HYPER-VINES: A **HY**brid Learning Fault and **P**erformance Issues **ER**adicator for **VI**rtual **NE**twork **S**ervices over Multi-Cloud Systems

Lav Gupta
Dept. of Computer Science & Engineering
Washington University in St. Louis
St. Louis, USA
lavgupta@wustl.edu

Tara Salman
Dept. of Computer Science & Engineering
Washington University in St. Louis
St. Louis, USA
tara.salman@wustl.edu

Ria Das
Dept. of Computer Science & Engineering
Washington University in St. Louis
St. Louis, USA
ria.das@wustl.edu

Aiman Erbad
Dept. of Computer Science & Engineering
Qatar University
Doha, Qatar
aerbad@qu.edu.qa

Raj Jain
Dept. of Computer Science & Engineering
Washington University in St. Louis
St. Louis, USA
jain@cse.wustl.edu

Mohammed Samaka
Dept. of Computer Science & Engineering
Qatar University
Doha, Qatar
samaka.m@qu.edu.qa

Abstract—**Fault and performance management systems, in the traditional carrier networks, are based on rule-based diagnostics that correlate alarms and other markers to detect and localize faults and performance issues. As carriers move to Virtual Network Services, based on Network Function Virtualization and multi-cloud deployments, the traditional methods fail to deliver because of the intangibility of the constituent Virtual Network Functions and increased complexity of the resulting architecture. In this paper, we propose a framework, called HYPER-VINES, that interfaces with various management platforms involved to process markers through a system of shallow and deep machine learning models. It then detects and localizes manifested and impending fault and performance issues. Our experiments validate the functionality and feasibility of the framework in terms of accurate detection and localization of such issues and unambiguous prediction of impending issues. Simulations with real network fault datasets show the effectiveness of its architecture in large networks.**

Keywords— Network Function Virtualization, Virtual Network Function, Service Function Chain, Virtual Network Service, Multi-Cloud Environments, Fault Management, Performance Management, FCAPS, Machine Learning, Deep Learning

I. INTRODUCTION

Virtualization of datacenter resources has been immensely successful in the Information Technology (IT) world. Carriers (an entity providing telecommunication or Internet services) see Network Function Virtualization (NFV) as a paradigm that could help them transpose this success to their networks by instantiating network functions on virtual resources, like Virtual Machines (VMs), hosted on commercial, off-the-shelf servers. The resulting Virtual Network Functions (VNFs), e.g., virtual routers and virtual load-balancers, form the basic building blocks of the Virtual Network Services (VNSs) like cellular mobile service and broadband service. To carriers, such deployments mean freedom from proprietary solutions, ease of scaling, reduced cost of operation and reduced time to market. Additionally, instantiating a VNS on multiple clouds adds advantages like proximity to users and avoidance of a single point of failure.

The catch in this utopian scheme is that VNSs do not yet meet the requirements of five-nines availability and quality of service that the traditional, largely physical and standards-based carrier networks have been assiduously built to provide [1]. Areas like the quality of service, performance monitoring and useful metrics are current research challenges relating to NFV [2]. Our study shows that the lack of credible Fault, Configuration, Accounting, Performance and Security (FCAPS) standards, the complexity of architecture and ill-defined interfaces, among the involved management platforms, are the primary causes for these services not measuring up.

In this paper, we make a case for the **HY**brid Learning Fault and **P**erformance Issues **ER**adicator for **VI**rtual **NE**twork **S**ervices over a Multi-cloud (HYPER-VINES) framework that will improve the availability and reliability of VNSs thereby benefitting the carriers as well as their subscribers. This framework is designed to work with the available markers (alarms, notifications, counter values and measurements) and uses an innovative combination of predictive shallow and deep machine learning models for detection and localization of faults and performance issues in operational VNS deployments. Symbolically, much like friendly vines help in choking harmful weed in plantations, HYPER-VINES reaches deep into networks to weed out fault and performance issues.

The framework achieves detection and localization accuracies which are substantially better than the baseline and any reported results in a similar environment. The main contributions of this work are summarized below:

i) Examination of the major reasons for performance and availability challenges in NFV and cloud-based VNS deployments. We find that handling detection and localization of faults and performance issues is difficult because of multiple layers in implementation and distributed and overlapping responsibilities for fault management. Besides, incomplete definition of interfaces among cloud, NFV and operation support platforms worsen the situation.

ii) Development of mechanisms, within the described architectural framework, which make use of the network's operational markers for detection and localization of faults and performance issues.

iii) The innovative use of shallow and deep predictive algorithms to obtain high accuracy of detection and localization. We achieve accuracies markedly better than the baselines and any other reported result in similar environment.

iv) Demonstration of the feasibility and effectiveness of the proposed HYPER-VINES framework using real network data

II. Background

A. Terminology Related to VNSs

Figure 1 illustrates a broadband service created on virtual resources, i.e., as a VNS. It consists of a Service Function Chain (SFC), an ordered and linked collection of VNFs, containing four VNFs, viz., an aggregation switch, two types of Broadband Remote Access Servers (BRAS) and a core router. It also has multiple instances of a Physical Network Function (PNF), viz., Digital Subscriber Line Access Multiplexers (DSLAMs), retained from the legacy networks. The switch has a built-in load balancing function for distributing traffic between the two forked paths.

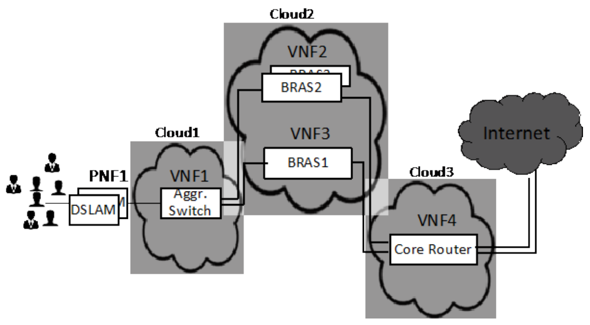


Fig. 1. Broadband Access Virtual Network Service

B. Fault and Performance Management of VNS over Multiple Cloud Systems

In VNSs over multi-cloud systems, we deal with three interacting platforms. (i) The Management and Orchestration (MANO) platform that creates and manages VNSs over the virtual resources provided by one or more cloud service providers, (ii) A multi-cloud management and control platform (MMCP) that optimizes and manages the complete placement across all used clouds and collects performance and other telemetry information from various clouds [3] and, (iii) The Operation Support System (OSS) that manages the carrier's network. Figure 2 shows how these platforms are positioned relative to each other. Communication among all the platforms, flows through the reference interconnection points as shown. The broken line shows that the specific interconnection has not been defined yet.

The responsibility of the fault and performance management of VNS, within MANO, is distributed among its three main modules (Figure 2). The Virtualized Infrastructure Manager (VIM) records the events and collects performance measurements of the resources which are part of a service provider's NFV-Infrastructure (NFVI) domain. The Virtual Network Function Manager (VNFM) manages VNFs and carries out fault and performance related functions for them. The NFV Orchestrator (NFVO) interacts with the carrier's OSS for fault management of network services [4].

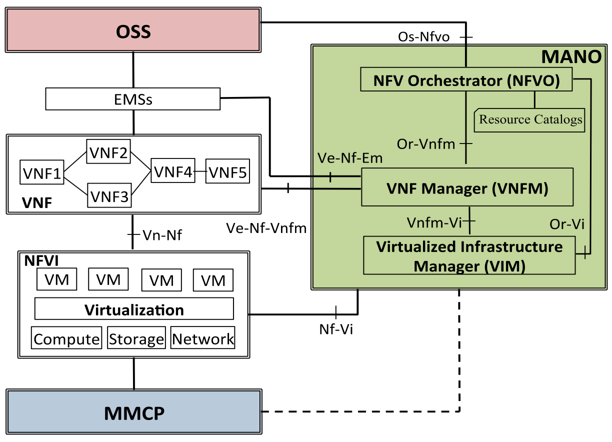


Fig. 2. Virtual Network Service and its Management

C. Unsuitability of the Existing Systems in the VNS Environments

The traditional systems are not equipped to handle the complexities of the VNS environments. The former's built-in diagnostic systems directly probe the hardware and software through well-defined interconnections between the control unit and other sub-systems. Even though the possible set of markers is relatively small, maintenance personnel still have to work through the linked set of tables of possible causes to reach the root cause of the problem. VNSs have the added complexity of having multiple layers - physical infrastructure, virtual resource and VNFs - all possibly owned by different providers. The same faults may have a different appearance across layers. For instance, a bootable disk failure in the physical layer may manifest as a VM crash in the virtual resource layer and as a failure of a VNF instance in the virtual network function layer. The failed VNF could be, for example, a virtual router in a broadband SFC and would result in a low-performance issue or a total failure of the service.

The situation in VNS gets even more intractable and fuzzy, as the locations of the appliances, middleboxes and links, that form part of a service, do not remain fixed. Many instances relating to the same appliances may be created on different servers and even on different clouds. Virtual resources are routinely migrated for optimization of factors like cost and performance. Any fault and performance framework, which attempts to work in the NFV multi-cloud VNS environment, has to meet the following challenges:

Challenge 1: The framework should be able to handle gaps in the specifications proposed by standards bodies [2] [5]. Additionally, it should be able to reconcile overlapping responsibilities among the management platforms. Both the OSS and the MANO have to manage the VNS jointly. Both the MMCP and the MANO are jointly involved in the creation of the virtual infrastructure on which the VNFs are instantiated.

Challenge 2: There are many more layers of abstraction in virtualized networks than in physical networks. Pinpointing the location and exact nature of the fault is not trivial.

Additionally, the variety of issues relating to the fault and performance problems that can affect such a system is large, making the work of diagnosis difficult [6] [7].

Challenge 3: The data is high dimensional making extraction of anomalous conditions difficult. Markers pertaining to different events overlap, making differential diagnosis difficult.

A number of researchers, and standards bodies like ETSI, believe that the fault and performance management in the NFV environment needs predictive analysis [8] [9]. In Section IV, we will see the design of the proposed framework for such analyses.

III. Related Work

Relevant to FCAPS are ETSI specifications of NFV resiliency requirements [9] and service quality metrics [10]. The former provides a list of faults and relationship between them while the latter gives VNF related metrics useful for quality of service. Both ETSI and ONAP describe the specifications for fault management support functionality [11] [12].

There has been extensive work on performance modeling systems for distributed Internet applications of the *pre-NFV era*, notably TIPME (2000) [13], Pinpoint (2002) [14] and Magpie (2003) [15]. 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-Resoner,' which uses machine learning to detect anomalies involving multiple flows or users. It compares the current distribution with stored average distributions [16]. The OPNFV Doctor project deals with the problems of the underlying hardware [7].

Most techniques relate to the IT environment with the aim to achieve three nines availability as against the five nines required for carrier networks. These techniques work on the static or dynamic dependency models, which makes previously unobserved faults difficult to detect. These methods are limited by the implausibility of having up-to-date models in a dynamic environment. They deal with faults in the physical compute components. The NFV over cloud networks have a virtual network function layer that calls for a totally different set of markers, metrics and methods.

The proposed HYPER-VINES framework has been designed to overcome the problems of the legacy frameworks and work in a multi-cloud VNS environment. It works on markers at physical and virtual levels, works with multiple instances, can discover known and unknown fault and performance issues, and predict impending faults with good accuracy. In the next section, we discuss this framework in detail.

IV. Design and Implementation of HYPER-VINES

The operational basis of the HYPER-VINES framework is to consume markers from large volumes of multi-source and high dimensional operational data to accurately detect and localize faults and performance issues of VNSs in a carrier's environment. Figure 3 gives a simplified illustration of the process.

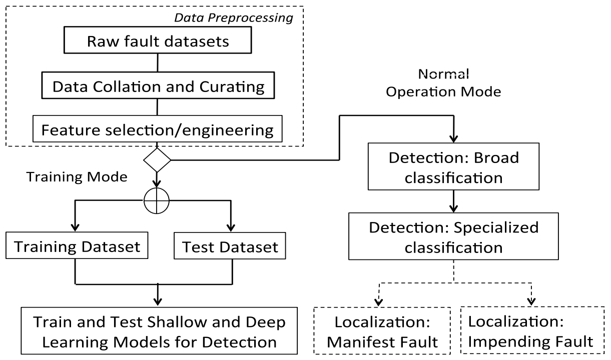

Data Preprocessing
Raw fault datasets
Data Collation and Curating
Feature selection/engineering
Normal Operation Mode
Detection: Broad classification
Training Mode
Training Dataset
Test Dataset
Detection: Specialized classification
Train and Test Shallow and Deep Learning Models for Detection
Localization: Manifest Fault
Localization: Impending Fault

Fig. 3. HYPER-VINES Fault and performance management mechanism

In the training mode, the data is curated and partitioned into training and test datasets. Shallow machine learning and deep learning models, used in various stages of detection and localization, are trained and tested. The models are fine-tuned till the mean square error of classification or prediction is optimized. The framework is called hybrid as it involves both machine leaning and deep learning models. During operation, the generated markers are pre-processed and run through the trained models for detection and localization. We discuss more details of the process in the next sub-section.

A. Architecture of the Proposed Framework

The relationship of the HYPER-VINES framework with its environment is shown in Figure 4. It collects performance markers primarily from MANO, OSS and the cloud management platforms over standard interfaces. Additional information about operational statuses of individual VNFs comes from the EMS via OSS or is pulled directly by HYPER-VINES. Since HYPER-VINES obtains markers from multiple sources, it mitigates the problem of overlapping responsibilities and ill-defined interfaces of the management platforms.

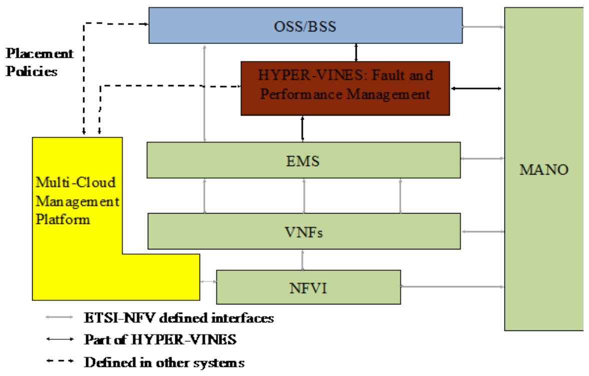

OSS/BSS
Placement Policies
HYPER-VINES: Fault and Performance Management
EMS
MANO
Multi-Cloud Management Platform
VNFs
NFVI
ETSI-NFV defined interfaces
Part of HYPER-VINES
Defined in other systems

Fig. 4. The environment of HYPER-VINES

The internal architecture of HYPER-VINES is shown in Figure 5. The main sub-systems are the Detection and Localization functions with an optional inclusion of the data pre-processing module. We discuss the important details of the framework below.

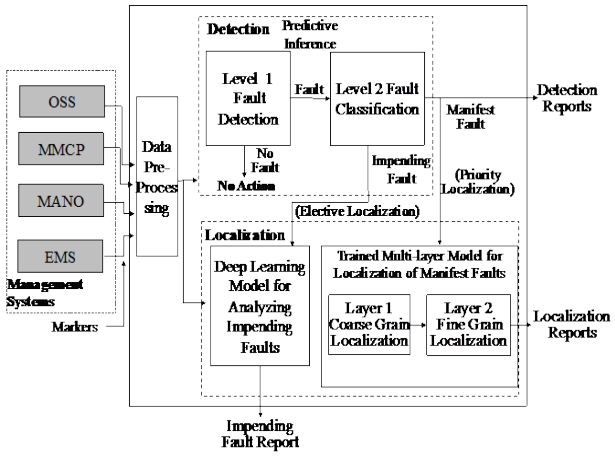


Fig. 5. Architecture of HYPER-VINES

i) Data Pre-Processing: The marker data, obtained from various sources, are collated and normalized to remove biases. The pre-processing policy may involve reduction of features based on some criterion like correlation with the labels. Tools like Weka can be used to select features based on correlation with the labels [24]. With the dataset used in this study, inclusion of features up to a correlation threshold of 23% improved accuracy. This may be changed based on the dataset used. 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. We shall see more details of the actual datasets in the next section.

ii) The Detection Subsystem: We have organized the detection subsystem into two levels of chained binary classification. Level 1 uses a shallow machine learning model to filter out the no-fault cases, which are much larger in number than fault cases. Level 2 classifies 'fault' cases into 'manifested' and 'impending' fault conditions. A two-stage model removes 'skewness' in the data and performs better than one level classifiers [17]. It also enhances the accuracy of prediction [18]. Algorithm 1 describes the process succinctly. X and Y are predictor variable and labels, respectively, in training or test datasets. 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.

```
Algorithm 1: Detection Levels 1 & 2
procedure detect_level1 (X,Y)
#fault/no-fault classification
{p_d} ← values of hyper-parameters for the chosen model
use trained model for detect_level1 with X,Y, {p_d})
if 'fault' is true
    call detect_level2 (X',Y')
produce detection report
procedure detect_level2 (X',Y')
# classify as manifest/impending and call localization
{p_d'} ← values of hyper-parameters for the chosen model
use trained model for detect_level2 with X,Y, {p_d'}
if manifest is true
    call manifest_localization (X,Y) #defined in Algorithm2
elseif impending is true
    call impending_localization (X,Y) #defined in Algorithm2
```

iii) The Localization Subsystem: 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 Section V). Details of the models and strategies *manifested and the impending* fault *classes* are explained below.

```
Algorithm 2: Localization Layers 1 & 2
procedure manifest_localization (X,Y)
# Coarse grain localization
{p_s} ← values of hyper-parameters for the chosen model
call localize_layer1( X,Y,{p_s})
# fine grain localization with the appropriate model
if class_category ==1
    {p_1} ← hyper-parameters class_category 1
    call localize_layer2(X",Y",{p_1})
    …
if class_category==7
    {p_7} ← hyper-parameters class_category 7
    call localize_layer2(X",Y",{p_7})
produce localization report
procedure localize_layer1(X,Y, {p_s})
use trained model localize_layer1 with X,Y, {p_s}
procedure localize_layer2(X",Y",{p_n})
use trained model localize_layer2 with X,Y, {p_n}
procedure impending_localization (X,Y)
{p_i} ← parameters neurons, sparsity parameters
use deep_learning_model (X,Y,{p_d}
produce impending fault report
```

Manifest Faults: The manifested fault analysis has been designed as a multi-layered, multi-class strategy based on actual data from a carrier's network. For this study, we have limited the broad fault classes, at Layer 1, to 7 (Table I). At Layer 2, we have N (=7 in our case with 7 broad classes) separate multi-class models, one for each of the broad classes. Some examples of Layer 2 issues that would fall under *Class Category 1 (Network Availability)* are *antenna height, backhaul failure, traffic channel congestion, radio unit failure* and *power module failure*. For the multi-class classification with SVM, we chose to work with simple models like One vs. One (OvO) and One vs. All (OvA) [19]. 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 $f(x)=\text{argmax}_i f_i(x)$, i.e., choose the class that classifies the example with the maximum margin.

TABLE I. LOCALIZATION LAYER 1 FAULT CLASSES

Class #	Class Category	Class #	Class Category
1	Network Availability	5	Virtualization-Component Failure
2	Connection Maintenance	6	Virtualization – Software vulnerabilities
3	Network Performance	7	Miscellaneous
4	Security		

Impending Faults: In traditional systems, in the absence of predictive analysis, preventive maintenance is relied to catch issues early. In HYPER-VINES localization of impending faults consists of predicting the severity and location of the fault. An operational network produces data continuously. In a stable network, most of these would be 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! Thus, the data are quite sparse. Impending faults may also contain previously unseen faults. Thus, while manifest 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 networks). A single SAE contains an input, an output and a hidden layer. With an undercomplete hidden layer, the autoencoder is forced to learn the most useful features (automatic feature selection). The advantage can be accentuated with stacking a number of autoencoders and carefully designing the hidden layers [20].

Figure 6 shows the stack of three sparse autoencoders used in this work. Input layer [**x**], an output layer [**p**] and three hidden layers consisting of paired encoders and decoders. The colored neurons show three matching 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.

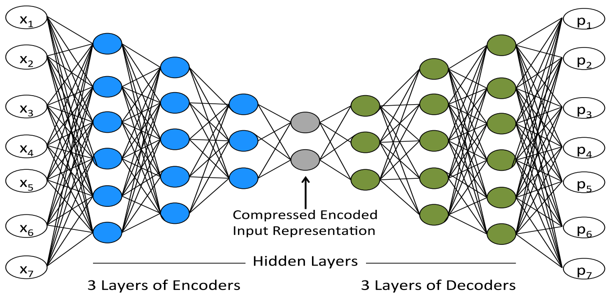


Fig. 6. Stacked Sparse Autoencoders

We train our model to have good reconstruction of the input at the output (decided by the L2-norm), with unsupervised data, in a layerwise greedy method (one hidden layer at a time). A model that reconstructs well also gives good predictions [24]. During training, features (**z**) learned by each hidden layer are input to the next layer. Pairs of {weights, biases}, viz., ($\omega 1$, b_1), (ω_2, b_2) and (ω_3, b_3), are learned in achieving good reconstruction.

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

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

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

After achieving good reconstruction of the input, the decoders are removed and a prediction layer is added in tandem with the encoded representation layer. This layer is trained in a supervised manner to learn ω_4 and produce predictions y' for given labels y. ω_4 are the weights for minimum prediction MSE (mean square error). Thus, for labels y and its prediction y' we have,

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

The model is fine-tuned using back-propagation, for improving predictions [20] [22].

V. Evaluation Of HYPER-VINES And Discussion Of Results

In this section, we discuss the dataset used and the evaluation of the proposed approach. The performance of the HYPER-VINES framework is demonstrated by good accuracies achieved by the detection and the localization subsystem.

A. Analysis of the Dataset Used

Having been drawn from the real fault logs of the Telstra Telecommunications' network, the Telstra dataset, provides a good basis for evaluating our models [23]. The complete dataset consists of sub-datasets for ***resource_type, event_type, event_volume, features, fault_severity and severity_type.*** All sub-datasets have ***'id'*** as the common field. *The event_type* sub-dataset encodes the fault and performance events. Each event is given an *id*, which also gives a chronology of these events. This sub-dataset contains 31,170 events recorded with 53 unique event types. There are 21,076 examples in the *resource_type* sub-dataset with many events involving multiple resources. The *features* sub-dataset with 58,671 records gives reported markers with each incident generating some of the *feature1* to *feature386*. The training and test datasets have *id, location,* and *fault_severity*. The *fault_severity* feature describes the severity level of a fault event and has values 0, 1, or 2 for no faults, a few faults, and many faults, respectively. The *severity_type* feature describes the intensity of the warning with values 1 to 5 with 5 being the highest.

TABLE II. FEATURES FROM NETWORK FAULT DATASETS

1	*Id (1)*	5	*Resource type (10)*
2	*Location (1)*	6	*Severity type (1)*
3	*Features (386)*	7	*Event type (5)*
4	*Volumes for features (386)*	8	*Fault severity (1)*

Table II gives a list of features contained in the sub-datasets mentioned above with the number in brackets indicating how many can occur in one event. There are about 800 features (columns) in the consolidated dataset. Following the mechanism given in Section IV, pre-processing of data, including feature selection, has been carried out using label-class correlation method in weak to prepare the training and test datasets [24].

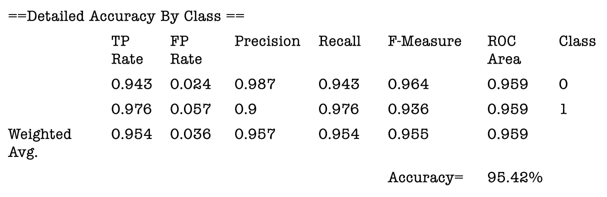
B. Evaluation of the Framework

We provide, in this sub-section, the results of the evaluation of the detection and localization parts of the framework. Table III gives the metrics used.

TABLE III. 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	

i) Detection: The Telstra dataset prepared, as explained in Subsection V (A), was used in the binary supervised two-stage model for 'fault'/'no-fault' and 'manifested'/'impending' classification. Various standard algorithms were tested for the two stages (called Level 1 and Level 2) in the framework. Tuning of various parameters, peculiar to any model, was carried out judiciously. In the comparative test, it was seen that Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel stands out with ≥ 95% accuracy for Level 1, with a high ratio of true positives to false positives, indicating good fault detection. Precision is high, indicating 'no-fault' cases are correctly classified. Compared to these results, the baseline results with Zero-R (which predicts the majority class) has 65% accuracy. We see some of the simulation results in Figure 7.

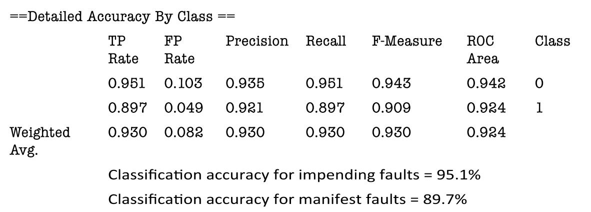


==Detailed Accuracy By Class ==

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.943	0.024	0.987	0.943	0.964	0.959	0
	0.976	0.057	0.9	0.976	0.936	0.959	1
Weighted Avg.	0.954	0.036	0.957	0.954	0.955	0.959	
					Accuracy=	95.42%	

Fig. 7. Level 1 detection accuracy, using Telstra dataset

For the Level 2 classification into manifested/impending classes again a tuned SVM with RBF Kernel works well (Figure 8).



==Detailed Accuracy By Class ==

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.951	0.103	0.935	0.951	0.943	0.942	0
	0.897	0.049	0.921	0.897	0.909	0.924	1
Weighted Avg.	0.930	0.082	0.930	0.930	0.930	0.924	

Classification accuracy for impending faults = 95.1%

Classification accuracy for manifest faults = 89.7%

Fig. 8. Level 2 detection accuracy, using Telstra dataset

For baseline at Level 2, we have used One-R, a simple but accurate classification algorithm, which generates one rule for each predictor and then selects the one with the smallest error [25]. Accuracy for HYPER-VINES is 95%, which is markedly higher than the baseline as shown in Figure 9.

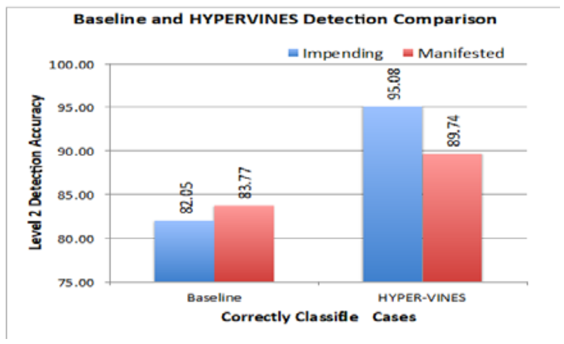
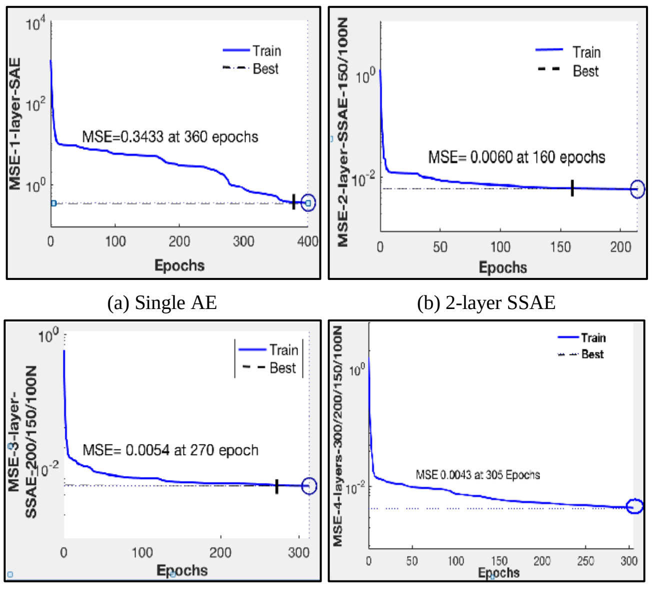


Fig. 9. Detection sub-system Level 2 classification, effectiveness compared to baseline

ii) Localization: One of the main concerns handled in the framework is to localize impending faults and predict their severity levels. While preprocessing selected the 353 features, further condensation was left to the stacked autoencoders used.

The comparative reconstruction performance is given in Figures 10(a) through 10(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 [21].

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.



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

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

Fig. 10. Mean Square Error for reconstruction of Input

The graph in Figure 11 shows that the model has good generalization characteristics as MSE for the test dataset is close to that of the training dataset. Having achieved good reconstruction results, the model was tested for prediction of severity of impending fault and performance issues. Having achieved good reconstruction results with stacked autoencoders, the model was tested for prediction of severity of impending fault and performance issues.

To see the effect of fine-tuning with backpropagation, experiments were carried out for SR=1 and SP=0.4. Figure 12 shows that fine-tuning may yield better results for some configurations of the model. The accuracy ranges between 72 and 85% with the abridged dataset (~1000 examples) and ~92% with the enhanced dataset (~5000 examples).

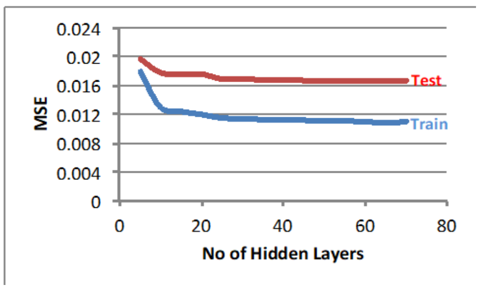


Fig. 11. MSE in Training and Test dataset

We baselined the above results with those obtained with a shallow model, viz., SVM with RBF kernel which did quite well for detection, and could only obtain 70% accuracy in localizing impending faults. The deep structure, thus provided a substantial improvement in terms of accuracy of prediction of the severity level of the impending faults.

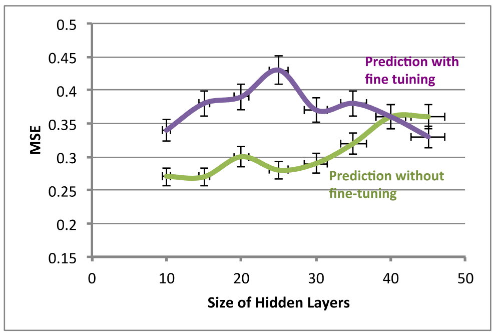


Fig. 12. MSE of predictions before and after fine tuning

VI. Discussion and Conclusion

In this work, we have identified the challenges faced by carriers in creating virtual network services based on NFV and instantiated over multiple clouds. The primary concerns addressed are availability and performance of virtual network services. To overcome these problems, we propose HYPER-VINES as a fault detection and localization framework to help cloud service providers, carriers, and independent NFVI providers to overcome these challenges. We demonstrated the effectiveness and feasibility of the hybrid framework using shallow machine learning and deep learning algorithms with a mix of unsupervised and supervised learning in a multi-layer configuration. We show that the proposed framework can handle both the detection and localization of faults and performance issues with good accuracy. Lastly, we believe that HYPER-VINES would be useful to the industry by improving the proliferation of NFV and cloud-based deployments and also spur other researchers to further develop and improve the framework.

Acknowledgment

This publication was made possible by NPRP grant #8-634-1-131 from the Qatar National Research Fund (a member of Qatar Foundation) and NSF grants CNS-1718929. The statements made herein are solely the responsibility of the authors.

References

[1] R. Mijumbi et al., " Network Function Virtualization: state-of-the-art and research challenges," IEEE Communications Surveys and Tutorials, 2016, pp.236-262.

[2] C. J. Bernardos et al., " Network Virtualization research challenges," Internet Engineering Task Force (IETF) Draft. https://tools.ietf.org/html/draft-irtf-nfvrg-gaps-network-virtualization-10 Accessed Sept 2, 2019

[3] H. R. Kouchaksaraei, and H. Karl, "Joint orchestration of cloud-based microservices and Virtual Network Functions. ArXiv: 1801.09984 https://arxiv.org/pdf/1801.09984. Accessed Feb 13, 2018

[4] ETSI GS NFV-MAN 001. "European Telecommunications Standards Institute: Network Functions Virtualization (NFV); Management and Orchestration," 2017

[5] P. Moore, "The current state of NFV: Standards," 2016 https://www.itential.com/blog/the-current-state-of-nfv-intro/. Accessed Feb 21, 2018.

[6] T.Nakamura, "Network Functions Virtualization (NFV) Network Operator Perspectives on NFV priorities for 5G," ETSI-White Paper, February 2017

[7] OPNVF, "Building fault management into NFV deployments background and purpose of OPNFV's Doctor Project," https://www.opnfv.org/wpcontent/uploads/sites/12/2016/11/opnfv_faultmgt_final.pdf. Accessed January 20, 2018

[8] M. Ladki, "Developing a blueprint for zero-touch, end-to-end service orchestration across hybrid and multiple networks," April 2017, https://inform.tmforum.org/features-and analysis/2017/04/developing-blueprint-zero-touch-end-end-service-orchestration-across-hybrid-multiple-networks/. Accessed March 21, 2018.

[9] ETSI GS NFV-REL001, "European Telecommunications Standards Institute: Network Functions Virtualization (NFV); Resiliency Requirements," 2015

[10] ETSI GS NFV INF 010, "European Telecommunications Standards Institute: Network Functions Virtualization (NFV); Service Quality Metrics," 2017

[11] ETSI GS IFA 013, "European Telecommunications Standards Institute: Network Functions Virtualization (NFV); Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification," 2016.

[12] ONAP Whitepaper, "Open Network Automation Platform: Architecture Overview," 2017, https://www.onap.org/wpcontent/uploads/sites/20/2017/12/ONAP_CaseSolution_Architecture_120817_FNL.pdf, Accessed March 20, 2018

[13] 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.

[14] 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.

[15] 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.

[16] B. Trammell et al., "mPlaneBuilding an intelligent measurement plane for the internet," IEEE Communications Magazine, Volume: 52, Issue: 5, 2014, pp. 148-156.

[17] M. L.Antonie, O. R.Zaiane, and R. C. Holte, "Learning to use a learned model: a two-stage approach to classification," IEEE International Conference on Data Mining, 2016, pp.33-42.

[18] F.Hachmi and M.Limam, "A two-stage process based on data mining and optimization to identify false positives and false negatives generated by Intrusion Detection Systems," IEEE International Conference on Computational Intelligence and Security, 2015, pp. 308-311.

[19] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, Vol 313, Issue 5786, July 2006, pp. 504-507.

[20] L. Wang et al., "A computational-based method for predicting drug-target interactions by using stacked autoencoder deep neural network," Journal of Computational Biology, 2018, pp. 361-373.

[21] W. Huang, " Dynamic boosting in deep learning using reconstruction error," IEEE International Joint Conference on Neural Networks (IJCNN'14)., 2014, pp. 473-480.

[22] I. Goodfellow, Y. Bengio, and A. Courville. 2016. "Deep Learning (1st. ed.)", MIT Press book.

[23] Kaggle datasets, Available: https://www.kaggle.com/datasets. Accessed, March 12 2017

[24] G. Holmes, A. Donkin, and I. H. Witten, "WEKA: A machine learning workbench," Proc ANZIIS of the Australian and New Zealand Conference on Intelligent Information Systems, 1994, pp. 357-361.

[25] Saed Sayad, "Classification Methods," http://chemeng.utoronto.ca/~datamining/Presentations/Basic_Methods.pdf, Accessed March 2018