Fault and performance management in multi-cloud based NFV using shallow and deep predictive structures

Lav Gupta1 · M Samaka2 · Raj Jain1 · Aiman Erbad2 · Deval Bhamare2 · H Anthony Chan3

Received: 2 October 2017 / Accepted: 21 October 2017 / Published online: 14 November 2017
© Springer International Publishing AG, part of Springer Nature 2017

Abstract Deployment of network function virtualization (NFV) over multiple clouds accentuates its advantages such as flexibility of virtualization, proximity to customers and lower total cost of operation. However, NFV over multiple clouds has not yet attained the level of performance to be a viable replacement for traditional networks. One of the reasons is the absence of a standard based fault, configuration, accounting, performance and security (FCAPS) framework for the virtual network services. In NFV, faults and performance issues can have complex geneses within virtual resources as well as virtual networks and cannot be effectively handled by traditional rule-based systems. To tackle the above problem, we propose a fault detection and localization model based on a combination of shallow and deep learning structures. Relatively simpler detection has been effectively shown to be handled by shallow machine learning structures such as support vector machine (SVM). Deeper structure, i.e., the stacked autoencoder has been found to be useful for a more complex localization function where a large amount of information needs to be worked through to get to the root cause of the problem. We provide evaluation results using a dataset adapted from fault datasets available on Kaggle and another based on multivariate kernel density estimation and Markov sampling.

Keywords NFV · Network function virtualization · Virtual network service · Virtual network function · Service function chain · Multi-cloud · FCAPS · Fault detection · Fault localization · Machine learning · Deep learning · Support vector machine · Stacked autoencoder

1 Introduction

Virtualization of network services using NFV can transform a network service provider’s business model in many ways. By consolidating appliances and middleboxes onto commercially available high-volume servers, they reduce the time to market new services, provide flexibility of scaling and lower the cost of operation [1]. Cloud technology can multiply the benefits of NFV [2,3]. It could provide greater flexibility in obtaining resources, bring network service provider’s (NSP’s) points of presence close to customers, provide an opportunity to optimize performance and control cost. Deploying virtual resources on clouds, especially multiple clouds, is becoming a key to successful deployment of large real-time and distributed services. However, in this NFV on cloud scenario, concerns about five nines availability (99.999%) and quality of service parameters like latency and packet loss still remain [5,6].

Traditional networks have rigorous availability and quality control. They have time-tested standards relating to fault, configuration, accounting, performance and security (FCAPS) as embodied in ISO Common Management Information Protocol (CMIP) and ITU TMN M.3010 and M.3400 recommendations [7,8]. As against this, even though the

This work has been supported under the grant ID NPRP 6 - 901 - 2 - 370 for the project entitled “Middleware Architecture for cloud based services using software defined networking (SDN),” which is funded by the Qatar national research fund (QNRF) and by Huawei Technologies. The statements made herein are solely the responsibility of the authors.

© Springer
complexity of NFV implementations is many notches higher, they still require serious work to become carrier grade and this is not going to be trivial [11,12].

In this paper, we propose mechanisms to handle manifest and latent fault and performance issues in NFV over a multi-cloud environment. Our work shows that a combination of shallow and deep machine learning architectures would be useful in handling voluminous operational data of high dimension for detection and localization of fault and performance issues. The rest of the paper is organized as follows: Sect. 2 discusses how network services are organized in the target environment. In Sect. 3, we discuss the extent and complexity of the FCP part of the FCAPS problem and state the aspects we propose to handle in this paper. In Sect. 4, we bring out state of the art through related works. Sect. 5, deals with the solution strategies while Sect. 6 presents the evaluation results. In Sect. 7, we present the summary.

2 Network service structure

Based on the ETSI specifications [9] and IETF RFC [13] a network service can be described as an ordered set of virtual network functions (VNFs) that represent functions such as routers, broadband network gateways or middleboxes such as load balancers and firewalls. These functions are chained into service function chains (SFC) or VNF graphs interconnected by virtual network resources to handle the traffic in a desired way. VNFs are created in software and hosted on virtual machines (VM) in one or more clouds [10]. Creating virtual network service on multiple clouds gives a number of compelling advantages such as lower capital and operational cost, flexibility of resource selection, scaling and descaling, closer points of presence and faster time to market. Figure 1 shows an end-to-end network SFC using resources from cloud service providers (different domains).

VNFs and SFCs are created and managed by the NFV management and orchestration (NFV-MANO) [14] on VMs that are managed by management and control platform (MCP) of the cloud service provider (CSP). The ISP service (e.g., broadband) is managed by the business support system (BSS)/operation support system (OSS) owned by the ISP. Effective coordination of CSPs’ MCP, ISPs BSS/OSS and the NFV-MANO is key to successful virtual network service delivery. Figure 2 shows the interfaces between the service provider and the NFV domains as defined by ETSI.

As far as the fault and performance management are concerned, the VNF infrastructure manager (VIM) interacts with the NFVI for collecting and forwarding performance measurement and events. The VNF manager (VNFM) obtains device level statistics, logs notifications, alarms and events from the respective EMSs. NFV orchestrator (NFVO) interacts with the BSS/OSS for configuration and fault management of network services.

3 FCP problem description

Faults may occur for a number of reasons, prominent of which are malfunctioning or failed devices because of hardware or software failures in VM or VNF, failure of links and configuration errors. There could be other reasons such as cyber attacks, disasters or environmental factors. Faults appear as errors. Errors in turn are deviations of a system from normal operations. Errors are reported through system alarms. Alarms are notifications about specific events that may or may not be errors. Four levels of severity of events and alarms have been defined in ITU standard X.733: critical, major, minor, and warning [15]. The critical alarm appears when the service can no longer be provided to the user. Major alarm indicates the service affective condition while minor means no current degradation is there, but if not corrected may develop into a major fault. A warning is an impending service affecting fault or performance issue. To be able to make use of the multi-cloud paradigm effectively, it is extremely important that the fault and performance management issues are fixed for this environment [16]. The FCP system should be able to identify which issues are potential performance hazards or may result in a fault that would require resources to rectify.

There are a number of reasons why the combination of multi-cloud and NFV needs a strong fault and performance management system. Cloud reliability today does not mea-
sure up to the network services’ need of five nines. Detection and localization of faults and performance issues have a higher degree of complexity here as the anomalous behavior could be in the hardware, virtual machines, VNFs, SFCs or at the service levels. This involves coordination among management systems of the CSP, ISP and the NFV provider. The inter-se responsibilities have not been clearly demarcated. There are gaps in definition of multi-domain NFV-MANO coordination with the BSS/OSS and its effect on fault management [17]. ETSI document [14] speaks of including fault correlation and root cause analysis in the NFV system operation and maintenance. Fault escalation from lower to higher layers has not been defined. ETSI supported proof of concept (POC#35) has shown the shortcomings of the OpenStack deployment and alarming and is not geared up for multiple clouds [25]. The current problem with OpenStack is its development being driven primarily by IT community resulting in lack of some critical capabilities for NFV. Use of shallow architectures in machine learning has been reported in a number of industrial settings such as loss of performance in industrial plants using support vector machine (SVM) [26–28]. Deep learning architectures have been lately applied, e.g., deep neural network based fault diagnosis in [29,30]. However, work on fault and performance management of virtual network services has been scarce. Some examples are: artificial neural networks (ANN) for single and double alarm simulated scenarios [31], a system for fault analysis and prediction in telecommunications access network for Rijeka area of Croatia [32], fault prediction in telecommunication networks [33].

5 FCP solution components

The virtualized systems require features that are either not present in the traditional diagnostic methods or not fully exploited. Prediction of impending failure, dealing with incomplete information and analyzing trends to predict failure are some of the key requirements. Modern communication systems produce large volumes of high-dimensional operational data. In such a case, analyzing the data to get an actionable understanding of the situation becomes difficult. It would be extremely difficult, if not impossible, to capture intricate relationships between the features and the labels through the traditional methods. In general, the researchers agree on predictive approaches that take a learning route to solve the problem of the complex interaction of features of fault detection and localization [34]. We have worked on shallow as well as deep structures in machine learning in an attempt to tackle this problem. We feel that a model based on a judicious combination of these could be used for prediction of faults and performance issues along with the severity levels of impending faults with a high level of accuracy.

5.1 Model for fault detection and localization

The proposed model, shown in Fig. 3, has predictive and deductive properties to meet the FCP requirements. The detection system first decides whether there is a manifest or an impending fault or a performance issue. Based on this,
5.2 Markers and metrics for fault detection and localization

There are events relating to communication, quality of service, processing, equipment and environment that produce alarms, notifications, warning or error messages, measurements, counter values and conditions. Of course, many of the markers will appear in more than one type of fault or performance issue. Once trained the detection and localization algorithms would be able to pick out relevant markers and use them to predict the type of condition that may have arisen. Some of the markers related to mobile, fixed and broadband networks are given in Table 1.

Table 1 List of markers

| Mobile network | Fixed network | Broadband |
|----------------|---------------|-----------|
| C/I ratio      | No dial tone  | Intermittent connection |
| Radio link time out | Channel noisy | Low data rate         |
| Time slot shortage | MDF jumper dis | Phone works broadband down |
| Occupied bandwidth | Line card port faulty | Repeated training   |
| RX noise floor | Primary cable fault | LAN lamp off |
| Radio power    | Distribution cable fault | Line noisy |
| Frequency error | DP fault | DSLAM port mismatch |
| Antenna tilt   | House wiring  | No ping |
| Signal strength | MDF fuse blown | ADSL lamp flashes/off |
| BTS down       | Customer instrument fault | No line sync |
| Handover failure | Dis in one limb | Browsing issues |
| Roaming failure | Earth contact | Micro-filter faulty |
| Packet loss    | Drop wire fault | No comms |
| Hypervisor alarm | Ring tone fault | Dropouts |
| Registration failure | Message fault | No authentication |
| Low CSSR       | Delayed dial tone | |

Table 2 shows two common problems in cellular mobile networks in which markers overlap.

ETSI documents on service quality [21,22] on reliability mention metrics that need to be collected and analyzed. The ETSI group specification on service quality metrics recognizes that it is important to have an objective and quantitative metrics for good service to the consumers, fault localization and identification of the root cause of performance deviations. Examples of metrics and their realistic values (where applicable) from an actual network are given in Table 3.

5.3 Training datasets

The quality and quantity of the datasets affect the learning and prediction performance of machine learning algorithms [69]. Information about faults, observations and restoration details in the telecommunication networks is contained in the fault dockets, test reports, central office system logs, outdoor maintenance staff logs, cable maintenance staff diaries and docket closure reports. Publicly available datasets such as UCI repository [35], Stanford SNAP [36], DMOZ [37] have datasets relating to faults in contexts other than telecommunications networks. A number of authors [32] have either used proprietary datasets not publicly available or synthetic datasets.

We have used two datasets for evaluating our algorithms. One of the datasets pertains to actual faults and their severity levels in the Telstra network disruptions and the other is a synthetic dataset generated through multivariate kernel density estimation (KDE) technique augmented with real faults from a live network. The Telstra datasets (2016) are derived from the fault log files containing real customer faults [38] and adapted for this study. The training dataset (Table 4) contains location and a time point. They are identified by the “ID” column, which is the key “ID” used in other data files. Fault severity has three categories with 0 indicating no fault, 1 indicating a few faults and 2 indicating many faults. There are datasets for event type, the features logged, the resource affected and the severity type. The severity_type is different from fault severity and classifies the warning given by the system. Table 5 gives an example of one of these datasets (resource type).
Table 2  Faults and markers

| Fault                  | Phase error | Power | EVM | Rx noise floor | Origin offset | Occupied BW | Frequency error | C/I |
|------------------------|-------------|-------|-----|----------------|---------------|-------------|-----------------|-----|
| Call drop\(^a\)        | Y           | Y     | Y   | Y              | Y             |             |                 |     |
| Call blocked\(^b\)     | Y           | Y     | Y   | Y              | Y             |             |                 | Y   |

\(^a\)Radio link timeout  
\(^b\)Time slot short

Table 3  Metric for network availability and resiliency (not exhaustive)

| Metric                              | Typical value |
|-------------------------------------|---------------|
| Mobile network                      | Broadband network |
| BTS total downtime                  | ≤ 2%          |
| Connection establishment            | ISP-POP to IGSP/IXCH latency < 120 ms |
| CSSR (call set up success rate)     | ISP G/W to international NAP (terrestrial) < 350 ms |
| SDCCH/paging channel congestion    | ≤ 1%          |
| TCH congestion                      | ≤ 2%          |
| Connection maintenance              | Avg. throughput for packet data > 90% |
| CDR (call drop rate)               | ≤ 2%          |
| POI congestion                      | ≤ 0.5         |
| Signal Strength in vehicle          | ≥ 85 dBm      |
| Fixed network                       |                |
| Fault incidences                    | ≤ 5%          |
| CCR (call completion ratio)         | > 55%         |

Table 4  Training dataset

| Id       | Location | Fault severity |
|----------|----------|----------------|
| 14121    | Location 118 | 1               |
| 9320     | Location 91  | 0               |
| 14394    | Location 152 | 1               |
| 8218     | Location 931 | 1               |
| 14804    | Location 120 | 0               |
| 1080     | Location 664 | 0               |
| 14964    | Location 704 | 2               |
| 12896    | Location 613 | 2               |
| 7288     | Location 846 | 2               |
| 13300    | Location 613 | 1               |

Table 5  Table captionThe resource type dataset

| Id       | Resource_type |
|----------|---------------|
| 6597     | Resource_type 8 |
| 8011     | Resource_type 8 |
| 2597     | Resource_type 8 |
| 5022     | Resource_type 8 |
| 6852     | Resource_type 8 |
| 5611     | Resource_type 8 |
| 14838    | Resource_type 8 |
| 2588     | Resource_type 8 |
| 4848     | Resource_type 8 |
| 6914     | Resource_type 8 |

Table 6  Features and classes in the KDE dataset

| Features          | Classes          |
|-------------------|------------------|
| BTS hardware      | 1 Call drop      |
| Radio link phase  | 2 Call setup     |
| EVM               | 3 No roaming     |
| C/I ratio         | 4 Weak signal    |
| BSIC fault        | 5 No registration|
| BCC fault         | 6 No outgoing    |
| Time slot short   | 7 Data not working|
| Power             | 21 Virtual resource|
| Rx noise          | 22 ID signal strength|
| Antenna tilt      | 23 OD signal strength|
| Occupied BW       | 24 Handover      |
| Filter fault      | 25 CSSR          |
| TCH congestion    | 26 SDCCH congestion|

Generation of high-dimensional synthetic data can be done using kernel density estimation with Markov sampling. This type of data is commonly employed as a substitute for real data and can be used to provide a controlled testing environment that meets specific conditions [39]. We have used the kernel density estimator for high dimension [39]. We call this the KDE dataset (Table 6).
5.4 Discussion on shallow and deep learning methods used

In the context of this paper, we use the terms shallow structure and shallow architecture interchangeably and the same is the case with deep structure and deep architecture. Shallow structures are simpler with one stage of non-linear operation, e.g., one hidden layer in neural networks. Deep learning architectures would have more than one level of the composition of non-linear operations in the function learned. Deep neural networks have finally attracted widespread attention, mainly by outperforming alternative machine learning methods such as SVM in numerous important applications [40,41]. One of the key advantages of deep learning is the automatic extraction of high-level features from the given dataset. This is a distinct advantage over the difficult feature engineering in shallow structures that requires human intervention. In deep learning, higher level features are learned as a composite of lower level features. In this way, features are learned at many levels of abstraction making it easier to grasp complex functions that map the input to the output directly from data. As our model uses both shallow and deep structures, we discuss here briefly the important aspects of the algorithms that we have used in our study leaving the details to the references mentioned therein.

1. Support vector machine (SVM): the SVM for classification (SVC) is a supervised learning method that analyzes data and recognizes patterns. An SVM classifier produces an optimal hyperplane that separates different classes of data in the given labeled training data. The decision function is fully specified by a subset of training samples, the support vectors that lie closest to the hyperplane. This model can then be used to predict the class of new data points. Reference is made to [42] for a detailed description.

2. Alternating decision trees (ADT): this method is a variation of the standard decision tree that partitions instance space into disjoint sets by splitting the leaf nodes. Whereas in standard decision trees the leaf node is split once, in ADT it can be split multiple times. Boosting brings performance enhancements. More details are available at [43].

3. Random forest: the random forest (RF) is a useful classifier that gives good results in many situations. It can be run efficiently with large datasets. It can also produce estimates of the relative importance of the predictor variables. It does not need a separate test dataset as the OOB-error gives an unbiased estimate of test or classification error [44].

4. Deep learning through stacked autoencoders: An autoencoder is a neural network consisting of two parts, an encoder function \( h = f(x) \) and a decoder that produces reconstruction \( r = g(h) \). The encoder maps input data to a lower dimension or compressed feature representation while the decoder converts it back to the dimensions of the input space. Because the model is forced to prioritize which aspects of the input should be copied to the output it often learns useful properties of the data. A stacked autoencoder consists of multiple layers of the sparse autoencoder in which the output of each layer is wired to the inputs of the successive layer. Learning an under complete representation (with \( h \) having a smaller dimension than \( x \)) forces autoencoder to capture the most salient features of the training data. In the learning process the loss function, \( L(x,g(f(x))) \) is minimized, where \( L \) is the loss function and there is a penalty on \( g(f(x)) \) for being dissimilar from input \( x \). In a sparse autoencoder each layer is pretrained to learn an under complete or sparse representation to reduce training time and improve accuracy. The training criterion then involves a sparsity penalty \( \Omega(h) \) in addition to the reconstruction error and can be written as \( L(x,g(f(x))) + \Omega(h) \), where \( \Omega(h) \) can be thought of as a regularizer term.

If we consider a stacked autoencoder with \( n \) layers then \( h^{(n)} \) is the activation of the deepest layer of the hidden units, which gives us a representation of the input in terms of higher order features. For classification, the features from the stacked autoencoder can be used by feeding \( h^{(n)} \) to a softmax classifier. Softmax is a regression method that can work with multiple classes.

The stacked autoencoder is trained in a greedy layer-wise manner. When all the layers have been trained fine-tuning is done using back propagation to tune the parameters of all the layers and produce better results [45].

5.5 Detection of fault and performance conditions

Faults and performance issues may range from simple single point failures to multiple correlated or uncorrelated events. A fault presents itself in the form of system malfunction and notifications from faulty and other connected devices. The failure detection mechanism should be able to filter out dependent and routine operational events so that resources are not wasted in localizing these problems. In NFV, we deal with VM, VNF and virtual network faults that cause virtual network services to behave abnormally. For example, failure of a Gigabit Ethernet interface on the core router may cause some or all of the virtual private network (VPN) links of many customers to be non-functional. In the context of this paper, the goal of the FCP detection mechanism is to correlate alarms, notifications, measurements and other markers generated by events to infer manifest or predict impending performance and fault conditions.

Some errors may be cleared by the system, others may produce warnings that may signal impending problems while still another may produce faults that bring down functionalities and make themselves evident. In our implementation, the trained shallow machine-learning models learn from the past events relating to faults and their resolutions. The
models work in two stages: the first stage just makes a decision between ‘fault’ and ‘no-fault’ conditions, while the second stage does a more detailed examination of the markers to choose between ‘manifest’ and ‘impending’.

Minor faults and warnings discussed above would be the main contributors to the impending faults and need to be analyzed to make this decision. With correct segregation the localization stage would be able to carry out its functions properly. Figure 4 shows the detection process (boxes for localization are not part of detection).

5.6 Localization of fault and performance conditions

The severity level of the faults would indicate whether they are warnings, minor, major or critical. In the case of major and critical faults, devices have degraded performance or have stopped working and need immediate action. Minor faults do not affect service and can be scheduled for localization accordingly. Warnings, along with the state information, provide insight into the degrading health of devices and could signal a major impending fault. We have evolved a multi-layer fault identification and localization system. At Layer 1, it detects the broad category of fault and then at Layer 2 does a fine-grained classification. In the case of impending faults, the system predicts the locations and severity levels of the developing faults. Table 7 gives examples for each of the layers and associated markers.

Both the machine learning models for detection use kernel-based binary SVM. For localization of manifest faults initial tests have been carried out using SVM with multi-class modification and Random Forest. Localization of impending faults and performance issues has been implemented using stacked autoencoder—a deep neural network. Some of the major advantages of this algorithm are automatic selection

| Table 7 | List of performance and fault issues (not exhaustive) |
|---------|-----------------------------------------------------|
| **Layer 1** | **Layer 2** | **Markers** |
| Network availability | • Base station distance | Call drop rate (CDR), signal strength, multi-path interference |
| | • Antenna height | |
| | • Backhaul failure | Roaming not functional, GGSN alarm |
| | • User equipment (UE) settings | |
| | • Power module failure | BTS downtime, No. of affected cells power failure alarm/notification |
| | • Radio unit failure | |
| | • Insufficient TCH | Traffic channel (TCH) congestion, blocking |
| | • Locked transceivers (TRXs) | |
| Connection maintenance | • Hopping sequence no (HSN) clash | Handover success rate |
| | • LAC boundary | |
| Network performance | • BCCH, TCH freq. plan | Bad Rx quality |
| | • Handover parameters | |
| | • Antenna tilt | |
| | • PGW fault | Packet loss |
| Security | • Radio network fault | |
| | • DOS attack | Tenant authentication failure |
| | • Home subscriber server (HSS) failure | |
| | • Record deletion | |
| Virtualization-component failure | • EM fault | Hardware failure alarm |
| | • NE failure | |
| Virtualization–Software vulnerabilities | • Physical resource congestion | Hypervisor alarm |
of features from high-dimensional data and filtering information through the layers to achieve better accuracy. For training and testing our models, we adapted a composite feature dataset from the ones described in Sect. 5.3. The implementation was done in MATLAB running on 2.7 GHz Intel Core i7 with 8 GB 1600 MHz DDR3 RAM and NVIDIA GeForce GT 650M 1024 MB graphics. In the next section, we present our evaluation results.

6 Evaluation results

6.1 Detection with shallow structures

Initial classification tests were carried out with SVM, ADT and random forests for binary ‘fault’, ‘no-fault’ classification. Each of the models was trained with 240 examples and 10% cross-validation. In the comparative test (Table 8) it can be seen that SVM performs better than the other methods with ≥ 95% accuracy. It can also be seen that true positive rate (TP) is high and false positive rate (FP) is low for the fault class 1 for SVM. High precision shows that no problem cases were correctly classified in SVM and RF.

Selection of SVM was ratified by running it on the KDE dataset. The KDE dataset has been generated using kernel density estimation and Markov sampling. An extract can be seen in Table 9. Only eight of the 26 features mentioned in Table 6 have been used to simplify data preparation and yet keep the data close to the real network. The data contain values of the selected features along with the severity of each fault case. The first model trained to detect faults (including warnings) given the feature values for all the new cases. It has to decide whether the given information indicates a possible fault or no detectable fault is there.

The second model was trained to analyze the values of the features of all the faults and to decide whether there is a manifest fault or an impending fault. As discussed earlier, the features interact in a complex way to decide network performance. Not all warnings become faults in the future. The acid test for the predictive trained model was to examine whether it correctly predicts the risk of impending fault or performance issue that would need a truck roll out or other maintenance actions. The SVM model had nearly 100% accuracy of predicting the warning cases in our test sets.

We compared ours with fault detection in other areas. In [18] the authors used SVM to classify wind turbine faults using operational data and achieved 99.6% accuracy. In [27] wind turbine faults were detected with accuracy 98.26% for linear SVM and 97.35 for Gaussian. The authors in [19] studied faults in rotating machinery and got 99.9% accuracy of classification with SVM, higher than other methods studied.

6.2 Localization with deep structure

As mentioned before, we have implemented and tested localization of impending fault to predict where the impending fault might eventually happen and with what severity level. To evaluate the model we used the Telstra network fault

| Table 9 | Extract of training dataset for detection |
|---------|------------------------------------------|
| Docket# | C/I Ratio | Power margin (dBm) | POI cong (%) | CSSR (%) | TCH cong (%) | SDCCH cong (%) | Signal strength dBm | Packet loss (%) | Severity |
| 23      | 49        | 22                  | 6           | 99       | 0           | 0            | 96               | 0              | 2        |
| 52      | 10        | 13                  | 8           | 83       | 10          | 0             | 109              | 2              | 1        |
| 68      | 28        | 20                  | 1           | 96       | 1           | 1             | 100              | 1              | 0        |
| 69      | — 10      | — 15                | 11          | 91       | 6           | 0             | 98               | 0              | 1        |
| 134     | 67        | 25                  | 14          | 85       | 3           | 4             | 80               | 2              | 1        |
| 201     | 49        | 15                  | 1           | 98       | 2           | 0             | 95               | 2              | 0        |
| 215     | 49        | 75                  | 6           | 99       | 5           | 0             | 75               | 0              | 3        |
datasets described in Section V(C). The training dataset has 7382 records, the feature dataset 58672 records and resource type 21077 records. Similarly, the other files are also fairly large. The feature set used for analysis is given in Table 10.

Fault_severity levels are: No fault (0), a few faults (1) and many faults (2) and are based on actual faults reported by users. Severity_type, on the other hand, describes the intensity of the warning, which can be used to predict impending faults. The stacked autoencoder was setup with 100 hidden layers in the first and 50 in the second autoencoder. The Softmax layer provides supervised back-propagation improvement of the weights learned during unsupervised training. A varying number of epochs (iterations) were used for different layers to obtain best results. Figure 5 gives a representation of the stacked autoencoder.

With default sparsity regulation of 1, the confusion matrix shows 97.3% accuracy of prediction (Fig. 6) of severity levels of impending faults. With sparsity regularization parameter of 4 and sparsity proportion 0.1 (parameters arrived using grid search), the confusion matrix indicates 100% accuracy for the given training and test datasets (Fig. 7). The accuracy of the model is dependent on the size of the hidden layers compared to the input data. Also, the relative sizes of the hidden layers affect the accuracy. Figure 8 shows the accuracy and mean square errors (MSE) for various sizes of the first hidden layer (H1) with the size of H2 as a parameter. The idea is to show that accuracy and MSE are good together for a certain range of H1 given the size of H2 and the parameters need to be carefully selected.

Comparing with results obtained by other researchers in studies of HVAC fault detection with deep learning deep belief network (DBN) 2 layer model the overall accuracy

| Table 10 | Features from network fault datasets |
|----------|--------------------------------------|
| 1        | Id                                   |
| 2        | Location                             |
| 3        | Features 1–386                       |
| 4        | Volumes for features                 |
| 5        | Resource type 1–10                   |
| 6        | Severity type 1–5                    |
| 7        | Event type                           |
| 8        | Fault severity                       |

![Fig. 5](image5.png) The stacked encoder used for prediction

![Fig. 6](image6.png) Confusion matrix without sparsity regularization

![Fig. 7](image7.png) Confusion matrix with sparsity regularization

![Fig. 8](image8.png) Effect of hidden layers on accuracy and MSE
was claimed to be $\geq 95\%$ [46]. Our results compare favorably with these results.

7 Summary

In this paper, a study of the markers and metrics related to fault and performance management of communication networks has been performed. Handling fault and performance anomalies when they occur is crucial for the success of NFV deployments over clouds. We propose a model for detection and localization of manifest and impending fault and performance issues. Some of the aspects of detection and localization of faults have been implemented using shallow and deep structures, respectively. It has been observed that SVM for classification performs well for detection of fault/no-fault or manifest/impending fault situations. This information could then be used with localization function for deeper analysis of the warnings to predict impending faults and their severity. Deep structure of stacked autoencoder was used for careful examination of ‘warning’ cases to predict the severity of faults that may arise in future. This would help in planning resources for maintenance activities. The models have been evaluated with actual network and synthetic datasets with good results. It is proposed to carry out further work on fine-grained localization for identification of resources, location and nature of the problem for more actionable information.

References

1. Network functions virtualisation: an introduction, benefits, enablers, challenges & call for action (2012). ETSI Whitepaper published at the SDN and OpenFlow World Congress, Darmstadt-Germany
2. ETSI ISG Network Functions Virtualization (ISG NFV) (2016) http://www.etsi.org/technologies-clusters/technologies/nfv. Accessed Dec 2016
3. ETSI GR NFV-IFA 015 V2.1.1 (2017) Network functions virtualisation (NFV) release 2; management and orchestration; Report on NFV Information Model
4. Bhamare D, Jain R, Samaka M, Erbad A (2016) A survey on service function chaining. J Netw Comput Appl 75:138–155
5. (2000) Recommendations M.3010 Series M: TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile, and leased circuits TMN management functions
6. Bhamare D, Jain R, Samaka M, Erbad A, Gupta L, Chan HA (2017) Optimal virtual network function placement and resource allocation in multi-cloud service function chaining architecture. Comput Communications
7. (2002) ITU-T Recommendation M.3400 Series M: TMN AND network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits TMN management functions
8. (1992) ITU-T Recommendation X.733, information technology-open system interconnection-systems management-alarm reporting function
9. (2013) ETSI GS NEV 002, Network functional virtualization; architectural framework
10. Gupta L, Samaka M, Jain R, Erbad A, Bhamare D, Metz C (2017) COLAP: a predictive framework for service function chain placement in a multi-cloud environment. IEEE 7th Annual Computing and Communication Workshop and Conference (CWCW), Las Vegas
11. Lopez D (2014) Network functions virtualization: beyond carrier-grade clouds. In: Optical fiber communications conference and exhibition (OFC), pp 1–18, IEEE
12. Han B, Gopalakrishnan V, Ji L, Lee S (2015) Network function virtualization: challenges and opportunities for innovations. IEEE Commun Mag 53(2):90–97
13. (2015) IETF RFC: 7665, J. Halpern, C. Pignataro, Service function chaining (SFC) architecture
14. (2014) ETSI GS NFVMAN001 V.1.1.1, Network Functions Virtualization (NFV); Management and Orchestration
15. (2008) ITU-T Recommendations E.800 [i.10]: Definitions of terms related to quality of service (2008)
16. Priyadarsini RJ, Arockiam L (2013) Failure Management In Cloud: An Overview. Int J Adv Res Comput Commun Eng
17. Gap analysis on network virtualization activities draft-berndos-nfvr-gaps-network-virtualization-00, (2016)
18. Leahy K, Hu RL, Konstantakopoulos IC, Spanos CJ, Agogino AM (2016) Diagnosing wind turbine faults using machine learning techniques applied to operational data. In: International conference on prognostics and health management (ICPHM)
19. Nicchiotti G, Fronzaigleat L, Etienne L (2016) Machine learning strategy for fault classification using only nominal data. In: European conference of the prognostics and health management society
20. Kompella RR, Yates J, Greenberg AG, Snoeren AC (2010) Fault localization via risk modeling. IEEE Trans Dependable Sec Comput 7(4):396–409
21. (2015) ETSI GS NFV-REL 001 V1.1.1 Network functions virtualisation (NFV); resiliency requirements
22. (2014) ETSI GS NFV-INF 010 V1.1.1 Network Functions Virtualisation (NFV); service quality metrics
23. Mijumbi R, Serrat J, Gorricho J-L, Latre S, Charalambides M, Lopez D (2016) Management and orchestration challenges in network function virtualization. IEEE Commun Mag 54(1):98–105
24. Boucadair M, Jaqueten C (2015) Handbook of research on redesigning the future of Internet architectures. IGI Global
25. Kunzmann G, Goncalves C, Mibu R (2015) OPNFV–doctor fault management. OPNFV Summit, Linux Foundation
26. de Souza DL, Granzotto MH, de Almeida GM, Oliveira-Lopes LC (2014) Fault detection and diagnosis using support vector machines — A SVC and SVR Comparison. J Saf Eng 3(1):18–29. http://doi.org/10.5923/j.safety.20140301.03
27. Santos P, Villa LF, Reñones A, Bustillo A, Maudes J (2015) An SVM-based solution for fault detection in wind turbines. Sensors 15(3):5627–5648
28. Samil M, Sharma SK, Singh R (2015) Fault detection and classification in electrical power transmission system using artificial neural network. SpringerPlus 4(1):334
29. Lv F, Wen C, Bao Z, Liu Meiqin (2016) Fault diagnosis based on deep learning. In: American Control Conference (ACC)
30. Zhang L, Lei Y, Chang Q (2017) Intermittent connection fault diagnosis for CAN using data link layer information. IEEE Trans Indus Electronics
31. Yilmaz A (2017) Comparative study for identification of multiple alarms in telecommunication networks. Turk J Electr Eng Comput Sci 25(2):677–688
32. Rozaki E (2015) Network fault diagnosis using data mining classifiers. Int J Data Min Knowl Manag Process
33. Jaudet M, Iqbal N, Hussain A, Sharif K (2005) Temporal classification for fault-prediction in a real-world telecommunications network. In: Proceeding of the International conference on emerging technologies
34. Lee BY, Lee BC (2016) Fault localization in NFV framework. In: 18th International Conference on Advanced Communication Technology (ICACT), PyeongChang, Korea
35. Lichman M (2017) UCI machine learning repository, Irvine, CA: University of California, School of Information and Computer Science. https://archive.ics.uci.edu/ml/datasets.html Accessed Dec 2013
36. Stanford SNAP: large network dataset collection. https://snap.stanford.edu/data/. Accessed Jan 2017
37. DMOZ open directory project (ODP). http://www.dmoz.org/docs/en/about.html. Accessed Jan 2017
38. Kaggle datasets, https://www.kaggle.com/datasets
39. Botev Z (2016) Fast multivariate kernel density estimation for high dimensions. https://ch.mathworks.com/matlabcentral/fileexchange/58312-kernel-density-estimator-for-high-dimensions. Accessed Jan 2017
40. Mhaskar HN, Poggio T (2016) Deep vs. shallow networks: an approximation theory perspective. Anal Appl 14(06):829–848
41. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural networks 61:85–117
42. Smola AJ, Scholkopf B (2004) A tutorial on support vector regression. Stat Comput 14(3):199–222
43. Jabber MA, Deekshatulu BL, Chandra P (2014) Alternating decision trees for early diagnosis of heart disease. In: Proceedings of international conference on circuits, communication, control and computing, pp 322–328
44. Louppe G (2014) Understanding random forests—from theory to practice. PhD Dissertation, University of Liege, France
45. Stacked autoencoder (Unsupervised Feature Learning and Deep Learning). http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders. Accessed Jan 2017
46. Lee D, Lee B, Shin J. W (2015) Fault detection and diagnosis with modelica language using deep belief network. In: Proceedings of the 11th international modelica conference