject.

Protocols for routing of traffic, through an SFC, are being worked upon by the standards organizations. For example, two protocols - Segment Routing (SR) and the Network Service Header (NSH) - are under development in the IETF [84]. SR is a modified version of source routing. In SR the IPv6 header is extended to include the Segment Routing Header (SRH), which decides the path of the traffic packets. A segment is a path through a carrier network. The internals of the segment may not be exposed to the users. For example, it may be a Multi-Protocol Label Switching (MPLS) tunnel or may be a sequence of IP routers. Each segment has an ID and may contain information about the treatment of the traffic on that segment. A Software Defined Networking (SDN) controller may utilize the Path Computation Element Protocol (PCEP) to find the most appropriate segments and instruct the classifier to direct the traffic flow accordingly. NSH, on the other hand, can work with IPv4, IPv6, and Ethernet. NSH is an 8-byte header followed by a number of optional variable length context

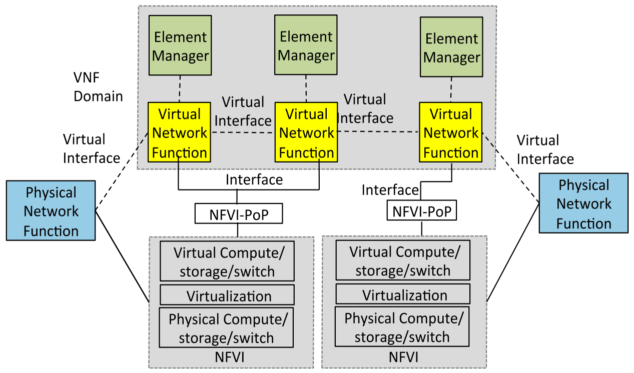


**Fig. 6.** Network function virtualization infrastructure

headers containing some metadata to be used by NSH-aware devices. Implementation of service function chains with NSH capabilities requires NSH-aware virtual switches and a central controller.

### 4) Virtual Machines (VMs) and Network Function Virtualization Infrastructure (NFVI)

Network Function Virtualization Infrastructure (NFVI) consists of all the hardware and software used to deploy

VNFs. This infrastructure-hosting site is referred to as NFVI Point of Presence (PoP). The virtual compute and storage resources, in an NFVI-PoP, are interconnected to form a network of virtual resources that can host carrier services. Fig. 6 shows an example of a system with three VNFs, their EMSs, hosted on two NFVIs of two cloud service providers. External connectivity may be possible through the designated switching and routing devices [23].

### 2.2 Management of VNSs from an FP Management Perspective

Fig. 7 shows the management set-up of a VNS. An understanding of the functions of interacting platforms would help the reader appreciate the FM management framework as described in Section 6. As can be noticed, the figure shows three different management platforms interacting to make VNSs work. NFV *Management and Orchestration* (NFV-MANO) and its subsystems use the virtual infrastructure provided by the *Multi-cloud Management and Control Platform* (MMCP) to create and manage VNFs, SFCs, and VNSs [24]. MANO has the responsibility of performance measurement, event reporting, correlation and assistance in fault management of the VNSs and their constituents. The MMCP creates virtual machines, virtual storage, and virtual networking links. It also manages the placement and migration of these virtual resources over the available clouds [20]. The *Operation Support System (OSS)* of the carrier, with its *Network Management System (NMS)*, manages the deployment and operation of the VNSs. The OSS carries out the network management by providing support for the provisioning of services, management of fault and performance and maintaining an inventory of the resources used.

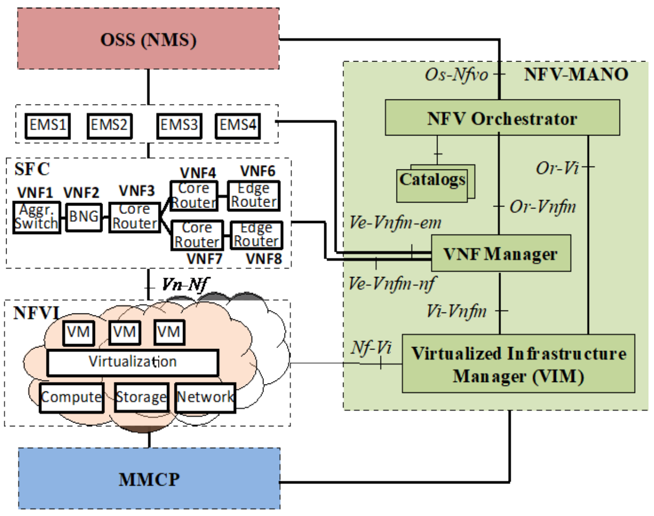


**Fig. 7.** Orchestration and management of VNS

In view of what has been said above, the fault management function becomes a shared responsibility. The relative distribution of responsibilities among various platforms and their interactions are yet to be fully defined. In order to understand the fault and performance management of VNSs, we need to discuss the sub-systems of MANO and their interactions in some more detail [22].

#### 1) Virtual Infrastructure Manager (VIM)

VIM manages all the virtual and physical resources in NFVI to enable higher layers of MANO to do their jobs of creating VNFs and SFCs. VIM manages the lifecycle of all the virtual resources in one NFVI domain (one infrastructure provider's domain) and applies security policies on them. VIM collects information about the performance events and measurements from NFVI over the Nf-Vi reference point and forwards them to NFV Orchestrator (NFVO) through its northbound reference point (Or-Vi). In the cloud environment, VIM would interact with the cloud management platform for obtaining virtual resources. In a multi-cloud or a multi-carrier service, there may be multiple VIMs managing the resources.

#### 2) Virtual Network Function Manager (VNFM)

The VNFM instantiates and configures VNFs with resources obtained through the VIM. During the lifetime of a network service, VNFM manages the complete lifecycle of the VNFs (scaling, descaling and eventually terminating when they are no longer required). It is entrusted with the important functions of FP management of VNFs. For this, VNFM interacts with the EMSs of the VNFs to obtain fault and performance markers. The EMS (not a part of the MANO) collects device statistics, logs notifications, alarms and events, and performance statistics [25]. As shown in Fig. 7, VNFM shares this information with the NFVO over the Or-Vi reference point. Since a VNS may have VNFs from multiple providers, it is important that NFVO can interact with them through the standard reference points.

#### 3) NFV Orchestrator (NFVO)

NFVO is at the heart of the MANO architecture. It carries out two of its main functions: resource orchestration and service orchestration. Using its resource orchestration function, NFVO coordinates the acquisition and release of the NFVI resources by interfacing with the VIMs. NFVO instantiates the VNF Manager, which in turn manages VNFs as explained above. Service orchestration functionality deals with onboarding new network services using the information from descriptor files within various catalogs. For fault and performance issues, NFVO has to coordinate with the carrier's OSS and multi-cloud management platform.

#### 4) Catalogs and Repositories

MANO has several catalogs and repositories containing descriptor files, which NFVO uses to carry out the orchestration functions [26]. For example, there is a catalog for service onboarding templates and another for requirements for creation and operation of the VNFs. There is an NFV repository for storing all instances of network services and yet another for the available and used NVFI resources.

5) MANO Reference Points – Interaction with Other Functional Blocks.

All exchanges among the sub-systems of MANO and between them and external entities, including those pertaining to fault and performance status, take place through the defined reference points. Table 2 contains a brief description of these reference points and what fault and performance related information they carry [27]. Interactions between MANO and MMCP have not been defined in the NFV specifications. This has to be taken into consideration in a VNS fault and performance management solution.

**TABLE 2**
FAULT AND PERFORMANCE INFORMATION OVER NFV REFERENCE POINTS

| Reference Point | Endpoints | Functions |
|---|---|---|
| **OS-Nfvo** | OSS and NFVO | 1. Carries information related to VNS requirements from OSS to NFVO<br>2. NFVO creates VNS and applies carrier policies<br>3. Carries usage, accounting, fault and performance events for all VNS, VNF and NFVI resources. |
| **Or-Vnfm** | NFVO and VNFM | 1. VNF and NFVO exchange information related to the creation and management of VNFs.<br>2. Forwards events related to VNF to the NFVO |
| **Vnfm-Vi** | VNFM and VIM | 1. Carries information about NFVI requests from VIM. |
| **Or-Vi** | NFVO and VIM | 1. Reserve NFVI resources for VNS<br>2. Coordinating scaling and release |
| **Nf-Vi** | VIM and NFVI | 1. Creating/Obtaining virtual resources for creating VNS<br>2. Failure event, measurement results, and configuration information to VIM |
| **Vn-Nf** | VNF and NFVI | 1. Physical and virtual resource information to VNFM for ensuring creation scaling and performance and portability of VNFs. |
| **Ve-Vnfm-Vnf** | VNFM and VNF | 1. Event reporting by VNF to VNFM<br>2. Communication from VNFM to VNF regarding configuration and events<br>3. VNF aliveness check |
| **Ve-Vnfm-em** | VNFM and EMS | 1. Same functions as Ve-Vnfm-Vnf for virtualization-aware EMS |

## 2.3 Comparison of Competitive Network Service Orchestration Offerings

It is evident from the discussion in Section 2.2 that the components of MANO are important parts of the FP management of VNSs. We present some of the well-known MANO platforms in this section and compare their features relating to the management of FP problems that threaten the availability and reliability of these services. The most important purpose of this comparison is to bring out the necessity for carrying out research work in the area of FP management in the NFV and multi-cloud environment. We include multi-cloud and multi-carrier domain support and interaction with the OSS, which are the important considerations for our discussions. Most MANO implementations are in initial releases and under active development. The idea, therefore, is to be representative and not comprehensive.

Table 3 gives a comparison of the Network Service Orchestration platforms. The following criteria have been used for classification [28][29]. It may be noted that a blank cell indicates that sufficient information is not available to adjudge the product on the corresponding criterion. Orchestration of end-to-end service is important from the carrier's point of view. In the absence of this feature, manual configuration and a large amount of scripting may be required to orchestrate complete services. Handling of multiple VNFM and VIM support allows management of SFCs across multiple carrier domains. Three important criteria are whether the orchestrator maintains the carrier-grade performance, whether it can coordinate with the OSS for fault and management functions and how sophisticated is the fault and performance management. We see that many of the platforms are yet to achieve the required level of sophistication of fault and performance management.

## 3. Fault and Performance (FP) Problem Description

FP issues may range from simple single point failures to complex faults with many devices involved. A fault may appear because of some hardware or algorithmic error in the system. If the error were due to a malfunction or a deviation of the system from the accepted normal performance, then a fault would result. Additionally, one faulty entity may affect other neighboring entities and faults may propagate. In such a case, the faulty or other connected devices may give out notifications. The variety of FP issues that can affect the carrier networks is large and difficult to detect, diagnose and localize [30] [31][32]. When we add to this the virtualization and the cloud computing layers, the number of ways faults can affect the virtualized network far exceeds that of their physical counterparts. In this setup, when faults traverse through the physical and virtual layers they change their presentation and produce a different set of markers in different layers, making it even more difficult to correlate an observed system disorder with the original fault [33].

Traditional failure detection mechanisms are ineffective or inapplicable in NFV environments. Traditional methods depend on probing or running tests on hardware, which are not accessible to the carriers who deploy services on virtual resources. Too much of probing or software testing may overload the VMs that have been optimized for the network function hosted on them. Attempts to apply other traditional methods, like rule-based approaches involving direct correlation of the markers with the faults, get mired in complexity and prove to be inadequate. New methods would be required to deal with faults in VMs or VNFs, which cause

**TABLE 3**
COMPARISON OF SOME COMPETITIVE NETWORK SERVICE ORCHESTRATION SOLUTIONS FROM FP PERSPECTIVE

| Platform / Criteria | ETSI | Linux Foundation | Open Networking Foundation | Gigaspace | Cisco | Netcracker (NEC) | Oracle |
|---|---|---|---|---|---|---|---|
| **NFVO solution nomenclature** | Open Source MANO (OSM) | OPEN-O/ONAP | XOS/CORD + ONOS[1] | Cloudify | Network Service Orchestrator | RT MANO Network Orchestration | Network Service Orchestration |
| **Inception date** | Launched 2016, Spearheaded by Carriers | Launched 2016 | Launched 2015 | Launched 2014 | | Launched 2015 | Launched 2015 |
| **Current Release** | Rel 6, June 2019 | Casablanca, April 2019 | Gambia 7.0, Nov 2018 | Rel 4.6, June 2018 | Resease 4.7 June 2018 | Rel 12 May 2017 | Release 7.3.5 April 2017 |
| **Whether carrier-grade** | | Planned | Field Trials | Yes | | Yes | Yes |
| **Open Solution** | Yes | Yes | Use case of open source ONOS | Yes, TOSCA based | | | Partly |
| **End-to-End service** | Planned | For defined use cases | For carrier use | Yes, may require plug-ins for underlay | Yes | Yes | Yes |
| **Fault/Performance Management Sophisticaton**[2] | Level 1 | Level 3 | Level 1 | | Requires extension with Crosswork Network Automation | Level 2 | |
| **Support for Multiple VNF /VIM** | Yes | OpenStack VIM+ generic VNFM | OpenStack VIM+ VNFM like functions | Yes | Yes | Yes | Yes |
| **Cloud platform neutral/Multiple Clouds** | Yes | Planned | Multi-access edge cloud | Yes | | | |
| **Integrates with BSS/OSS** | Yes | | Yes | Yes | OSS | | Yes[3] |

[1] ONOS is under Linux Foundation. CORD is under Open Networking Foundation.
[2] Level 1: e.g., log-based correlation; Level 2: includes a detection mechanism and root cause analysis; Level 3: predictive detection/localization
[3] Proprietary APIs

the VNSs to behave abnormally, even if the underlying hardware is fault-free. VMs are managed by cloud service providers and VNFs by the network service providers making it difficult for the traditional systems to deal with problems in virtualized services. Consider a situation where the virtual private networks (VPNs) of many customers do not work. In this situation, FP detection and localization would require investigation all the VMs, on which virtual core router is hosted, the VNF that is working as the core router, the virtual network interface controller (vNIC) with fast Ethernet or Gigabit Ethernet ports and even the physical machines. Many alarms and other markers would result, which have to be correlated.

The fault detection mechanism should be able to separate out the error conditions that do not constitute a fault from the ones that do. The fault conditions have to be further classified into *manifested* or *impending* so that further action can be accordingly taken. As the name suggests, the manifested faults are those that have already occurred and have affected the system in some way. The impending problems may not have manifested as faults yet, but may soon materialize with varying degree of severity. We discuss in this section how faults are classified according to their criticality, see in detail the sharing of FP responsibilities among different platforms and enunciate the FP problem that this work solves.

### 3.1 FP Issues and Their Criticality

As far as the virtual entities, VNFs, and their interconnections, are concerned, faults would happen due to algorithmic causes in the system software or in the application software. Faults in the application software affect the network functions or the links while those in the system software affect the VMs on which the VNFs are implemented. In the multi-domain scenario, besides the usual faults occurring in the carriers' networks, there would be issues due to the interconnection of networks. For example, non-provision of a sufficient number of inter-carrier interconnections at the Points of Interconnect (POI) would lead to congestion and failure of calls from one network to the other.

Some events that cause alarms may not always be errors. For example, degradation in service can happen with some devices underperforming or because of being underprovisioned. Since, in such cases, the devices may still not be faulty, there may be no alarm or just a minor alarm. The degradation of a service can be detected through notifications, counters or meters set up to count events at the virtual function or the service level. Many of these markers would be routine warnings. At the same time, some alarms may be automatically taken care of by the network's resilience features, i.e., by using the redundant units instead of the one not performing properly. Some of these alarms may be coded to indicate the severity of the events. The confusion does not end here. There could also be problems with the management platforms themselves – multi-cloud platform, MANO, or the OSS/BSS. In this tutorial, we confine ourselves to the faults of VNFs or of SFCs that affect the performance of VNSs.

ITU recommendation X.733 classifies the alarm events into the following four severity classes: Critical, Major, Minor, and Warning [6]. Critical alarms are caused when service to one or more users is totally stopped. If the service is highly degraded, but not stopped, then a major alarm may result indicating a condition that is preventing the service to be given as contracted. A minor event does not indicate present degradation, but if the condition is not corrected, it may cause a major fault to develop. A warning may be the most benign, but usually indicates an impending fault or performance issue which could eventually turn into a major fault. In addition to detection and localization functionalities, the predictive capabilities of the fault and performance monitoring system should be able to indicate what faults will develop and with what severity levels. Impending faults are, thus, an important source of concern. It would be very helpful to the carriers if they can identify which performance deviations or impending faults may potentially result in an FP problem that would require personnel and material to resolve.

### 3.2 Shared FP Responsibilities

The fault and performance related responsibilities are jointly exercised by the MANO, the MMCP, and the OSS. Their interrelationship in the context of VNSs was illustrated in Fig. 7. Table 4 summarizes the fault and performance related responsibilities of these management systems. As can be seen from the description, the functions of many systems overlap. For example, OSS and NFVO may both obtain information from the EMSs for knowing the status of VNFs. Similarly, the marker collection functions of VNFM and EMS overlap. The precise distribution of FP related functionalities would, therefore, have to be done in the implementations. Standardized reference points among the management systems would help with interoperability of management functions of different carrier networks. Some of the reference points have, either not been defined, or not completely defined. These issues make the fault detection and localization problems more difficult to handle as complete information is not available with any system. The framework that we have developed and described in this paper uses information from various

TABLE 4
SHARED FP RESPONSIBILITIES OF DIFFERENT MANAGEMENT ENTITIES

| Management Block | Fault and Performance Functions |
|---|---|
| **1. MANO** | |
| 1.1 NFVO | NFVO orchestrates services and monitors parameters required to meet SLAs. It manages the lifecycle of VNSs and uses available resources or requests additional resources to maintain the required performance. For handling FP issues, it gets VNF level alarms from VNFM and NFVI level alarms from VIM. It interacts with OSS to share measurement results and notifications regarding network services. Its functions overlap carrier OSS function. |
| 1.2 VNFM | VNFM interacts with VNF instances to obtain VNF related FP information like software inter-module communication failure. It also collects VNF-instance related NFVI information. It sends intelligence to NFVO for fault detection and localization. VNFM functionality overlaps with EMS functionality as both collect network function information. |
| 1.3 VIM | VIM collects alarms related to physical and virtual resources contained in NFVI. It forwards FP alarms to VNFM and NFVO for broader correlation and root cause analysis. The fault information may include VM crashes, virtual port malfunction, storage failure, resource unavailability, etc. |
| *2.* **MMCP** | MMCP keeps an inventory of and monitors all virtual compute, storage and networking resources from different CSPs. It logs analytics for VM related faults. It adjusts resources to changing workloads and maintains the required performance level. The division of FP responsibilities among MMCP, OSS, and MANO is still to be finalized. |
| **3. OSS** | OSS monitors network services and resources and detects anomalous conditions. It interacts with EMSs to obtain the status of network elements. In the virtual network service environment, it may directly or through NFVO get information about VNFs. It correlates alarms from various sources to localize faults and performance conditions. Its functions spread from VNS down up to the VNF level. |
| **4. EMS** | Each network function/device is monitored and managed by an EMS. They collect operational status and alarms from VNSs and forward them to the OSS and VNFM. |

management platforms to improve FP management.

## 3.3 FP Problem Statement

The FP problem of the carrier networks can be defined as follows:

*1) Detection of any condition that has already led to or could lead to degraded performance or failure*:

The reasons could be manifested faults, hidden faults or inconspicuous deviations. The goal of FP issue detection is to sense and notify impending or actual fault and performance issues.

*2) Identification and localization of manifested and impending faults:*

The goal of FP issue localization is to determine the root cause of the problem by identifying the resources that are malfunctioning or the severity with which they may malfunction in the future.

## 4. Discussion of works related to FP Management

During their operation, carrier networks produce large volumes of high dimensional data in the form of markers like alarms, notifications, observed behavior, warnings, counter values and measurement of performance indicators. These are discussed in some more detail in Section 6.1. The markers used by carriers are predominantly at the service and network function level. Any FP management system should take into account all the relevant markers to carry out the required functions. Traditionally handling FP issues as part of FCAPS has been considered a difficult problem as abnormal behavior has to be interpreted from large amounts of high dimensional and noisy data [34]. While the FP management has been well studied in traditional networks, work on this problem in virtualized network services in multi-cloud environment is scant. To be reasonably exhaustive, we examine the recent related work on FP management in four different categories: 1) Surveys highlighting the need for FP management 2) NFVI level diagnostics with or without active probing 3) Causality inference based diagnostics, and 4) statistical methods including those based on AI techniques. We'll discuss each of these briefly here and take up a more detailed study of the selected method in the next section.

1) Surveys highlighting the need for FP management in virtual environment.

The survey in [85] discusses research, development efforts and open challenges (among other issues like standardization) relating to Network Service Orchestration. The authors mention fault tolerance and performance among important orchestration functions. More specifically, in the next generation mobile networks the concept of network slicing can be used for fault and performance management. The authors also state that fault and performance is essential part of the effort of 3GPP directed towards standardization of the management of 5G networks.

In their related work in [86], the authors discuss fault management in the Software-Defined Networking (SDN) environment. Effect of various faults on network performance can be controlled by techniques such as system state monitoring, fault detection, localization and resolution, and fault tolerance mechanisms. The authors are of the opinion that most works handle fault from a partial perspective leading to incomplete and flawed solutions. According to their assessment, the design of suitable fault management solutions is indispensable for achieving good reliability of the network. There are many ways faults can arise in SDN. Most of these faults can be categorized as logic and coding errors. Software based data agents may contain functional defects that can cause network failures. Also frequent are malfunctions due to inconsistent rule installation because of hardware faults that may flip bits or because of attacks or misconfigurations. Many network troubleshooting tools like 'tracroute' and 'tcpdump' have proven to be inefficient for SDN environment. The authors discuss techniques like data agent testing, probe testing and interactive debugging as possible methods.

2) NFVI level diagnostics

We have seen previously that in VNSs, NFVI relates to the totality of hardware resources and the virtual compute, storage and networking resources created over these. The hardware component of the NFVI is in the domain of the CSP and generally inaccessible to the carriers. The methods in this category would rely on VM level alarms and metrics such as compute load or memory leak. These techniques thus rely on the monitoring and diagnostic techniques for cloud computing resources used for IT applications. An explicit or implicit assumption would usually be that the higher level alarms and other markers, e.g., those at network function and network service level, would usually have corresponding host level alarms which can be correlated to detect and possibly localize network function and service level manifested and impending issues. A correlation between telemetry information from the CSP and the higher level alarms in the domain of the carrier would have to be built up for diagnosing faults in the VNSs. Correlation of metrics with anomalies at the virtual layer has been applied by authors in [35]. The applicability of these techniques in a large distributed network needs to be studied.

3) Causal inference based methods

These methods are also normally applied on VM level alarms like high CPU load and insufficient memory availability. The expectation here is that determining the causal relationship among them would help to get to the root cause of FP issues at the network function and service levels. The process involves looking for anomalous behavior based on VM level alarms, correlate alarms in pairs or clusters, determine causality, i.e., the effect of one alarm on the others and attempt to build causality templates that could be used for future alarms. The complex architecture and dynamics of NFV pose significant challenges from the point of view of causality inference. For instance, in [36], the authors carry out analysis of uncorrelated alarms in order to recover the pairwise causal relationship between them. To take care of the fact that higher-level faults (e.g., VNF or VNS levels) do not only depend on the pairwise relationship among VM level alarms, the authors propose clustering to infer multi-way causality templates. The patent documentation at [37] goes a step further and uses alarm data from different layers (e.g., NFVI and VNF). It takes into account the temporal proximity and the order of the alarm types in the clusters to make causality templates.

4) Statistical and AI-based methods

The large volume of operational data generated in an operational telecommunications network could emanate from within one layer or across multiple layers and possibly contain many different types related and unrelated markers. In such a complex environment, it would be difficult to analyze the available data to produce information that can be used to manage FP issues. This situation, thus, creates a perfect set up for removing humans from the loop and resorting to machine intelligence. In this category, there are methods based on machine learning and deep learning that could be used for the detection and localization of FP issues.

There has been extensive work on performance modeling systems for distributed Internet applications of the pre-NFV era, notably TIPME (2000) [38], Pinpoint (2002) [39] and Magpie (2003) [40]. TIPME helps in identifying and eliminating causes of long response times. Pinpoint uses data mining to correlate the behavior of each active user request with the past failures and successes to determine failed components. Magpie works on individual user requests and compares the observed behavior, with saved normal models, to identify anomalous requests and malfunctioning components. Recently, the 'mPlane' consortium of European telecom companies and academic institutions, has worked on developing a measurement plane for Internet and CDN (2013-2016). The core of the project is 'mpAD-Reasoner,' which uses machine learning to detect anomalies involving multiple flows or users. It compares the current distribution with stored average distributions [18]. In [88] the authors provide concepts related to into cross stratum optimization to meet the QoS requirement. The work in [89] extends the idea to multi-dimensional resource optimization optimal networks in 5G domain. The work, though does not directly focus on fault detection and localization nevertheless provides insight multi-stratum resources optimization (MSRO) in NFV in the cloud environment.

It has been shown that learning methods give a way to relatively easily learn structure in the data and draw inferences [41]. Shallow machine learning algorithms, characterized by a single convolution stage, are suitable for cases where a large amount of labeled training data, including normal and fault cases, are available. They can derive intelligence from data and do not depend on experts to build complex interacting rules to derive patterns or models. Even dependencies, which cannot otherwise be explicitly modeled, can be learned. These advantages make them attractive for handling FP problems. In FP applications, machine learning methods can not only be trained with historical fault and performance data but can also be made to improve themselves as they operate and encounter new situations. This makes the machine learning systems, adaptive and intelligent and when they have been adequately trained, as they can generalize well from the training environment to the real-life situations. Use of different algorithms has been reported for detection and localization. We shall see more about this method in the next section.

## 5. AI-Based Handling of FP Issues

Researchers' interest in AI-based machine intelligence for the identification of FP issues dates back to the era of expert systems [42] [43] [44] [45]. During the intervening decades, the carrier networks have undergone changes in technology and form, but the interest in intelligent fault handling has persisted. We look at AI as a way to empower machines to mimic and outperform human intelligence. Machine learning is a subset of AI, chiefly consisting of statistical techniques that allow machines to exhibit behavior that improves with learning. Deep learning is a way to implement machine learning using neural networks with more than one level of non-linearity. When using neural networks for difficult tasks, complex relationship among variables modeled with several levels of non-linearity improves the generalization process [46] [47] [48].

VNSs are a new development and their deployment over multi-cloud is still to be explored fully. Many of the AI methods developed for intrusion detection have been explored, with varying degrees of success, for managing the FP issues. Some researchers have applied AI methods directly to the fault detection and, to a lesser extent, to fault localization. A very important reason for exploring AI for the problem of FP management for cloud-based NFV is the intractability introduced by the known gaps in the NFV specifications. Interaction among multiple domains, especially between the legacy OSS and the MANO and the legacy OSS and the MMCP [19]. ETSI supported proof of concepts (POC) have also resulted in highlighting the gaps in the NFV framework and carved out research work for the future. The present NFV framework, rather simplistically, assumes that VNFM will be primarily responsible for fault management actions. In real implementations, there will be multiple layers of cooperating fault managers. The OSS tackles customer fault reporting and management, which interacts with the EMS and NFV-MANO for the element level and VNFM level inputs, respectively. Besides, state change events for fault management actions have not been defined which are required for avoiding conflicting multi-layer actions and also an escalation from lower to higher layers. In this situation the learning methods of AI make the best use of the features learned from the available markers and can assist in FP management.

The authors in [49] use Artificial Neural Networks (ANN) for one and two alarms simulated scenarios. They show that in a simulated environment ANN provides better performance in comparison with the other implemented methods. The researchers in [50] propose a system for fault analysis and prediction in the telecommunications access network for the Rijeka area of Croatia. The Authors in [51] have used temporal decision trees for fault prediction in telecommunications networks. As per findings in [52], fuzzy cluster means can be used to classify network faults. The current research indicates the possibility of advancing the state-of-the-art in FP management through deep learning structures.

In [53], the authors use the Random Forest machine learning method to detect performance degradations in the VNFs. However, these researchers have chosen to rely on virtual resource layer level features data like CPU consumption, disk I/O, and free memory based on their suitability to computing systems. Evaluation has been carried out in a centralized IMS system. Application of the proposed method to a highly distributed multi-domain network has not been reported.

The authors in [54] have worked on the premise that underlying all the VNF failures are the NFVI level failures like disk I/O or memory usage. They propose Self Organizing Map (SOM), a type of unsupervised learning neural network, for clustering the statistical data and analyzing them to detect the faults. In [55], the author mentions that machine learning algorithms are expected to detect invisible failures and anomalies. However, more work is required to validate them.

Machine learning can be used for root cause analysis and failure localization in optical networks [87]. The authors of this work discuss fault management including detection of degradation and localization of faults. According to them, restoration procedure can be initiated in cases where traffic has been affected by a fault. However, early detection of degradation allows remedial action to be taken to prevent network downtime. The paper does not delve into specific techniques for detection and localization and none of the techniques have been evaluated.

We now discuss in more detail the architecture and design of an AI-based FP management framework for NFV deployment in the multi-cloud scenario.

## 6. Description of the Proposed FP Management Framework

FP management in the cloud-based VNS has to be a collaborative process among the elements constituting the VNS and the management systems involved in creating and managing the service. VNSs impose new requirements on the FP management system. Some of these requirements include mining of large volumes of high dimensional, multi-source and multi-layered data. This is, in some parts, imposed by the necessity of making up for the gaps in the 'NFV on clouds' specifications in relation to the FP management and prediction of impending problems. The proposed framework takes care of these requirements.

The data generated by an operational system is large and high dimensional. In such a case, it would be very difficult to capture the intricate relationships among the features (e.g., the location of the fault, resources involved, markers produced, etc.) and the corresponding labels (faulty, non-faulty, impending fault, manifested, fault-severity, etc.) through traditional methods. It is being increasingly realized by researchers and echoed by standards bodies as well that a predictive approach to fault detection and localization, based on methods that learn the structure of and relationship among features from the data itself are more likely to succeed [9][41] [56] [57].

As a case study, we will also discuss the work that we have carried out, to tackle the problem described above, using deep learning and shallow learning methods [15] [58]. It has been found that a hybrid framework consisting of a combination of shallow and deep learning models could be used for detection and localization of FP issues as well as predicting severity levels of impending faults with a high level of accuracy.

### 6.1 Markers and Metrics for Fault Detection and Localization

We have introduced markers before as indicators produced by an operational network and measurements taken by the operations staff. There are a large number of markers that are directly or indirectly related to the occurrence of an FP issue. These markers become important features in our datasets. Events, that produce these markers, relate to communication, QoS, processing, equipment, and environment. Of course, not only each FP issue would usually have multiple markers, but also many of the markers would appear in more than one type of issue. Also, at any given time the markers produced may be a result of more than one FP issue. Thus, there is a complex relationship between the markers and the FP issues. This would usually mean that when using machine learning for fault detection and localization, feature engineering, i.e., selection of appropriate markers would be required to get better results. However, deep-learning models, are able to extract relevant features automatically, without human intervention. Some of the markers related to mobile, fixed and broadband networks are given in Table 5.

TABLE 5
LIST OF MARKERS FOR DIFFERENT CARRIER SERVICES

| Broadband | Mobile Network | Fixed Network |
|---|---|---|
| Intermittent connection | Handoff alarm | Earth on a limb |
| Low data rate | BTS power alarm | No dial tone |

| | | |
|---|---|---|
| NPOT | Packet loss counter | Loop resistance |
| Repeated training | Backhaul congestion | Line card port faulty |
| LAN lamp off | RX noise floor | Permanent ground alarm |
| Line noisy | Frequency error | Distribution cable fault |
| Port mismatch | Antenna tilt | DP fault |
| No ping | C/I ratio | Insulation measurement |
| ADSL lamp flashes | Signal strength | MDF fuse blown |
| No line sync | Radio link failure | Handset fault |
| Browsing issues | Cell site failure | Dis on one limb |
| Micro-Filter Faulty | Interference level | No incoming calls |
| No Communication | CQ indicator | Drop wire fault |
| Dropouts | Virtual eNB capacity | Ringtone fault |
| No authentication | Hypervisor alarm | Message fault |
| vRouter failure | Registration failure | Delayed dial tone |
| BTS: Base Transceiver Station, C/I: Carrier to Interference, CSSR: Call Set-up Success Rate, MDF: Main Distribution Frame, MU: Multi-User, eNB: eNodeB, NPOT: No Power in Optical Network Terminal, XCOA: Contact with AC, CQ Indicator: Channel Quality Indicator | | |

The metrics used by carriers to measure the health of the network provide important information about the FP problems at the macro level. Use of these as features in the dataset would help learning algorithms to narrow down the scope of localization effort. According to ITU Recommendation regarding the QoS criteria and parameters, a number of basic aspects have to be considered while identifying measurable metrics of service availability [59]. ETSI documents on service availability [56] and on service quality [60] mention metrics that need to be collected and analyzed. The ETSI group specification on service quality metrics [61] recognizes that it is important to have an objective and quantitative metrics to assist in identifying problems when they arise and provide good service to the consumers. Examples of metrics, and their realistic values (where applicable), from an actual network [62] are given in Table 6. We will see how these markers and metrics are used in our framework in later sections.

TABLE 6
METRIC FOR NETWORK AVAILABILITY AND RESILIENCY

| Metric | Typical Value | Metric | Typical Value |
|---|---|---|---|
| **Broadband Network** | | POI congestion | <0 .5% |
| Packet loss | < 1% | Assistance response | > 95% |
| Customer PoP to Internet exchange latency | <120ms | **Mobile Network** | |
| Peak international bandwidth utilization | < 90% | BTS total downtime | ≤ 2% |
| Connection data rate availability | > 80% | Traffic Channel Congestion (TCH) | ≤ 2% |
| Average throughput for packet data | > 90% | Call Drop Rate (CDR) | ≤ 2% |
| Latency (audio) | <150ms | Call Set up Success Rate (CSSR) | ≥95% |
| **Fixed Network** | | Paging channel congestion | ≤ 1% |
| Fault incidences | < 5% | Signal strength in vehicle | ≥ 85dbms |
| Call completion rate | > 55% | | |
| PoP: Point of Presence, BTS: Base Transceiver Station, POI: Point of Interconnection | | | |

## 6.2 Potentially Applicable AI Techniques

There are quite a few AI techniques, involving machine learning and deep learning that are potentially applicable to the problem of detection and localization of FP anomalies. Following the practice of applied machine learning researchers, we designate models with a single layer of non-linearity, e.g., Support Vector Machine (SVM) and neural network (NN) with one hidden layer, as shallow structures or shallow machine learning architectures and the models with more than one layer of non-linearity, e.g., stacked autoencoders are referred to as deep structures or deep learning architectures [63] [64] [65]. It is common for shallow models with a linear hypothesis to have O($n$) prediction time complexity and the training time complexity of O($l^2$+$n^3$) where $l$ denotes the size and n is the number of features in the dataset used. However, with such models, approximation errors are large for the high dimensional and large volume of data that are usually associated with the FP problem. Thus, if the data is not linearly separable then kernels could be used to map data into a higher dimension where it shows linear properties. This implies that linear models like SVM could be applied to the new space. This kernel trick reduces the approximation errors at the cost of higher complexity of the training time which is O($l^3$ + $l^2n$) and prediction speed of O($ln$). Of the prevalent shallow machine learning architectures, supervised methods (where each training example consists of the feature vector as well as a label) such as SVM and Random Forest (RF) are considered useful for diagnostic applications [66]. Another supervised learning technique, Bayesian Network (BN), has been applied to FP management in the industrial settings. Our preliminary exploration of these methods with small datasets has shown that SVM and Alternating Decision Tree (ADT) produce comparable and encouraging results for the detection problem. We will discuss the evaluation results in the next section.

In deep learning, increasingly improved features are learned as the hidden layers are traversed. Learning of complex features and structure in the data can be broken down into simpler tasks performed at many levels. This way, deep learning can achieve low generalization errors, even for functions otherwise difficult to represent [67]. Lately, better results than SVM have been achieved with deep neural networks in a number of important applications [68] [69]. A key advantage of deep learning over shallow learning is the automatic extraction of high-level features. Each algorithm that we have used is briefly described here. For more details, readers may consult the references mentioned.

*1) Support Vector Machine (SVM):* Geographically dispersed elements of the network may generate similar or different markers at different locations, for example, at the carrier's OSS location or the NFV provider's MANO location. The information contained in these markers is non-unique across the domain of faults and performance issues. The SVM classifier can analyze the data and learn inherent patterns, which are otherwise not evident to the human senses. It works by finding optimal hyperplanes that separate different classes in a given labeled dataset. Once trained, it can classify unseen data. References at [70] [71] give a more detailed description of SVM. As we have use SVM in our framework, we mention some more details of parameter C, $\Upsilon$ and $\epsilon$ that require careful selection to minimize prediction errors. As the exact solution is impractical, precision $\epsilon$ is used to indicate the error insensitive tube around the decision boundary in which the errors are ignored. The aim is to minimize $||\omega||^2$ which is equivalent to maximization of the margin between the classes. The constant C determines the tradeoff between the flatness of function learned and the amount of error allowed above $\epsilon$. A low C makes the decision surface smooth; a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors. We choose how significantly the misclassifications should be treated and how large the insensitive loss region should be, by selecting suitable values for the parameters C and $\epsilon$. The data $\boldsymbol{X}$ is projected to a higher dimension using function $\phi(\boldsymbol{X})$. Poor generalization and computational complexity that may result from projecting data to higher dimensions can be avoided by the use of a kernel function that maps the input feature space of dimension *d* to a higher dimensional space in which the relationship becomes linear. In our studies, we have found that the performance of the Radial Basis Function (RBF) kernel performs better than others. The RBF kernel has the form given below. Here, $\mathbf{x_i}$ and $\mathbf{x_j}$ are two sample feature vectors, and $\Upsilon$ is the parameter that sets the spread of the kernel.

$$K(\mathbf{x_i}, \mathbf{x_j})=\exp(-\Upsilon||x_i-x_j||)$$

In all cases where SVM had been used, these parameters had been arrived at by a grid search.

*2) Alternating Decision Trees (ADT):* This method combines Decision Trees with Boosting. The ADT is different from normal decision trees as it has predictor and test nodes alternately, while the normal decision tree has just test nodes with each branch representing an outcome of the test. Another difference is that while each leaf can only be split once into two, in ADT each part can be split multiple times. This increases the accuracy of classification/regression. The splitting criterion could be impurity based like information gain or Gini index or based on a statistical test like chi-square. Boosting, on the other hand, brings in performance-enhancing capabilities. However, it adds more test and predictor nodes. The complexity is quadratic in boosting iterations, but can be reduced by using a suitable heuristic [72].

*3) Random Forest*: Among supervised learning algorithms of its class, the Random Forest (RF) is a classifier that is likely to give more accurate results. It proves to be efficient and robust in many use cases with large databases. It can help in feature selection by estimating the relative importance of the predictor variables. This is done by selecting an impurity measure like entropy and measuring the contribution of each feature. Another very useful feature is that it does not need separate test data or any cross-validation. The Out-of-bag error (OOB-error) gives an unbiased estimate of test or classification error [73].

*4) Deep Learning using Stacked Sparse Autoencoder*: An autoencoder is a neural network, which has an input layer, an output layer, and one or more hidden layers. It learns the feature of a dataset in an unsupervised manner (i.e., the training examples are just feature vectors with no labels). Such a model reconstructs the input values at the output with accuracy depending on how well the features are represented by the hidden layer(s). A sparse autoencoder (SAE) contains a hidden layer with a smaller number of neurons than the inputs. Thus, the high dimensional inputs are mapped to a lower dimension forcing them to learn the best representations of the given features. Extraction of features takes place according to their relative importance. More than one sparse autoencoders can be put in tandem to construct a stacked sparse autoencoder (SSAE). Training of the stacked autoencoder is done in a layerwise greedy manner. The first layer is trained with the input data $\mathbf{x}$ to obtain weights $\omega$ and bias $b$ for the hidden units such that the output $k(f(\mathbf{x}))$ is as close to the input as possible, i.e., minimizes the loss function $Ø(x, k(f(\mathbf{x}))$ [74]. The L2 norm (mean square error) is often used as the loss function. The primary feature activations of the first hidden layer are then used as input to the second hidden layer and so on. Since the L2 norm may not reduce the error to zero, a sparsity penalty term is added to constrain the neurons to be mostly close to zero. The training criterion can be written as $Ø(x, k(f(x))) + \Omega(h)$, where $\Omega(h)$ is the sparsity penalty.

If we consider an SSAE with n layers then the weight and bias parameters for the m$^{th}$ autoencoder can be written as $\omega^{(m, 1)}$, $\omega^{(m, 2)}$, $b^{(m, 1)}$, $b^{(m, 2)}$. The encoding step in the feed-forward direction for each layer k of the stacked autoencoder is given by:

$$h^{(k)} = f(x^{(k)}) \quad (1)$$

$$x^{(k+1)} = \omega^{(k,1)}h^{(k)} + b^{(k,1)} \quad (2)$$

The decoding stack of each autoencoder is run in the reverse order

$$h^{(n+m)} = f(x^{(n+m)}) \quad (3)$$

$$x^{(n+m+1)} = \omega^{(n-m, 2)}h^{(n+m)} + b^{(n-m, 2)} \quad (4)$$

Then, as the layer-wise training proceeds, each successive layer learns increasingly more and more useful features with the innermost layer $h^{(n)}$ giving a representation of the input in terms of the most compressed and useful features for the input of higher dimension. With appropriate settings of the parameters, the compressed layer reconstructs the original

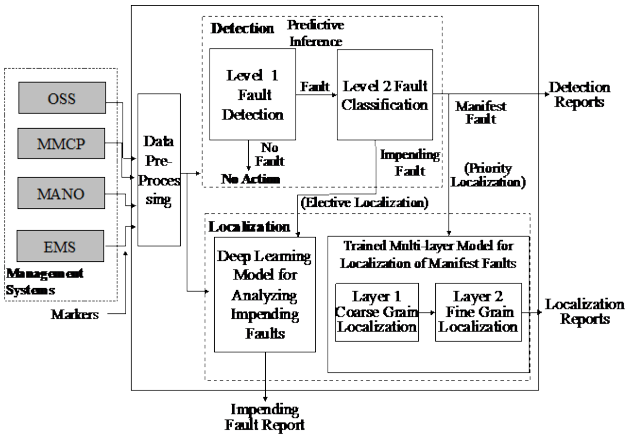


**Fig. 8.** The FP management framework

input with good accuracy. Good reconstruction performance helps in achieving good prediction. For prediction of fault classes or severity of impending faults, a layer of Softmax classifier replaces the decoder layers with $h^{(n)}$ forming the input to this layer. Softmax regression can be used for multi-class classification as it gives probabilities of output being close to the target value in the range 0 to 1 with the sum of probabilities being 1.

TABLE 7
MACHINE/DEEP LEARNING ALGORITHMS FOR THE FP PROBLEM

| Algorithm | Advantages | Watch out for |
|---|---|---|
| **Support Vector Machine** | • Works well for the detection problem.<br>• Works with linearly separable as well as non-linear feature space (with RBF kernel). | • Select kernel function and fine-tuning of parameters.<br>• Select the cross-validation method carefully.<br>• Long training time with the big dataset. |
| **Random Forest** | • Works well for the detection problem and localization of manifested faults.<br>• Works for binary as well as multi-class classification.<br>• Less prone to overfitting.<br>• Handles non-linearity.<br>• Handles categorical features.<br>• Handles high dimensional spaces and a large number of examples. | • Fine tuning of parameters like the number of features in any tree, number of trees in the ensemble and leaf size.<br>• Watch out for classification time and complexity of the model. |
| **ADT** | • Works well for the detection problem.<br>• It has the speed of a decision tree and is robust to noise and missing values.<br>• It can be used for mixed categorical and numerical data.<br>• It helps in finding significant features. | • Must be used carefully to avoid overfitting.<br>• Keep control of parameters like depth and number of features to split on. |
| **Autoencoder /Stacked sparse autoencoder** | • Useful for localization of impending problems.<br>• It gives better control over quality.<br>• With the appropriate number of layers and neurons, it performs better than the shallow algorithms | • Sensitive to number and size of layers.<br>• Careful fine-tuning of sparsity and regularization parameters is required. |
| **SoftMax** | • Used as the last stage of stacked autoencoder in the localization problem.<br>• Trained in a supervised manner.<br>• It can do binary as well as multi-class classification.<br>• It can be used for prediction of faults, severity, etc. | • Watch for bias due to the distribution of data.<br>• If sufficient labeled data are available fine-tuning by backpropagation may improve results. |

Table 7 summarizes the machine learning and deep learning techniques useful for NFV-Cloud FP problems.

## 6.3 Framework for FP Detection and Localization

We propose learning models that have predictive and deductive properties to meet the FP requirements of virtual network services. All the markers available from the management platforms, i.e., the runtime monitoring and measurements, alarms, notifications and warnings, configuration changes, and environmental factors are used along with machine learning models trained with historical data to draw inferences about the manifested performance and fault issues. Additionally, the capability of deep learning to map the intricate relationship among the features has been used to predict the impending faults. The framework shown in Fig. 8 consists of three main sub-systems: Data pre-processing, Detection and Localization. Data pre-processing involves collation and normalization of the dataset to remove biases. The pre-processing policy may also involve the reduction of features based on some criterion like correlation with the labels. In the training mode, the available dataset is split into training and test datasets, which are used to train and test all the models. During operation, the marker data is run through the framework to detect and localize problems.

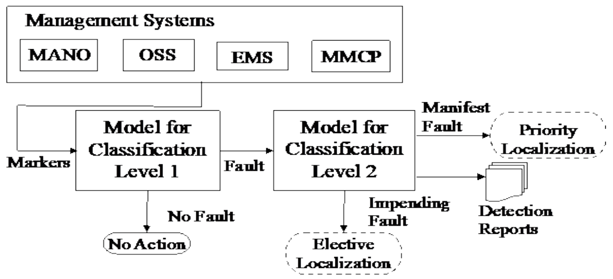


**Fig. 9** The detection subsystem

The first part of the FP problem, i.e., detection is essentially a two-stage binary classification problem that first classifies the outcome as ‘normal performance’ or ‘abnormal performance’ or alternatively as ‘fault’ or ‘no fault’ classes. Then for the ‘fault’ or ‘abnormal performance’ cases, it decides whether the problem is *manifested*, i.e., it has already occurred somewhere in the network in some form, or impending, i.e., it might happen in the near future. We shall see in the next section why a two-stage model is better in this case. It is important for the detection models to have good accuracy as manpower and material resources are committed for rectification of detected faults. This is particularly important, as the presence of alarms does not always indicate a fault.

The second part of the FP problem is the localization of the detected faults. Localization of manifested faults is taken up on priority while for the impending faults it is elective, nevertheless important. For the manifested faults, the model uses a multi-layered localization strategy using machine-learning classification models. At Localization Layer 1, the broad category of the manifested fault is determined, e.g., network performance problem. At Localization Layer 2, the system makes a finer identification of the problem to assist in the identification of the root cause of the problem, i.e., malfunctioning resources or resources suffering from performance degradation. In the case of the *network performance* class of problems at Layer 1, the model at Layer 2 may narrow down the classification to a high bit error rate as the cause (Table 8). For the impending faults, a deep learning strategy uses the markers to predict the severity and location of faults.

**Algorithm 1:** Detection Levels 1 & 2

```
1: procedure detect_level1 (X)
2: #fault/no-fault classification
3: {$p_d$} ← values of hyper-parameters for the chosen model
4: use trained model for detect_level1 with X, {$p_d$})
5: if 'fault' is true
6:     call detect_level2 (X')
7: produce detection report
8: procedure detect_level2 (X')
9: # classify as manifested/impending and call localization
10: {$p_d'$} ← values of hyper-parameters for the chosen model
11: use trained model for detect_level2 with X,Y, {$p_d'$}
12: if manifested is true
13:    call manifested_localization (X") #defined in Algorithm2
14: elseif impending is true
15:    call impending_localization (X") #defined in Algorithm2
```

The massive amount of observations generated by the operational system can also be used for trend analysis to indicate abnormal behavior and degenerating devices.

## 6.4 The FP Detection Subsystem

The detection sub-system of the FP management framework is shown again in Fig. 9. In the two-stage implementation for detections, both levels use the shallow machine-learning models. As mentioned before, these models are trained on historical data consisting of FP events, resulting markers including the severity levels and the fault clearance description that the maintenance staff has entered after rectifying the fault. The cases, where no action is required or the fault is transient and corrects itself, are labeled as 'no-fault.' In the case of an actual fault, the nature of the fault and its actual clearance is indicated. The trained model can then take markers resulting from new events as inputs to decide at Level I whether the conglomeration of markers constitutes a fault. If it does, then the model at Level II uses the available information to decide whether the fault is impending or manifested. The use of markers from many management platforms may introduce redundancies, as a good amount of similar information may be available from OSS and MANO. However, making use of redundant data makes up for the gaps in communication among various management platforms.

**TABLE 8**
COARSE (LAYER1) AND FINE (LAYER 2) CATEGORIZATION OF FAULTS

| Layer 1 Fault | Layer 2 Fault | Markers |
|---|---|---|
| Network Performance | Traffic and Beacon Channel plan | Bad receive quality, Call drop at the cell boundary, Link degradation |
| | Handoff parameters setting | |
| | Bit error rate | |
| | High paging discard rate | |
| Security | Denial of Service attack | Client Authentication failure, Call initiation failure |
| | Home Subscriber Server Failure | |
| Virtual Resource | VM Fault | Network function failure alarm |
| | Hypervisor fault | |

However, the occurrence of multiple faults, the overlap of markers among faults and conflicting markers may render the task of detection difficult. If our detection sub-system is effective and can correctly segregate the conditions, then localization has better chances of succeeding. A two-level model for detection helps in filtering out a large number of 'no-fault' cases at level 1 so that level 2 is largely applied to the 'fault cases.' This makes classification better and faster.

Algorithm 1 describes the process succinctly. **X** is the vector of predictor variables. Hyper-parameters {$\mathbf{p_d}$} and {$\mathbf{p_d'}$} pertain to detection models at the two layers, {$\mathbf{p_s}$} and {$\mathbf{p_n}$} are for models at the Localization Layers 1 and 2 and {$\mathbf{p_i}$} are for deep learning model for impending faults.

The procedure *detect_level1* at line 1 takes the feature vector of a new event and populates the hyper-parameters (line 3). The trained machine learning model is used to predict labels. If it is 'fault' condition then detect_level2 is invoked (line 8) which uses another trained model to classify the fault as 'manifested' or 'impending'. Thereafter, the appropriate localization module is called (line 13 or line 15) to handle the manfiested fault localization or the impending fault localization. Use of X' and X" indicates the possibility of curating the feature vector used with the corresponding model. This algorithm also outputs the detection report, which includes fault cases as well as the type of faults.

## 6.5 The FP Localization Subsystem

The 'Manifested Fault' are those that have made themselves evident and many of them could be major or critical, threatening to seriously cripple the network service from which they originate. These faults cannot be allowed to persist and need to be handled on urgent basis. Since many faults may propagate and show up elsewhere in the network, the localization process has to cut across layers and domains to identify the faulty devices, links, or software correctly. The 'Manifested' faults are localized by a multi-class, two-layered model shown in Fig. 10. In cases of impending faults, the localization functionality requires prediction of the severity of the developing faults. A deep learning model consisting of stacked autoencoders has been used for this part. The stacked autoencoder was introduced in Section 6.2.

Algorithm 2 explains the localization function. X, Y and the set of hyper-parameters **{p}** have the same meaning as before (sparsity parameters have been explained in Subsection 7.4). Details of the models and strategies for the *manifested and the impending* fault *classes* are explained below.

The procedure *manifested_localization* (line 1) uses procedure localize_layer1 (line 4) to determine the broad category of manifested fault. Depending on the category determined, it calls the *localize_layer2* with corresponding parameters. For each category at Layer 1, the Layer 2 may have a specifically trained model. For Impending fault localization the procedure *impending_localization* (line 17) calls the deep learning model with the required parameters. Let us discuss a little more about the manifested and impending faults.

### 6.5.1 Manifested Faults

During operation, all the FP issues classified as 'Manifested' pass through the two layers. At Layer-1, the model works as a multi-class classification model that classifies the faults into one of the several broad categories of FP issues. Table 8 gives examples of three such categories, 'Network Performance,' 'Security' and 'Virtual Resource.' The model at Layer-2 is also a multi-class classification algorithm that localizes the FP issue at a finer granularity (e.g., a device, interface, or link) within the broad category predicted at Layer 1. The localization sub-system produces localization reports that can be used by the maintenance staff to carry out the rectification work. For the multi-class classification with SVM, we chose to work with simple models like One vs. One (OvO) and One vs. All (OvA) [24]. We eventually selected OvA since it provided more accuracy and was comparable to OvO in training and actual operations. In the OvA approach, for the ith classifier $f_i$, the examples can be classified with $f(x) = \arg\max_i f_i(x)$, i.e., choose the class that classifies the example with the maximum margin

### 6.5.2 Impending Faults

In traditional systems, in the absence of predictive analysis, preventive maintenance is relied on to catch issues early. In the proposed framework, localization of impending faults consists of predicting the severity and location of the fault. An operational network produces data continuously. In a stable operational network, most of the examples would constitute normal data with markers indicating anomalous conditions interspersed sporadically. While our data has more than 800 features, any anomalous condition would present <5% of these! In other words, the data are quite sparse. Impending faults may also contain previously unseen faults.

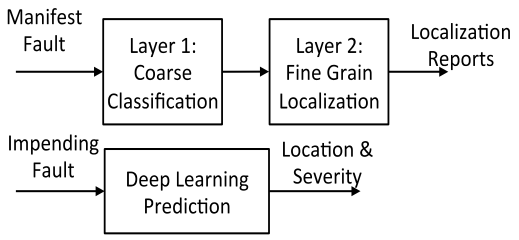


**Fig. 10.** The localization process

Thus, while manifested faults are manageable with shallow models, impending faults have been tackled with deep learning. We have used Stacked Sparse Autoencoder (SSAE) (a type of deep neural network). A single SAE contains an input, an output, and a hidden layer. With an under complete hidden layer, the autoencoder learns the most useful individual features as well as creates composite features. The advantage can be accentuated with stacking a number of autoencoders and carefully designing the hidden layers [75].

**Algorithm 2:** Localization Layers 1 & 2

```
 1: procedure manifested_localization (X)
 2: # Coarse grain localization
 3: {$p_s$} ← values of hyper-parameters for the chosen model
 4: call localize_layer1( X,{$p_s$})
 5: # fine grain localization with the appropriate model
 6: if class_category ==1
 7:     {$p_1$} ← hyper-parameters class_category 1
 8:     call localize_layer2(X",{$p_1$})
        …
 9: if class_category==7
10:     {$p_7$} ← hyper-parameters class_category 7
11:     call localize_layer2(X",{$p_7$})
12: produce localization report
13: procedure localize_layer1(X,{$p_s$})
14: use trained model localize_layer1 with (X,{$p_s$})
15: procedure localize_layer2(X",{$p_n$})
16: use trained model localize_layer2 with (X, {$p_n$})
17: procedure impending_localization (X)
18: {$p_i$} ← parameters neurons, sparsity parameters
19: use deep_learning_model (X,{$p_d$})
20: produce impending fault report
```

Fig. 11 shows the stack of three sparse autoencoders used in this work: the input layer (x), an output layer (p) and three hidden layers consisting of paired encoders and decoders. The colored neurons show three corresponding pairs of encoders and decoders. By reducing the size of hidden layers, the output is made reliant on increasingly lesser but richer features. Such a network can be trained in an unsupervised mode to reconstruct input data at the output with good accuracy. These networks can be tuned well for sparse data by using parameters like sparsity regularization and sparsity proportion as discussed in the evaluation section.

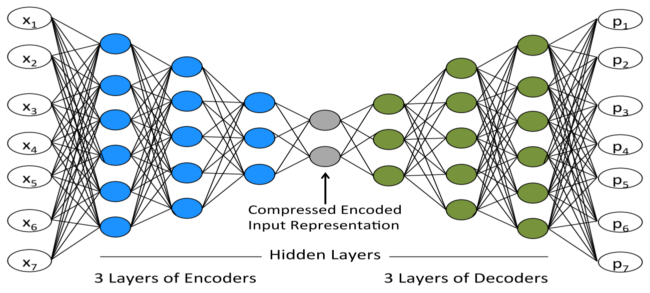


**Fig. 11.** Stacked sparse autoencoders

We train our model to have a good reconstruction of the input at the output (decided by the L2-norm), with unsupervised data, in a layer-wise greedy method (one hidden layer at a time). A model that reconstructs well also gives good predictions [28]. During training, features (z) learned by each hidden layer are input to the next layer. Pairs of {weights, biases}, viz., ($\omega_1$, $b_1$), ($\omega_2$, $b_2$) and ($\omega_3$, $b_3$), are learned in achieving good reconstruction.

$$\{\omega_k, b_k, \omega_{k'}, b_{k'}\} = \begin{cases} \text{argmin}\{L2\_norm(x, x'), k=1 \\ \text{argmin}\{L2\_norm(z_{k-l}, z_{k-l'}\}, k>1 \end{cases} \quad (1)$$

$$z_1 = f(\omega_1, x) \quad (2)$$

$$z_k = f(\omega_k, z_{k-1}), k>1 \quad (3)$$

After achieving good reconstruction of the input, the decoders are removed, and a prediction layer is added in tandem with the encoded representation layer (Fig. 12). Softmax assigns decimal probabilities to each class in a binary or multi-class problem. These decimal probabilities must add up to 1. This additional constraint helps training converge more quickly than it otherwise would. In simple terms, the Softmax function can be written as

$$F(y_i) = \exp(y_i)/\textstyle\sum_{j=1,k} \exp(y_j), \quad i=1, 2, \ldots, k \quad (4)$$

Softmax uses the rich features from the encoded layer of the stacked autoencoder to learn its weights $\omega_4$ and biases $b_4$. Training of Softmax is done in a supervised manner using the labeled examples available. $\omega_4$ are the weights for minimum prediction mean square error (MSE). It produces predictions y' for the given labels y. Thus, for labels y and its prediction y' we have,

$$\{\omega_1, \omega_2, \omega_3, \omega_4\}=\text{argmin}\{L2\_norm(y, y')\} \quad (4)$$

After the Softmax classifier has been trained in a supervised manner, the whole model is fine-tuned using back-propagation and simultaneous adjustment of weights of all the layers to minimize the mean square error in the labeled test datasets [20].

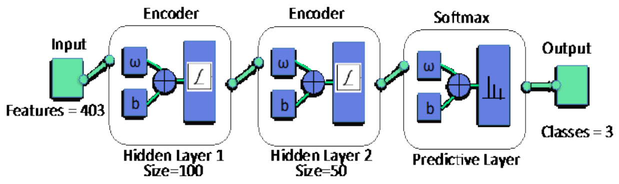


**Fig. 12.** The stacked encoder used for prediction

## 7. Evaluation of the model

In this section, we will discuss how the FP detection and localization framework, proposed in Section 6, has been evaluated. More specifically, we will see the training dataset used, curation of data, and the performance of the trained models for the unseen events. Curating may involve one or more of the following activities to improve the outcomes: feature pre-selection using some kind of technique to correlate features with the labels, cleansing of data, pruning or integration, synthesis or analysis of features.

### 7.1 Training Datasets

Having access to good quality datasets is important for proper training of the learning models and their predictive performances [75]. Records like fault dockets, switch room logs, outdoors logs, personal records of maintenance staff and fault closure reports contain a vast amount of information about complaints, faults, test results and restoration details of telecommunication networks. However, assembling a useful dataset from these primary data is not an easy task. Since network fault and performance datasets are not easily available, researchers commonly resort to either proprietary datasets that are not publicly available or generate synthetic datasets [50] [52]. We have used in our studies the real network FP dataset pertaining to faults and disruptions in telecommunication carrier Telstra's network [77]. The dataset, as available, is split into a number of sub-datasets, each containing different information derived from the logs. These sub-datasets give *event_type*, *log_feature*, *resource_type* and *severity_type*. They are related through the "*id*" column that acts as the key field and also conveys the timing information. It can be used in innovative ways to improve predictions based on the dataset. The *event_type* is the type of fault or performance incident. Any anomalous situation may have up to 5 different events associated with it. The *resource_type* gives the affected virtual resources. The feature *fault_severity*

TABLE 9A
TRAINING DATASET

| id | location | fault_severity |
|---|---|---|
| 4757 | location 508 | 0 |
| 1635 | location 257 | 1 |
| 1181 | location 116 | 0 |
| 7274 | location 830 | 1 |
| 4311 | location 704 | 2 |
| 1226 | location 1089 | 2 |
| 1475 | location 653 | 0 |
| 3304 | location 1099 | 1 |
| 9012 | location 975 | 0 |
| 9928 | location 1019 | 2 |
| 1001 | location 696 | 0 |

TABLE 9B
TEST DATASET

| id | location |
|---|---|
| 13484 | location 922 |
| 12392 | location 184 |
| 2322 | location 1019 |
| 567 | location 734 |
| 4436 | location 236 |
| 12156 | location 124 |
| 7508 | location 858 |
| 6184 | location 707 |
| 12213 | location 763 |
| 6458 | location 1100 |
| 13967 | location 155 |

is given in terms of the number of faults: many faults (2), a few faults (1) and no faults (0). The *'log-feature'* file identifies features or markers like alarms and notifications by their numbers. There can be up to 386 features associated with an anomalous event. The *severity_type* rates the warning conditions in terms of their seriousness (on a scale 1 to 5 with 5 being the most serious).

The training dataset contains *"id,"* the location of the incidence and the severity of the fault. The rest of the fields can be extracted from the other sub-datasets to make a complete dataset for training detection models. In the case of localization, the available sub-datasets as collated with the training dataset such that the localization model gives a good prediction of severity of faults. An extract from the training and test datasets are given in Table 9A and 9B respectively. The test dataset has "*id*" and the *location* for which severity has to be predicted.

The Telstra *log_feature* sub-dataset contains 58,672 examples, with events displaying the presence of different features. The *event_type* sub-dataset has 31,170 examples, the *resource_type* sub-dataset has 21,076 and *severity_type* sub-dataset has 18,552 examples. The test and the training files have 11,171 and 7381 records respectively. They have not been split from a common dataset so the standard 80:20 or a similar ratio is not maintained. A dataset prepared by the consolidation of all sub-datasets has more than 800 features as shown in Table 10. Each fault (with a unique id) is associated with a location, up to 6 features and corresponding volumes, up to three affected resources, up to 5 events, and up to 5 severity types indicating the intensity of the warning and *fault_severity* ranging from 0-2 as explained before.

**TABLE 10**
LIST OF FEATURES FROM NETWORK FAULT DATASET

| No of Features | Feature Name | Explanation |
|---|---|---|
| 1 | id | Unique id for an anomaly situation. It contains a time-stamp. |
| 2 | location | Location of the event |
| 3-12 | resource_type | Up to 10 resources may be involved |
| 13-398 | feature | There are 386 types of markers of which usually a few will be present |
| 399-797 | volume | There is volume information for each feature present |
| 798-802 | event_type | Up to 5 event_types may be associated with an anomalous situation |
| 803 | severity_type | Indicates severity of warning for the situation. The scale is 1-5 with 5 being the most severe |
| 804 | fault_severity | 0 indicates no fault, 1 indicates a few faults and 2 indicates many faults |

A part of the consolidated Telstra dataset is shown in Table 11. Only *feature1* (out of the complete set of features from *feature1* to *feature386*) is shown for compactness. As part of preprocessing of the dataset, selection of features was carried out based on the degree of correlation of each feature with the labels using the Weka tool [78]. With the dataset used in this study, a correlation threshold of 23% was found to improve accuracy.

**TABLE 11**
CONSOLIDATED TRAINING DATASET

| id | location | fault_ severity | resource1 | resource 2 | event1 | event2 | event3 | event4 | severity_type | feature1 | volume1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | location 243 | 0 | resource_type 2 | | event_type 34 | event_type 35 | | | severity_type 2 | 232 | 3 |
| 13 | location 418 | 0 | resource_type 2 | | event_type 35 | event_type 34 | | | severity_type 2 | 232 | 1 |
| 19 | location 644 | 1 | resource_type 2 | | event_type 42 | event_type 44 | | | severity_type 1 | 368 | 2 |
| 20 | location 79 | 0 | resource_type 2 | | event_type 54 | event_type 11 | | | severity_type 2 | 55 | 1 |
| 23 | location 257 | 0 | resource_type 8 | resource_ _type 2 | event_type 35 | event_type 34 | event_type 10 | | severity_type 2 | 307 | 1 |
| 24 | location 367 | 0 | resource_type 2 | | event_type 35 | | | | severity_type 4 | 312 | 2 |
| 26 | location 238 | 0 | resource_type 2 | | event_type 35 | | | | severity_type 4 | 312 | 1 |
| 27 | location 793 | 0 | resource_type 8 | | event_type 11 | | | | severity_type 1 | 73 | 3 |
| 28 | location 889 | 0 | resource_type 8 | | event_type 11 | | | | severity_type 2 | 68 | 2 |

## 7.2 Evaluation of the detection subsystem

To prepare the data for Level-1 detection, the *fault_severity* has been curated to have binary values with 0 indicating 'no-fault' and 1 indicating 'fault.' The detection classification of 'fault'/'no-fault' was implemented with a number of supervised learning techniques of which SVM, ADT, and RF have been shown in Table 12. On the basis of accuracy, SVM and ADT perform comparatively better than RF. In each case, 10% cross-validation was used.

**TABLE 12**
STAGE-1 DETECTION RESULTS

| Benchmark Algorithm | SVM | ADT | Random Forest |
|---|---|---|---|
| Time taken | 0.01 seconds | < 0.01 seconds | 0.1 seconds |
| Correctly classified instances | 95.42% | 95.00% | 86.67% |
| Precision (Average) | 95.7% | 95.2% | 86.9% |
| Mean absolute error | 0.0458 | 0.0859 | 0.2509 |
| Root mean squared error | 0.2141 | 0.2092 | 0.3261 |
| True positive for class 0 | 94.3% | 94.3% | 95.5% |
| False positive for class 0 | 2.4% | 3.6% | 30.1% |
| True positive for class 1 | 97.6% | 96.4% | 69.9% |
| False positive for class 1 | 5.7% | 5.7% | 4.5% |

With this dataset, SVM, on the whole, performs better than ADT and RF giving $\geq$ 95.4% accuracy. Considering the definitions in Table 13, the true positive (TP) rate for 'fault' cases were the highest for SVM showing that these were correctly classified as 'fault' cases. Considering the nature of the dataset, this result indicates a good result. There were no faults and system said fault in 5.7% cases, while there were faults and system said no faults in 2.4% cases. The false positive and negative rates were the lowest in SVM and the highest in Random Forest. A desirable outcome is that besides classifying faults and faults and no faults as no faults with high accuracy, it classifies a very low percentage of faults as no-faults, thus, helping to do what is intended to do – detect performance and fault issues. SVM and RF also gave high precision indicating that 'no-fault' cases were correctly classified by them.

**TABLE 13**
METRIC USED

| Metric | Interpretation |
|---|---|
| Accuracy | (TP+TN)/(TP+TN+FP+FN) |
| Precision | TP/(TP+FP) |
| Recall | TP/(TP+FN) |
| TP=True Positive, TN=True Negative, FP=False Positive, FN=False Negative | |

To get a sense of the performance of our detection model, using SVM with RBF Kernel, we compared the results with baseline results obtained by Zero-R model. The Zero-R model predicts the majority class. Running on our datasets, the baseline result was about 63%, which indicates that our chosen model gives a substantial improvement over the naïve baseline.

At Level-2, the detection module classifies the fault cases as 'manifested' or 'impending.' For the Level-2 classification into manifested/impending classes again a tuned SVM with RBF Kernel works well as can be seen from Table 14.

**TABLE 14**
LEVEL-2 DELTECTION MODEL PERFORMANCE

| Metric | Value |
|---|---|
| Correctly detected manifested faults | 89.7% |
| Correctly detected impending faults | 95.1% |
| Impending fault classified as a manifested fault | 4.9% |
| Manifested fault classified as an impending fault | 10.3% |

For Level-2 detection, we have chosen One-R as the baseline algorithm. One-R is a simple but accurate classification algorithm, which generates one rule for each predictor and then selects the one with the smallest error. The accuracy of our framework is 13.03% better for 'impending' faults and 5.97% better for 'manifested' faults, which is a significant improvement (Fig. 13.)

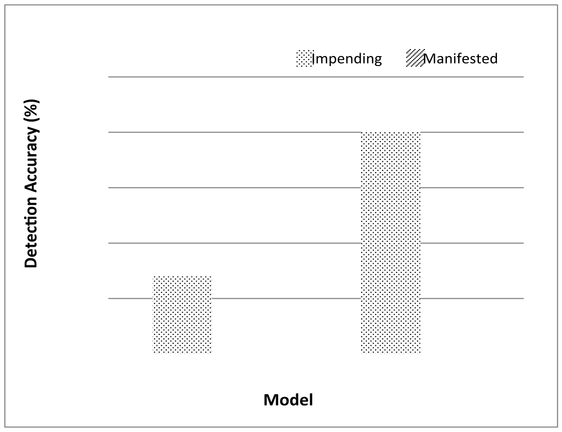


**Fig. 13.** Detection Level 2 effectiveness compared to baseline

## 7.3 Evaluation of Localization Subsystem

As discussed in Subsection 6.5, for handling manifested FP issues, the localization subsystem was implemented in two layers with multi-level classification carried out at both the levels. For the multi-class classification with SVM, we chose to work with One vs. One (OvO) [47]. In the OvO approach, for the ith classifier $f_i$, the examples can be classified with $f(x) = \arg\max_i f_i(x)$, i.e., choose the class that classifies the example with the maximum margin.

**TABLE 15**
LOCALIZATION LAYER 1 BASELINE PERFORMANCE

| === Stratified cross-validation === === Summary === | | | | | |
|---|---|---|---|---|---|
| Correctly Classified Instances | 86.14% | | | | |
| Root Mean Squared Error | 0.26 | | | | |
| === Detailed Accuracy By Class === | | | | | |
| Weighted Average of all classes | TP Rate | FP Rate | Precision | Recall | PRC Area |
| | 0.861 | 0.048 | 0.947 | 0.861 | 0.806 |

At Layer-1, the model classifies the faults into one of several broad categories of FP issues as was shown in Table 8. We set up the baseline performance with OneR as shown in Table 15.

At Layer-1, we chose the sequential minimal optimization (SMO) multi-class support vector classifier. With SMO and RBF Kernel and parameters C = 12, gamma = 0.01 epsilon = $1\times10^{-12}$, the accuracy of Layer-1 localization is 97%, which is a substantial improvement over the baseline performance of 86.14%. The performance of the model is given in Table 16.

**TABLE 16**
LOCALIZATION LAYER 1 MODEL PERFORMANCE

| === Stratified cross-validation === === Summary === | | | | | |
|---|---|---|---|---|---|
| Correctly Classified Instances | 97.03% | | | | |
| Root Mean Squared Error | 0.32 | | | | |
| === Detailed Accuracy By Class === | | | | | |
| Weighted Average of all classes | TP Rate | FP Rate | Precision | Recall | PRC Area |
| | 0.970 | 0.029 | 0.971 | 0.970 | 0.967 |

Fig. 14 gives a comparison of the performance of our Multi-Class Multi-Layer (MCML) model with the baseline. It can be seen that the accuracy of the classification of MCML is 97.03% against the baseline accuracy of 86.14%. A useful metric for comparison of classifiers is Precision-Recall Area (PRC Area), which gives the tradeoff between precision and recall. A high value indicates high precision (i.e., low false positives) and high recall (i.e., low false negatives). We can see that MCML Level 1 gives a high PRC Area of 0.967 compared to 0.806 of the baseline.

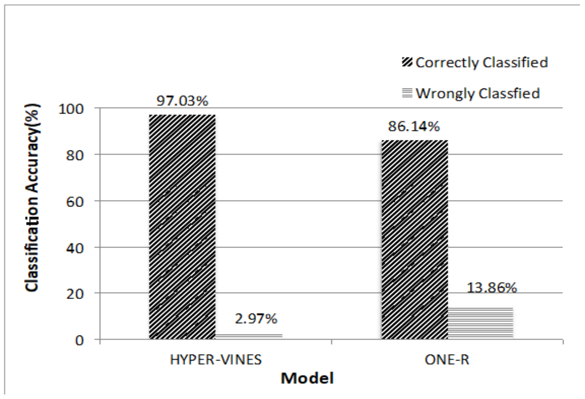


**Fig. 14.** Localization Layer 1 effectiveness compared to baseline

Once a broad category has been identified, the Layer 2 model does fine grain localization for each category of manifested fault. In a dataset containing Network Performance Faults at Layer 1 and 5 different faults at layer 2, we have the results in Table 17:

**TABLE 17**
LOCALIZATION LAYER 2 MODEL PERFORMANCE

| === Stratified cross-validation === === Summary === | | | | | |
|---|---|---|---|---|---|
| Correctly Classified Instances | 96.04% | | | | |
| Root Mean Squared Error | 0.294 | | | | |
| === Detailed Accuracy By Class === | | | | | |
| Weighted Average of all sub-classes | TP Rate | FP Rate | Precision | Recall | PRC Area |
| | 0.960 | 0.002 | 0.976 | 0.96 | 0.955 |

A seen from Table 18, when compared with the baseline algorithm result, we see that multi-class classification with SMO and OvO has a much superior performance, indicating the efficacy of the model. The localization accuracy of the model is 96.04% compared to 90.1% of baseline. The PRC Area of the MCML Level 2 classification is 0.955 against 0.846 of the baseline.

**TABLE 18**
LOCALIZATION LAYER 2 BASELINE PERFORMANCE

| === Stratified cross-validation === === Summary === | | | | | |
|---|---|---|---|---|---|
| Correctly Classified Instances | 90.099% | | | | |
| Root Mean Squared Error | 0.1573 | | | | |
| === Detailed Accuracy By Class === | | | | | |
| Weighted Average of all sub-classes | TP Rate | FP Rate | Precision | Recall | PRC Area |
| | 0.901 | 0.008 | 0.935 | 0.901 | 0.846 |

Fig. 15 gives the graphical comparison of Level 2 performance of the implemented model (MCML: Multi-class, Multi-layer, in our case SMO) and the baseline. It is seen that the implemented model gives a higher percentage of correctly classified and lower percentage of wrongly classified

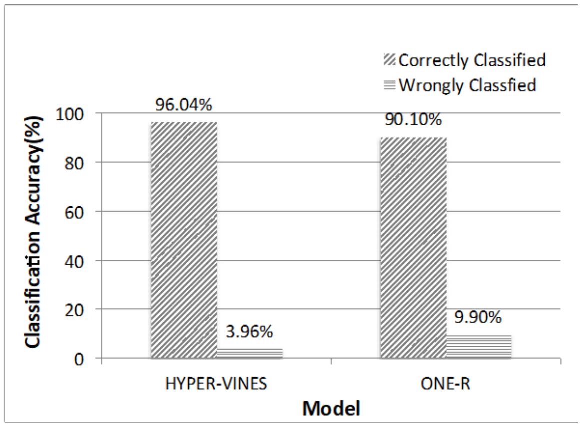


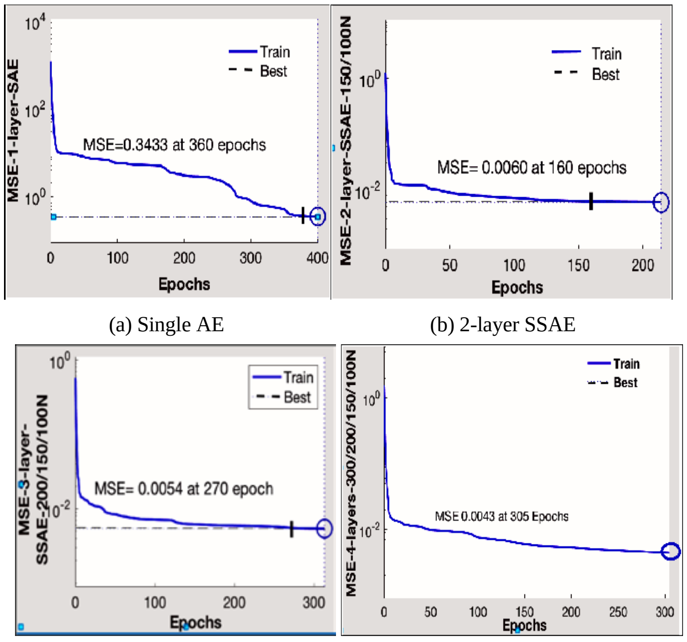
**Fig. 15.** Localization Layer 2 effectiveness compared to baseline

examples.

## 7.4 Localization of Impending FP Issues

One of the main concerns handled in the framework is to localize impending faults and predict their severity levels. We have seen in Section 6.5 that the localization sub-system uses stacked sparse autoencoder (SSAE) for faults detected as impending faults. While at the preprocessing stage a total of 353 features were selected, further condensation was left to the SSAE used.

To make the deep learning model predict with high accuracy, the first step is to train the stacked autoencoders such that the output is as close a replica of the input as possible. To achieve optimum performance, the stacked autoencoder parameters like the number of hidden layers, the size of the layers, sparsity regulation (SR) and sparsity proportion need to be judiciously arrived at. An example of comparative reconstruction performance is given in Figures 16(a) through 16(d). It is seen that the model with 3 hidden layers of 200/150/100 neurons, respectively, converges quite fast to a low mean-square error. Reconstruction accuracy is important as it affects the prediction based on the trained encoders, which the model is eventually used for [22].



(a) Single AE (b) 2-layer SSAE

(c) 3-layer SSAE (d) 4-layer SSAE

**Fig. 16.** Mean square error for reconstruction of the input

Sparsity in data is handled by using the Autoencoder parameters sparsity regulation (SR) and sparsity proportion (SP). SP gives the proportion of training examples a neuron reacts to. A low value of SP encourages sparsity.

Having achieved good reconstruction results with stacked autoencoders, the model was tested for prediction of the severity of impending fault and performance issues. As discussed in Section 6.5, a Softmax layer is added as a prediction layer. The graph in Fig. 17 shows that the model has good generalization characteristics as MSE for the test dataset is close to that of the training dataset.

Fine tuning of the model was done using backpropagation. The accuracy ranges between 72 and 85% with the abridged dataset (~1000 examples) and ~92% with the enhanced dataset (~5000 examples). Experiments were carried out for SR = 1 and SP = 0.4.

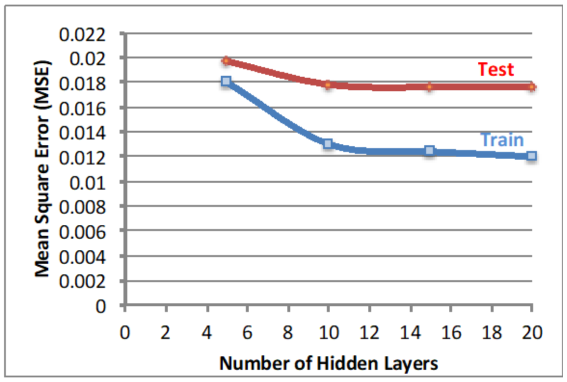


**Fig. 17.** MSE in training and test dataset

We baselined the above results with those obtained with a shallow model, viz., SVM with RBF kernel which worked very well for detection and could only obtain 73.1% accuracy in localizing impending faults. A comparison between SSAE and SVM models is shown in Fig. 18. The deep structure thus provided a substantial improvement in terms of accuracy of prediction of the severity level of the impending faults.

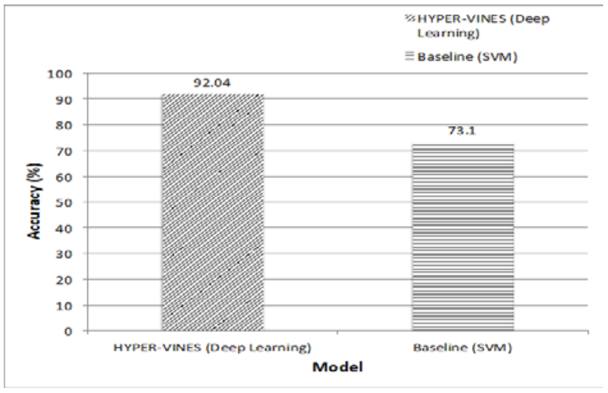


**Fig. 18.** Localization of impending fault stacked autoencoder and SVM (baseline)

## 8. Summary

This tutorial introduces the issue of availability and performance management of carrier services using NFV over a multi-cloud in a way that achieves the goals established in Subsection 1.2. Briefly recapitulating the following goals were set: a*) Discuss the architecture, creation and management of VNS, b) Elucidate FP problem, c) Usefulness of AI techniques in the cloud-based NFV environments, d) Describe the AI based FP management framework and e) Discuss the use of hybrid machine and deep learning techniques with a real-life case study*. To enable a holistic understanding of the fault and performance issues, the tutorial describes the design of VNSs like carrier mobile or broadband services that are designed using SFCs. All the management platforms (MANO, OSS/BSS, and MMCP) that play important roles in fault and performance management of VNSs and their interactions have been discussed. MANO is the main component of NFV life cycle and fault management. Our tutorial appropriately discusses its constitution and functions in detail. Responsibilities of each of the sub-systems of the MANO towards monitoring and management of fault and performance issues have been described. Interfaces that have been defined between the MANO and the multi-cloud manager (MMCP) and between the MANO and the OSS have been discussed. All these aspects cover goal a). Towards achieving goal b), a full section has been devoted to the description of the fault and performance issues wherein we also discuss the criticality of faults and the shared FP responsibilities of the management platforms. To meet goal c), explanation has been given for the importance of considering AI for achieving the goals of the FP problem. Towards achieving goal d), a generic framework for detection and localization of the FP issues has been proposed and described in detail. It has been brought out how the AI based framework would be able to go beyond the traditional models in predicting impending failures and their severity. Markers and metrics are important ingredients of any FP management system and have been given a fitting treatment. To accomplish goal e), we have discussed the results of a case study involving the implementation and evaluation of the detection and localization functionalities using the machine and deep learning respectively. Using an actual network fault data, we have shown how manifested and impending FP issues can be effectively handled by the detection and localization sub-systems of the FP management framework based on machine and deep learning models.

## Acknowledgment

This publication was made possible by NPRP grant #8-634-1-131 from the Qatar National Research Fund (a member of Qatar Foundation), NSF grants CNS-1718929 and CNS-1547380. The statements made herein are solely the responsibility of the authors. This paper draws from earlier works of the authors including 'HYPER-VINES: A Hybrid Learning Fault and Performance Issues Eradicator for Virtual Network Services over Multi-Cloud Systems**'** presented at the IEEE ICNC 2019 Conference in February 2019, and from the other references listed in the reference section.

## References

[1] D. Young, M. Toussaint "Hype Cycle for Enterprise Networking and Communications," Gartner Report #G00338722, 13 July 2018, 69 pp.

[2] ETSI Whitepaper, "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action," SDN and OpenFlow World Congress, 2012, 16 pp.

[3] ETSI Report, "ETSI Plugtests demonstrate high interoperability levels and increased feature support," http://www.etsi.org/index.php/news-events/news/1276-2018-02-news-2nd-etsi-nfv-plugtests-demonstrate-high-interoperability-levels-and-increased-feature-support, February 2018.

[4] ETSI GR NFV-IFA 015 V2.1.1, Group Report, "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Report on NFV Information Model," 2017.

[5] ITU-T Recommendation M.3400 Series M: "TMN AND Network Maintenance: International Transmission Systems, Telephone Circuits, Telegraphy, Facsimile and Leased Circuits TMN Management Functions," 2002.

[6] ITU Recommendation X.733, "Information Technology-Open System Interconnection- Systems Management-Alarm Reporting Function," 1992.

[7] ISO 9595, "Information Processing Systems - Open Systems Interconnection, Management Information Service Definition – Part 2: Common Management Information Service," 22 December 1988.

[8] ISO 9596, "Information Processing Systems - Open Systems Interconnection, Management Information Protocol Specification - Part 2: Common Management Information Protocol," 22 December 1988.

[9] Byung Yun Lee, Bhum Cheol Lee, "Fault Localization in NFV Framework," ICACT, 2016, pp. 352-355.

[10] C. J. Bernardas, A Rahman, JC Zunjia, L. M. Contreras, P. Aranda, P. Lynch "Network Virtualization Research Challenges," IETF internet draft, 2018.

[11] R. Glitho, "Cloudifying the 3GPP IP Multimedia Subsystem: Why and How?" 6th International Conference on New Technologies, Mobility and Security (NTMS), 2014, pp. 1-5.

[12] D. Lopez, "Network Functions Virtualization: Beyond Carrier-Grade Clouds," Optical Fiber Communications Conference and Exhibition (OFC)," 2014, pp. 1-18.

[13] ITU-T SG13 Q19 "Potential New Work For Q19/13" https://www.itu.int/md/T17-SG13-171106-TD-WP2-0139/en, 2017, accessed September 2018.

[14] R. Jain, "Fault and Performance Management in Carrier-grade Virtual Networks Over Multiple Clouds," NSF Proposal, NETS, 2017.

[15] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, H. A. Chan, "Fault and Performance Management in Multi-cloud based NFV using Shallow and Deep Predictive Structures," J. Reliable Intell Environ, 2017, pp. 1-8.

[16] Multi-domain Network Virtualization draft-bernardos-nvvrg-multidomain-04 C.J. Bernardos, Ed., et. al. Informational Internet-Draft , expires March 2019. https://tools.ietf.org/html/draft-bernardos-nfvrg-multidomain-05 Accessed 4 December 2018.

[17] ETSI GS NFV 002, General Specification, "Network Functional Virtualization; Architectural Framework," 2013

[18] J. Halpern, C. Pignataro, "Service Function Chaining (SFC) Architecture," IETF RFC 7665, 2015

[19] C. J. Bernardos, A. Rahman, J. C. Zuniga, L. M. Contreras, P. Aranda, P. Lynch, "Network Virtualization Research Challenges," IRTF draft draft-irtf-nfvrg-gaps-network-virtualization-10, September 2018, 40 pp.

[20] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, C. Metz, "COLAP: A Predictive Framework for Service Function Chain Placement in a Multi-cloud Environment," The 7th IEEE Annual Computing and Communication Workshop and Conference (CCWC), 2017, pp. 1-9.

[21] Y. Chen, A. Bernstein, "Bridging the Gap Between ETSI-NFV and Cloud-Native Architecture," SCTE/ISBE, Fall Technical Forum, 2017

[22] R Mijumbi, J Serrat, J-L Gorricho, S. Latre, M. Charalambides and D Lopez, "Management and Orchestration Challenges in Network Function Virtualization," IEEE Communications Magazine, 2016, pp. 98-105

[23] ETSI GS NFV 002 V1.2.1, General Specification, "Network Functions Virtualisation (NFV); Architectural Framework," 2014.

[24] ETSI GS NFVMAN001 V1.1.1, General Specification, "Network Functions Virtualization (NFV); Management and Orchestration," 2014.

[25] F. Khan, "A Beginner's Guide to NFV Management & Orchestration (MANO), http://www.telcocloudbridge.com/a-beginners-guide-to-nfv-management-orchestration-mano, 2015.

[26] "A note on descriptor files - ETSI NFV "Management and Orchestration - An Overview," Mehmet Ersue, ETSI NFV MANO WG Co-chair IETF #88, 2013

[27] ONF-XOS, https://www.opennetworking.org/xos/. Accessed November 2018

[28] Dmitriy Andrushko, Gregory Elkinbard, “What is the best NFV Orchestration platform? A review of OSM, Open-O, CORD, and Cloudify,” https://cloudify.co/2017/03/15/what-best-nfv-orchestration-platform-review-osm-openo-cord-cloudify.html, 2017, accessed October 2018

[29] P. Reynolds, C. Killian, J. L. Wiener, “Pip: Detecting the Unexpected in Distributed Systems,” 3rd Symposium on Networked Systems Design & Implementation, 2006, pp. 115-128.

[30] D. Gruer, I. Khan, R. Ogier, R. Keffer, “An Artificial Intelligence Approach to Network Fault Management,” SRI International, 2015

[31] P. Reynolds, C. Killian, J. L. Wiener, “Pip: Detecting the Unexpected in Distributed Systems,” 3rd Symposium on Networked Systems Design & Implementation, 2006, pp. 115-128.

[32] R. R. Kompella, J. Yates, A. G. Greenberg, A. C. Snoeren, “Fault Localization via Risk Modeling,” IEEE Trans. Dependable Sec. Computing, 2010, pp. 396-409.

[33] M Boucadair, C. Jaquenet, “Handbook of research on redesigning the future of Internet architectures,” IGI Global, 2015, 621 pp

[34] A. Lakhina, M. Crovella, C. Diot, “Diagnosing network-wide traffic anomalies,” SIGCOMM, 2004, 12 pp.

[35] A. Lakhina, M. Crovella, C. Diot, “Diagnosing network-wide traffic anomalies,” SIGCOMM, 2004, 12 pp.

[36] D. Kushnir, M. Goldstein, “Causality Inference for Failures in NFV,” IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS): SWFAN 16: International Workshop on Software-Driven Flexible and Agile Networking, 2016

[37] Cisco Content Hub, "Causality correlation," https://content.cisco.com/chapter.sjs?uri=%2Fsearchable%2Fchapter%2Fwww.cisco.com%2Fcontent%2Fen%2Fus%2Ftd%2Fdocs%2Fnet_mgmt%2Factive_network_abstraction%2F3-7-2%2Ftheory%2Foperations%2FTheoryofOperations%2Fcaus-theory.html.xml&platform=Cisco%20Active%20Network%20Abstraction, 2018. Accessed 20th September 2018.

[38] Y. Endo and M. Seltzer, “Improving interactive performance using TIPME,” Proc. ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Volume 28, Issue 1, 2000, pp. 240-251.

[39] M. Chen, E. Kiciman, E. Fratkin, E. Brewer, and A. Fox, “Pinpoint: problem determination in large, dynamic, internet services,” Proc. International Conference on Dependable Systems and Networks (IPDS Track), 2002, pp. 595-604.

[40] P. Barham, R. Isaacs, R. Mortier, and D. Narayanan, “ Magpie: online modeling and performance-aware systems,” Proc of the 9th conference on Hot Topics in Operating Systems, 2003, pp. 15-15.

[41] V. Dhar, “Data Science and Prediction,” Communications of the ACM, 2013, pp. 64-73.

[42] L. Lewis, “A case-based reasoning approach to the management of faults in communication networks,” Infocom, 1993, 1422-1429.

[43] S. Jiang, D. Siboni, A. A. Rhissa, G. Beuchot, “An intelligent and integrated system of network fault management: artificial intelligence technologies and hybrid architectures,” IEEE Networks, 1995, pp. 265-268.

[44] D. W. Gürer, I. Khan, R. Ogier, R. Keffer, “An Artificial Intelligence Approach to Network Fault Management,” SRI International, Citeseer 1996, pp. 1-10.

[45] R. D. Gardner, D. A. Harle, “Alarm correlation and network fault resolution using the Kohonen self-organizing map,” Globecom 1997, pp. 1398-1402.

[46] J. McClelland, A. Rumelhart, “Distributed model of human learning and memory,” *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. II). Cambridge, MA: MIT Press, 1986, 550 pp.

[47] G.E. Hinton, “Connectionist learning procedures. Artificial Intelligence,” 1989, pp. 185-234.

[48] P. E. Utgoff , D. J. Stracuzzi, “Many-layered learning,” Neural Computation, 2002, pp. 2497-2529.

[49] A. Yilmaz, “Comparative study for identification of multiple alarms in telecommunication networks,” Turkish Journal of Electrical Engineering & Computer Sciences, 2016, pp. 677-688.

[50] E. Rozaki, “Network Fault Diagnosis Using Data Mining Classifiers,” Eleni Rozaki International Journal of Data Mining & Knowledge Management Process, 2015, pp. 29-40.

[51] M. Jaudet, N. Iqbal, A. Hussain, and K. Sharif, (2005) “Temporal classification for fault-prediction in a real-world telecommunications network,” International Conference on Emerging Technologies, 2005, pp. 209-214.

[52] K. Qadar, M. Adda, “Network Faults Classification Using FCM” International Journal of Advanced Research in Computer and Communication Engineering, 2013.

[53] C. Sauvanaud, K. Lazri, M. Kaniche, K. Kanoun, “Anomaly Detection and Root Cause Localization in Virtual Network Functions,” IEEE 27th International Symposium on Software Reliability Engineering, 2016.

[54] M. Miyazawa and M. Hayashi, R. Stadler, “vNMF: Distributed Fault Detection using Clustering Approach for Network Function Virtualization,” IFIP/IEEE International Symposium on Integrated Network Management (IM2015), 2015.

[55] M. Hayashi, “Machine Learning-assisted Management of a Virtualized Network,” Optical Fiber Communication Conference, Optical Society of America, 2018.

[56] ETSI GS NFV-REL 001 V1.1.1, General Specification, “Network Functions Virtualisation (NFV); Resiliency Requirements,” 2015.

[57] B.P. Majumder, A. Sengupta, S. Jain, P. Bhaduri., “Fault Detection Engine in Intelligent Predictive Analytics Platform for DCIM,” (publication unknown), https://arxiv.org/pdf/1610.04872, 2016, 15 pp.

[58] L. Gupta, T. Salman, R. Das, A. Erbad, R. Jain, M. Samaka, “HYPER-VINES: A Hybrid Learning Fault and Performance Issues Eradicator for Virtual Network Services over Multi-Cloud Systems,**”** IEEE ICNC 2019

[59] ITU-T E.800 [i.10]: Recommendations, “Definitions of terms related to quality of service,” 2008.

[60] ETSI GS NFV-INF 010 V1.1.1, General Specification, “Network Functions Virtualisation (NFV); Service Quality Metrics,” 2014.

[61] ETSI GS NFV-INF 010 V1.1.1, General Specification, “Network Functions Virtualisation (NFV): Service Quality Metrics,” 2014.

[62] TRAI Report, “Performance Indicator Reports,” http://www.trai.gov.in/release-publication/reports/performance-indicators-reports, 2018

[63] H. N. Mhaskar and T. Poggio, “Deep vs. Shallow Networks: an Approximation Theory Perspective,” Center for Brains, Minds, and Machines (CBMM), CBMM Memo No. 054, 2016

[64] Vikas Sindhwani, “Shallow vs. deep: the great watershed in learning,” Princeton University Lectures, 2017

[65] G. Ososkova, and P. Goncharov, “Shallow and Deep Learning for Image Classification,” Optical Memory and Neural Networks 26(4):221-248, October 2017

[66] M. J. Kearns, “The computational complexity of machine learning,” MIT Press, 1990, 176 pp.

[67] Y. Bengio, “Learning Deep Architectures,” Bengio Foundations and Trends in Machine Learning, 2009, 127 pp.

[68] H. N. Mhaskar, T. Poggio, “Deep vs. Shallow Networks: an Approximation Theory Perspective,” arXiv:1608.03287v1 [cs.LG], 2016, 16 pp.

[69] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview,” Neural Networks, Elsevier, 2014, pp. 85-117.

[70] A. J. Smola and B. Scholkopf, “A tutorial on support vector regression,” Statistics and Computing, 2004, pp. 199-222.

[71] T. Hastie, R. Tibshirani, J. Friedman, “The Elements of Statistical Learning,” Springer Science & Business Media, 2009, 745 pp.

[72] M. A. Jabber, B. L. Deekshatulu, P. Chandra, “Alternating decision trees for early diagnosis of heart disease,” Proceedings of International Conference on Circuits, Communication, Control and Computing, 2014, pp. 322-328

[73] G. Louppe, “Understanding Random Forests – From Theory to Practice,” Ph.D. Dissertation, University of Liege, France, 2014.

[74] “Stacked autoencoder (Unsupervised Feature Learning and Deep Learning),” http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders, Accessed December 29, 2018.

[75] D. Bhamare, T. Salman, M. Samaka, A. Erbad, R. Jain, "Feasibility of Supervised Machine Learning for Cloud Security," 3rd International Conference on Information Science and Security (ICISS2016), 2016, pp. 1-5.

[76] B-E Laure , B. Angela, M. Tova, " Machine Learning to Data Management: A Round Trip," IEEE 34th International Conference on Data Engineering (ICDE), 2018, pp. 1735-1738.

[77] Kaggle datasets, available at https://www.kaggle.com/c/telstra-recruiting-network/data. Accessed October 2018.

[78] E. Frank, M. A. Hall, I. H. Witten, “The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques,” Morgan Kaufmann, Fourth Edition, 2016.

[79] G. Albuquerque, T. Lowe, and M. Magnor, “Synthetic Generation of High-Dimensional Datasets,” IEEE Transactions on Visualization and Computer Graphics,” 2011, pp. 2317-2324.

[80] E. Torlak, “Scalable Test Data Generation from Multidimensional Models,” SIGSOFT/FSE’12, 2012, pp. 1-11.

[81] J.S. Simonoff, “Smoothing Methods in Statistics,” Springer, 1996, 340 pp.

[82] Z. Botev, “Fast multivariate kernel density estimation for high dimensions,” Mathworks, 2016.

[83] Cloud-Native Network Functions (CNFs) White Paper, Cisco, updated June 2018, https://www.cisco.com/c/en/us/products/collateral/routers/cloud-native-broadband-router/white-paper-c11-740841.html. Accessed Nov. 2018.

[84] J Guichard et al., "NSH and Segment Routing Integration for Service Function Chaining," IETF draft-guichard-SFC-nsh-sr-00, June 2018.

[85] N. F. S. de Sousaa, D. A. L. Pereza, R. V. Rosaa, M. A. S. Santosb, C. E. Rothenberg, “Network Service Orchestration: A Survey,” arXiv:1803.06596v4 [cs.NI], May 2019.

[86] Y. Yu et al, “Fault Management in Software-Defined Networking: A Survey,” IEEE Communications and Survey1s Sept 2018

[87] L. Velasco, D. Rafique, “Fault Management Based on Machine Learning [Invited],” OFC 2019.

[88] H. Yang, Y. He, J. Zhang, Ji Y, W. Bai, Y. Lee, “Performance evaluation of multi-stratum resources optimization with network functions virtualization for cloud-based radio over optical fiber networks,” Optical Express, 2016, Pp. 8666-78.

[89] H. Yang, J. Zhang, Y. Ji, R. Tian, J. Han, Y Lee, “Performance evaluation of multi-stratum resources integration based on network function virtualization in software defined elastic data center optical interconnect,” Optics Express, 2015, Pp. 31192-31205.

**Lav Gupta** is a senior member of IEEE. He received BS degree from Indian Institute of Technology, Roorkee, India in 1978 and MS degree from Indian Institute of Technology, Kanpur, India in 1980 and a PhD in Computer Science & Engineering from Washington University in St Louis, Missouri, USA.

He has worked for about 15 years in the area of telecommunications planning, deployment and regulation. With the sector regulatory authority he worked on technology and regulation of next generation networks. He has also worked as senior teaching faculty of Computer Science and Access Network Planning for a number of years in telecommunications academies. He is the author of one book, 10 papers and has been a speaker at many international seminars. His current research interests include application of machine and deep learning to management of NFV deployments in multi-cloud.

He was recipient of best software award from Computer Society of India in 1982 and best faculty award at Etisalat Academy, UAE in 1998.

**Tara Salman** is a student member of IEEE. She received her BS and MS from Qatar University Doha, Qatar at 2012 and 2015, respectively. Her BS was in computer engineering while her MS was in computing-networking field. She is currently pursuing a PhD at Computer Science & Engineering at Washington University in St Louis, Missouri, USA.

From 2012 -2015, She has worked as a research assistant with Qatar University on a NPRP (NATIONAL PRIORITIES RESEARCH PROGRAM) funded project targeting physical layer security. From 2015, she is working as a Graduate Research assistant at Washington University in St. Louis. Her research interest spans network security, distributed systems, Internet of things and financial technology. She is an author of 1 book chapter, 6 research articles and has been a presenter at many international conferences.

Salman is a recipient of Cisco Certified Network Associate (CCNA) certification in 2012 and the priory completed all four level of CCNA at Cisco academy-Qatar university branch.

**Maede Zolanvari** is an IEEE student member. She received both her B.S. and M.S. degree in Electrical and Computer Engineering, in 2012 and 2015 respectively. She’s currently a Ph.D. candidate in Computer Science and Engineering at Washington University, St. Louis, MO, USA. During 2012 through 2015, her research was on performance improvement of communication networks, with a focus on OFDM systems. Since 2015, she has been working as a graduate research assistant at Washington University, St. Louis. Her current research focus is on utilizing machine learning and deep learning for network security of the Industrial Internet of Things. Her research interests include Internet of Things, machine learning, cyber-security, secure computer networks and wireless communications.

**Aiman Erbad** is an Assistant Professor at the Computer Science and Engineering (CSE) Department at Qatar University. Dr. Erbad obtained a PhD in Computer Science from the University of British Columbia (Canada) in 2012, a Master of Computer Science in Embedded Systems and Robotics from the University of Essex (UK), and a Bachelor of Science in Computer Engineering from the University of Washington (USA). Since September 2016, Dr. Erbad was the Director of Research Support, responsible for all research grants and contracts. Prior to that Dr. Erbad was the Coordinator of the Computer Engineering program and the Chair of the Curriculum and Quality Assurance committee leading ABET accreditation and curriculum enhancement efforts at the CSE department.

Dr. Erbad received the Platinum award from H.H. The Emir Sheikh Tamim bin Hamad Al Thani at the Education Excellence Day 2013 (PhD category) and graduated from Qatar Leadership Center, which trains rising leaders in different sectors. Dr. Erbad research interests span cloud computing, multimedia systems and networking, and security. Dr. Erbad research received funding from Qatar National Research Fund and his research is published in reputed international conferences and journals. Dr. Erbad is a member of various University committees (Policy, Ranking, Institutional Effective, Intellectual Property, Appeal and Re-instatement) and the Chair of the University Research Support Committee. He serves as an Editor in the European Alliance for Innovation (EAI) Endorsed Transactions on Collaborative Computing, and as a technical program committee member in various IEEE and ACM international conferences. Dr. Erbad acts as an expert in information technology strategy and research techniques for various national entities.

**Raj Jain** is a Fellow of IEEE, a Fellow of ACM, a Fellow of AAAS. He received BS degree in Electrical Engineering from APS University in Rewa, India in 1972 and MS in Computer Science & Controls from IISc, Bangalore, India in 1974 and the Ph.D. degree in Applied Math/Computer Science from Harvard University in 1978.

Dr. Jain is currently a Professor of Computer Science & Engineering at Washington University in St. Louis. Previously, he was one of the Co-founders of Nayna Networks, Inc - a next generation telecommunications systems company in San Jose, CA. He was a Senior Consulting Engineer at Digital Equipment Corporation in Littleton, Mass and then a professor of Computer and Information Sciences at Ohio State University in Columbus, Ohio. He has 14 patents and has written or edited 12 books, 16 book chapters, 65+ journal and magazine papers, and 10e5+ conference papers.

He is a winner of ACM SIGCOMM Test of Time award, CDAC-ACCS Foundation Award 2009, and ranks among the top 100 in CiteseerX's list of Most Cited Authors in Computer Science.