Cognitive Assurance Architecture for Optical Network Fault Management

Danish Rafique, Thomas Szyrkowiec, Helmut Grießer, Achim Autenrieth, and Jörg-Peter Elbers

(Highly-Scored Paper)

***Abstract*—In face of staggering traffic growth driven by cloud-based platforms, modern optical networks—forming the backbone of such connectivity—are faced with increasing requirements in terms of operational reliability. The challenge is that of cognition-driven learning and fault management workflows, cost-effectively assuring the next-generation networks. Machine learning, an artificial intelligence tool, can be conceived as an extremely promising instrument to address network assurance via dynamic data-driven operation, as opposed to static pre-engineered solutions. In this paper, we propose and demonstrate a cognitive fault detection architecture for intelligent network assurance. We introduce the concept of cognitive fault management, elaborate on its integration in transport software defined network controller, and demonstrate its operation based on real-world fault examples. Our framework both detects and identifies significant faults, and outperforms conventional fixed threshold-triggered operations, both in terms of detection accuracy and proactive reaction time.**

***Index Terms*—Fault detection, machine learning, optical communication equipment, optical fiber communication, predictive maintenance, software defined networking.**

I. Introduction

NETWORK assurance is an integral part of optical network operations. It plays a key role in maintaining performance and quality levels, satisfying safety and reliability requirements, meeting regulatory and industry standards, and observing overall service level agreements (SLA) amongst different stakeholders. Traditionally, optical networks have been maintained using a fail-safe approach based on guaranteed redundancies and conservative design solutions. However, with rapid adoption of internet services, together with wide-scale availability of cloud platforms, and consequent extreme data rate and reliability requirements [1], [4]; traditional procedures are not only prohibitive from a cost viewpoint, but also from a scalability perspective. With increasing network heterogeneity, scalability and dynamicity, the task of network assurance necessitates novel architectures incorporating automated, flexible, resource-efficient, and reliable – potentially open – network management solutions.

In this context, software-defined networking (SDN) proposes centralized network management to enhance network programmability by separation of data and control plane [5]–[8]. Likewise, network monitoring and analysis approaches have been reported using both distributed and centralized frameworks [9], [10]. However, most of these networks operate based on static pre-defined configurations which may be engineered, planned, installed, configured, and maintained by human experts. Typically, pre-determined threshold-based triggers for configuration, restoration, planning, etc., are in place, often leading to underutilization of network resources. Furthermore, service and network behavior can evolve in an unpredictable manner, and catering such behavior using fixed thresholds not only exposes the network to unexpected failures, but is also a non-scalable approach in context of increasing optical network complexity.

Machine learning (ML), a branch of artificial intelligence, shows promise to address aforementioned problems. It has been applied to a wide range of applications, ranging from image processing to online recommendation system, aiming to learn to perform a certain task based on prior experience [11], [12]. Several machine learning algorithms exists—targeting distinct system behavior—mainly divided into supervised (labeled data) and unsupervised (unlabeled data) methods. These algorithms differ in terms of their structure and functionalities, and include clustering, Bayesian analysis, neural networks, deep learning, and dimensionality reduction algorithms, to name a few [13]–[16]. Typically, these algorithms are well-suited for systems with complex interdependencies, together with abundance of monitored data – a common trait of meshed optical networks. However, ML solutions necessitate domain-specific software architectures, consequent integration, and execution frameworks. Fig. 1 introduces a generic concept of an autonomous operational assurance approach, where a ML-driven management system would be capable of analyzing monitored data from different network sources, apply pre-trained ML models to it, and further using the given dataset to learn network behavior, before making action recommendations. Recently service configuration [17], [18], capacity optimization [19], planning and quality of transmission prediction [20]–[22] have been proposed in optical networking literature, together with bit error rate

Manuscript received October 23, 2017; revised November 27, 2017; accepted December 6, 2017. Date of publication December 7, 2017; date of current version March 1, 2018. This work was performed in the framework of the CELTIC EUREKA project SENDATE-Secure-DCI under Project C2015/3-4, and was supported in part by the German BMBF under Project 16KIS0477K and in part by the EU project METRO-HAUL under Agreement G. A. 761727. *(Corresponding author: Danish Rafique.)*

The authors are with ADVA Optical Networking, Munich 82152, Germany (e-mail: drafique@advaoptical.com; tszyrkowiec@advaoptical.com; hgriesser@advaoptical.com; aautenrieth@advaoptical.com; jelbers@advaoptical.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JLT.2017.2781540

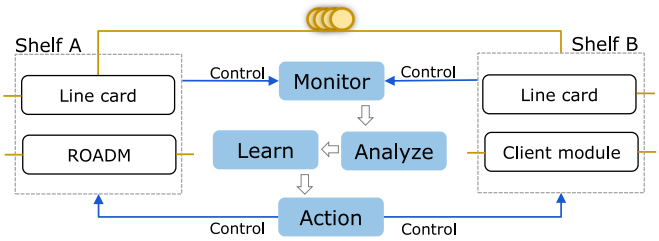

Fig. 1. Cognitive network assurance workflow.

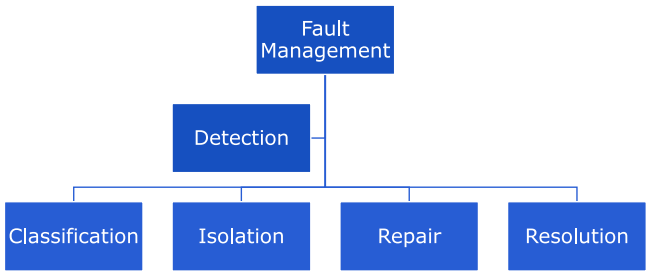

Fig. 2. Fault management tasks.

(BER) prediction using neural networks [9], however, network assurance targeting automated fault detection—involving monitoring, analysis and learning components of Fig. 1—remains largely an unexplored area.

In this paper, we propose a transport SDN-integrated (TSDN) cognitive assurance architecture [23], employing machine learning based autonomous fault detection. In particular, we disclose various types of real-life network fault use cases, extracted from a sample network, identify them using the proactive fault detection (PFD) framework, and compare its efficiency with traditional condition-based fixed threshold detection. The article is organized as follows: Section II introduces fault management hierarchy and relevant stages, Section III details network architectures incorporating cognitive engines, both globally and locally (TSDN). Section IV discusses machine learning algorithms, and software workflows used. Section V explains the results, comparing the cognitive and conventional frameworks, and finally Section VI draws conclusions.

II. Fault Management and Cognitive Networking

Fault management, a key constituent of network assurance, is a common element of asset-heavy industries like telecommunications, manufacturing, oil & gas, etc. For instance, in a typical optical networking scenario, a network segment may not only comprise of varied infrastructure like fiber types, transmission distances, geographical locations, etc., but also multitude of equipment types—potentially vendor agnostic—, running diverse software releases, addressing connectivity ranging from physical layer circuits to Ethernet services, and so on. Managing faults across such a diverse and complex mesh is a complicated task. Fault management allows for efficient control, operation, maintenance and preservation of such extensive and expensive infrastructure.

Fig. 2 shows typical fault management tasks, including,

– *Detection*
  Identification of a fault either reactively or proactively.
  Typically achieved using design-threshold based alarms.
– *Classification*
  Categorizing of malfunctions or failure, clustering similar faults together.
  Typically achieved using static data filtering methods.
– *Isolation*
  It is common for a single fault to trigger a chain of events. Fault isolation stage localizes the origin of failure.
  Typically addressed by second-level support departments.

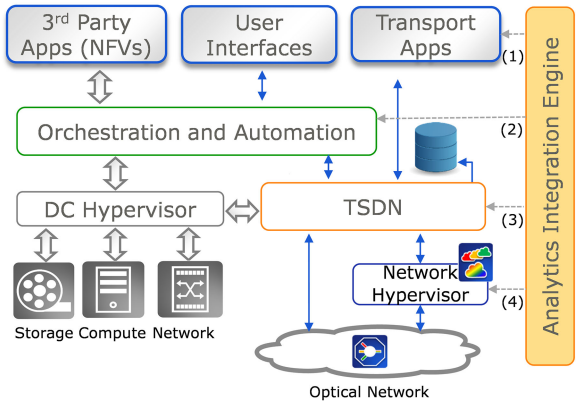

Fig. 3. Global network control architecture. DC: Data center, TSDN: Transport software defined network, NFV: Network function virtualization. Numbers represent deployment option of AIE.

– *Repair*
  Taking corrective measures to address the fault at hand, either using predefined workflows or data-driven methods.
  Typically addressed by domain experts, together with service engineers.
– *Resolution*
  Resolution concerns with restoration solution deployment, verification, and in-operation feedbacks.
  Typically addressed by service engineers.

Each of the aforementioned steps necessitates unique operational procedures like process workflows, periodic planned maintenance, etc. However, with the complexity and dynamicity of current infrastructures, traditional approaches show limited scalability—eventually leading to higher down times and related operational expenditures. While plethora of data is generated and consumed at each operational stage, and relevant networking layer, current architectures typically lack a coherent data-driven approach to gain relevant insights, and move from largely reactive to proactive fault management frameworks.

In this context, advanced machine learning based analytics solutions have the potential to exploit the immense amount of operational data, find hidden relationships among several entities, and eventually improve infrastructures' assurance. Fig. 3 shows a high-level network architecture, extending the, now state-of-the-art, orchestrator and SDN based control and management framework with an analytics integration engine (AIE).

The underlying network consists of intra- and inter- data center infrastructure. The intra-data center resources comprise storage, compute and network, whereas inter-data center connectivity is provided by a transport network, controlled by a TSDN controller. On top of that sits the orchestrator. Its main task is to coordinate and automate the data center workflows. Finally, a variety of applications may access the framework.

The AIE has several deployment options, with their respective advantages and disadvantages. (1) Transport Application: Deploying AIE as a transport App allows for vendor agnostic and portable analytics solutions, however with challenges related to synchronization of results and contention of actions in case of several apps addressing multiple network segments. (2) Orchestration Integration: This architecture allows for a bird's eye view on all the network resources, however it is completely centralized necessitating monitoring data access and processing in a single location. (3) TSDN Integration: Here, AIE has unique advantage of being distributed and centralized at the same time, allowing for resource efficient operation, however suffers from reduced interoperability, unless SDN controllers operate on fully open source models and APIs. (4) Hypervisor Integration: This deployment option is similar to that of TSDN, with higher level of distribution granularity, however it promotes vendor-specific analytics solutions. Finally, 3rd party applications invoking network function virtualization may also deploy analytics-as-a-service, however this will be limited to packet flow based monitoring (OpenFlow, etc.), without full access to optical data streams – available through the SDN controller.

In terms of functionalities, the AIE abstracts monitored data via suitable interfaces, implements a ML-based learning engine, using a historical database, and an exposure block to map suitable fault management actions, as discussed previously. The extent of assurance depends upon the target objective of the AIE. It may simply map results in aid of a fault management task, or it may be extended to enable a unified fault management solution with eventual actions, ranging from fault detection all the way to resolution. Moreover, several aspects need to be considered to enable the AIE, including data source access and unification, domain mapping to ML algorithms, back-end integration, application-specific protocols and so on.

III. Fault Detection and TSDN Architecture

In this article we focus on fault detection aspects of network assurance using ML methods. Fig. 4 depicts the concept of three fault modes, considering equipment or traffics' operational reliability. Physical degradation modes refer to typical network lifetime deteriorations including system aging, connector reuse, etc. Equipment dynamic modes refer to sudden changes in subsystem behavior, e.g., power level spikes, increased spectral tilt due to EDFA [24], etc. Failure modes represent subsystem or system failure, e.g., too high BER, resources not available, etc.

Clearly, optical networks need to operate within physical and equipment dynamic modes, taking appropriate actions before ever transitioning into failure modes. These failure modes can further be narrowed down to specific fault types observed in commercial networks, as shown in Table I. It is worth mentioning that the reported fault categories are feature agnostic, and applicable to multiple layers in the communication stack. Network faults can take many forms including a spike due to a short-lived event, a gradual change in behavior, a state change due to certain configuration changes, and finally localized abnormalities indicating potential faults.

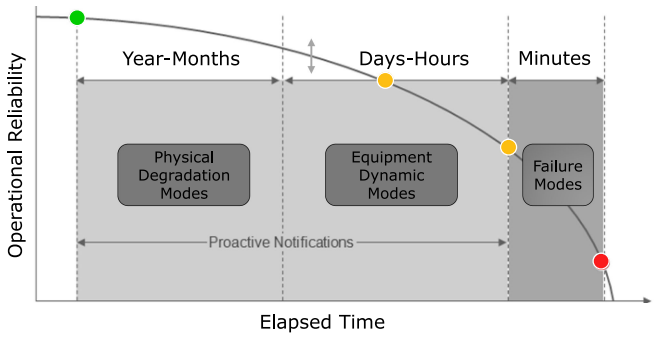

Fig. 4. Equipment or traffic operational reliability highlighting three failure modes. Green: available, Yellow: warning, and Red: failure.

TABLE I
Fault Types Typically Encountered in Commercial Optical Networking Systems

Fault Label	Description
I	Point abnormalities due to random flash events and may lead to abrupt device damage
II	Local abnormalities indicating potential flaws with potential long-term impact on service performance
III	Steady abnormalities due to preceding system configuration changes, and may lead to damage and/or consistent performance loss
IV	Ramp abnormalities representing gradual system and/or service distortion possibilities

It is customary of optical networking management tools to detect such faults by accessing fixed thresholds set by system engineers at various system levels; used in-operation to alert malfunctions. This approach, however, leads to underutilization of system resources and conservative operational choices. Furthermore, tools lack data correlation techniques to determine the root cause of faults across different network platforms so fault restoration is often less than optimal. Subsequently system downtime, and ensuing delays in other fault management stages leads to prohibitive costs. Here, we detail an architecture which aims to proactively detect potential failures in real-time, replacing the static fault thresholds based reactive approaches.

Fig. 5 expands on one of the manifestations of the AIE engine, presented in Fig. 3 (option (3)), and depicts the software architecture for our approach, where the PFD framework is located within the TSDN controller. The monitoring data is collected every 15 minutes through the southbound interface (SBI) using NETCONF. Subsequently, the data is abstracted and stored in

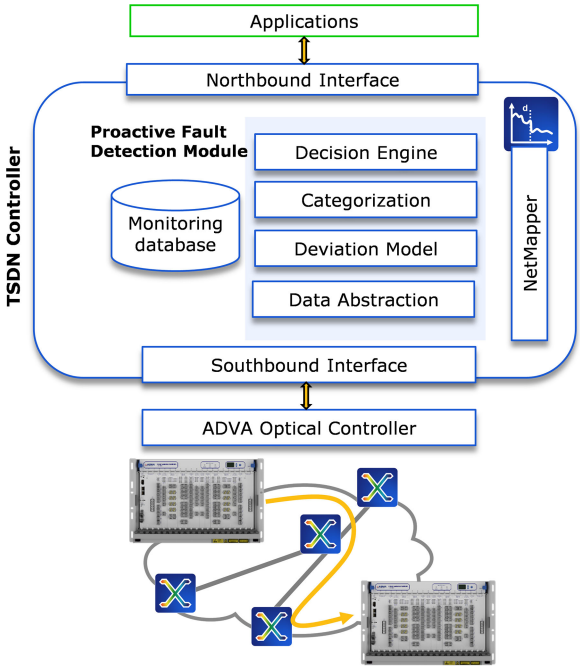

Fig. 5. TSDN controller with proactive fault detection architecture, representing TSDN deployment of AIE. Underlying network and service (yellow) is exemplary. PFD: Proactive fault detection.

a database. The engine performs fault detection, generates fault layer information (by mapping the metadata), fault locations, and maps the machine learning outcome to an internal decision engine followed by respective application (not specified here), exposed via RESTCONF to the northbound interface (NBI).

The optical controller is an ADVA controller, whereas hardware components – in the given example –, comprise of ADVA network elements [25]. Note that we consider a TSDN-integrated approach for our proposed framework, however, the proposed architecture may be disintegrated, decentralized or used as part of an orchestrator as well, as discussed in Section II. Furthermore while we test the proposed approach on features monitored at layer 0, the proposed cognitive architecture is scalable, and may be reused with, for example, packet flows (using OpenFlow [26], etc.).

IV. Algorithms and Implementations

Table II details the cognitive fault detection approach. The goal of this procedure is to retrieve and process data blocks (partitioned data), make abnormality decisions, prune the points which are insignificant, send the results to the next cycle, and eventually decide on true positives. The algorithm takes as input the monitoring data from the network, and outputs labels for normal and abnormal operation. The details of the steps are as follows: The core of the PFD framework traverses through the monitored data, and applies block based deviation and categorization tests for a given set, removing abnormal points, and repeating the tests again. Once all potential faults are identified, the decision engine predicts true abnormal behavior using a neural network based classifier, trained using historically observed fault patterns. Finally, the decisions are either exposed to NetMapper(), an interface to conventional SDN functionalities like path computation, configuration, etc., or directly exposed to applications.

TABLE II
Algorithm for Cognitive Fault Detection Framework, Processing Monitored Data as an Input and Generating Proactive True Fault Information as Output

INPUT: $\boldsymbol{X}_t^w \leftarrow metrics\ (\boldsymbol{X}_1^w{:}\boldsymbol{X}_t^w)$,
$\mathcal{L} = \langle \ell_1, \ell_2, \ldots \ell_m \rangle \in \mathbb{R}$
where $\boldsymbol{X}_t^w{:}\Leftrightarrow\{\boldsymbol{X}_t^w{:}w, t \in \mathbb{Z}^+\} \in \mathbb{R}$
OUTPUT: $\mathbf{R}{:}\Leftrightarrow\{r_k{:}k \in \mathbb{N}\}$

1. *for all blocks in* $\boldsymbol{X}_t^w$ *do*
2. $K \leftarrow getBlock\ (\boldsymbol{X}_t^w)$
3. $\forall\ k \in K\ do\ \rightarrow\ c_k \equiv \emptyset$
4. *for all point* $k \in \boldsymbol{X}_t^w$
5. $D_k \equiv DeviationCalc()$
6. $c_k \equiv CategorizationTest()$
7. $c_K \leftarrow (c_1{:}c_k)$
8. $\boldsymbol{C} \leftarrow (c_1{:}c_K)$
9. $R \equiv getDecision(\mathcal{L}, C)$
10. $f{:}\boldsymbol{R} \rightarrow \boldsymbol{Y}[Configuration, Alarm, Service]$ *defined as*
11. $NetMapper(\boldsymbol{R})$

In case of abnormality detection based on deviation and categorization, we use generalized extreme studentized deviate (ESD) test, which is a sequential processing based modification of Grubb's test [27], [28]. ESD is used to detect one or more faults in the given data, given as follows,

$$D_k = \left[\max_k |x_k - \bar{x}|\,; x_k \in X_w^t\right] / sd \tag{1}$$

where k is an index of the data points in each block, $\bar{x}$ and sd represent mean and standard deviation.

$$c_k = \frac{t_{N-k-1,p(N-k)}}{\sqrt{\left(N-k-1+t_{N-k-1,p}^2\right)(N-k+1)}} \tag{2}$$

where N is the block length, $t_{x,y}$ represent t-distribution, $p = 1 - \alpha/2(k-i+1)$, and α is confidence interval (typical values of 0.05). The abnormalities are categorized by finding largest k such that D_k is greater than c_k.

Once initial fault indicators are determined, the data is further tested for true abnormality using a neural network based classifier. It is worth mentioning that neural networks are computation-intensive, and are typically used with several hundreds of neurons and multiple hidden layers with very high complexity [29]. However, our novel approach of using ESD test before the neural network, which takes the load off these networks, results in a very simple architecture as used in our work. In particular, we employ a three layered neural network architecture, as shown in Fig. 6. The network has seven inputs, representing positive and negative fault label values, including the indicated fault generated from deviation and categorization tests. The inputs represent a pattern including the abnormal label identified by the first test and its neighboring data labels, be it normal or abnormal, in a sliding window. The role of neural network is to make true abnormality decision based on these

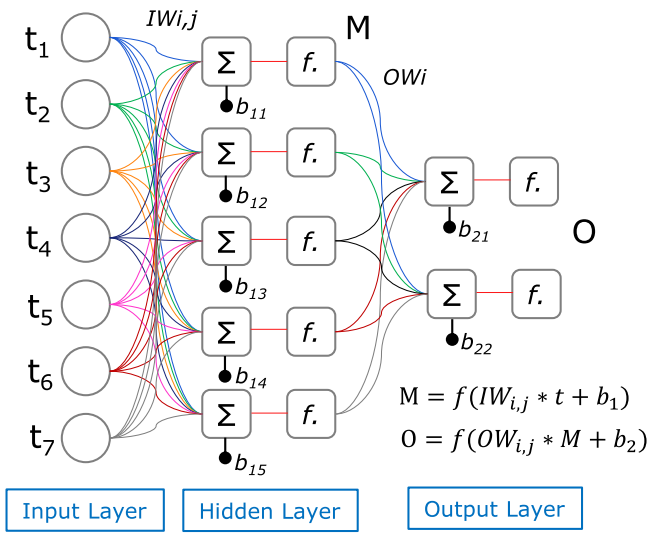

Fig. 6. Neural network architecture. t inputs, $IW_{i,j}$ are the input weights, OW_i are the output weights, b are the bias values, and f is the activation function.

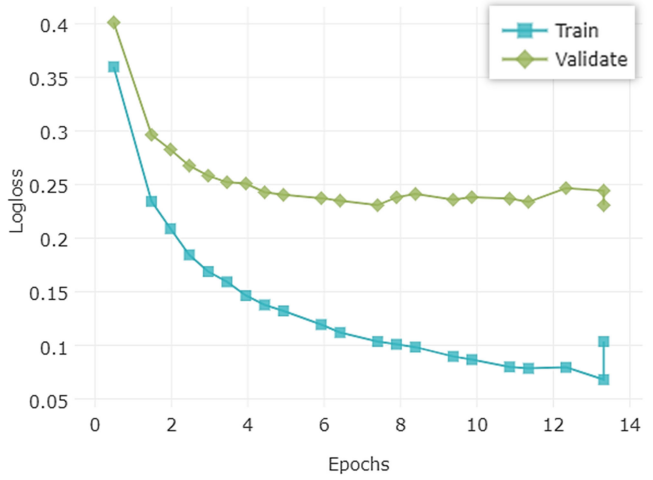

Fig. 7. Optimization curves for neural network shown in Fig. 6. Squares: Training, Diamonds: Validation.

fault label patterns. The hidden layer contains five neurons. The output suggests true normal or true abnormal behavior of a potential identified fault. Initially the weights and biases are randomly initialized, and backpropagation is used to train the network [30]. Note that the true abnormal behavior is identified based on service logs, support engineers, etc. This is a highly system dependent feature, and in reality this is pre-identified either conservatively or through experience of support engineers. With neural networks, we aim to learn and generalize this feature.

Fig. 7 shows training and validation phase of the proposed neural network, where log-loss is used as an optimization function as a function of epochs (where one epoch represents a single pass of input data through the learning model). Log-loss represents negative log-likelihood function of true fault given a

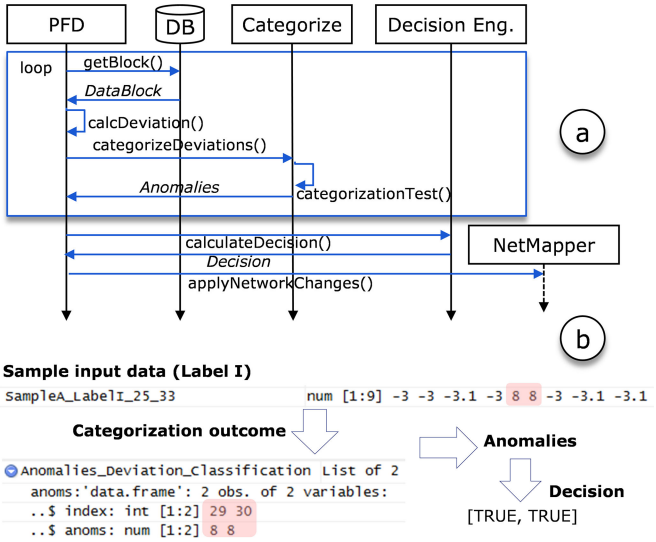

Fig. 8. a) Sequence flow, b) Implementation snapshot (Label I sample).

probabilistic classifier's predictions. For a binary classifier, as used in our work, the formulation is given as [31],

$$\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i + (1-y_i)\log(1-p_i)$$

where N is number of inputs, y is a binary indicator of whether or not a fault is correctly classified for instance i, and p is the model probability. A perfect classifier would have a log-loss of precisely zero. Less ideal classifiers have progressively larger values of log-loss. The available fault patterns are divided into training and validation sets at 80:20 ratio. This is done to avoid overfitting of the models to the training set, as observed in Fig. 7. It can be seen that the training error reduces as a function of epochs, whereas the validation error saturates, and even trends slightly upwards. The optimum model is the one which provides minimum validation error.

Fig. 8 shows the implemented workflow, together with an implementation trace. The workflow is as follows. The data blocks are retrieved sequentially and the deviation is calculated. The result is send to the classifier, which applies a test to received data and responds with the anomalies. Taking into account all anomalies, the decision engine determines true faults, and applies relevant network changes through network mapper.

Fig. 8(b) shows the implementation snapshot for one data samples' input and output features. The input (index 25:33) corresponds to fault label I, which is a point fault. After first level statistical tests, two anomalies are reported to the decision engine, which determines that, in the given sample, both detected anomalies represent true faults.

V. Configurations and Discussions

The experiment was carried out using diverse patterns extracted from ADVA's sample network, where various patterns were simulated, as discussed in Table I. The labels represent normal and abnormal behavioral states which may or may not lead

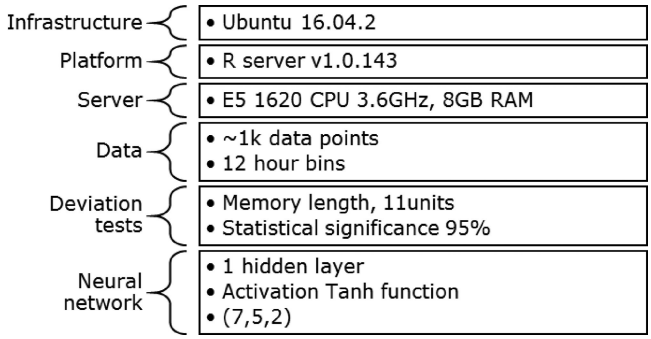

Fig. 9. Platforms used in this work, together with configurations details.

to potential equipment or service failure modes, as discussed earlier.

Fig. 9 enlists some of the key system configurations used in this work, related to compute environment and software details, and related machine learning algorithms. Note that these configurations are rather modest, and better platforms, together with increased algorithm complexity, may be used in upcoming studies.

As discussed in Section III, the physical layer received optical power levels were pulled in real-time via the optical controller, and the proposed architecture was executed on a host server. The monitored data was normalized to same scale (−1 dBm) for better visualization, and this action did not have any impact on the performance. We evaluated two scenarios: one employing the PFD engine, where the true potential failure scenarios were cognitively predicted, and another where the PFD was replaced by a fixed pre-determined fault determination threshold, representative of commercial network operational practice. Also, conventional approach lacks the capability to make a decision on true fault behavior. Since we normalized our data, a single threshold was used for comparison, whereas in practice several threshold are required to be defined by system engineers, and maintained for various network configurations.

Furthermore, first-order derivatives may be used to raise alarms for a few dB of variations, the key point, however, is that most of these alarms are in fact false positives, as first order derivatives don't intrinsically incorporate failure knowledge. Consequently, several fixed thresholds (service based, device based, etc.) are used to reduce the false positive rate. Nonetheless, precisely these issues are catered by our approach, where true fault labels are cognitively identified based on data patterns – simplifying operational complexity of setting multiple thresholds, and improving reaction times.

Fig. 10 shows various fault patterns, as described in Table I, for different time traces. Fig. 10(a) depicts the performance of threshold-triggered fault detection, where label I is fully detected, labels' III and IV are only partially detected, whereas label II remains entirely undetected. In comparison, Fig. 10(b) illustrates results from the proposed cognitive architecture, where all labels are largely detected, except for three and one potential faults in labels II and IV, respectively. It is worth mentioning that while the detection rate of faults in itself is an important system metric, the fundamental operational requirement is the ability to proactively react to such potential failures in time.

TABLE III
ALGORITHM FOR COGNITIVE FAULT DETECTION FRAMEWORK, PROCESSING MONITORED DATA AS AN INPUT AND GENERATING PROACTIVE FAULT

		Abnormality Detection Rate [%]	Proactive Reaction Time [hours]	Detected / True Faults
Condition-based	Label I	100	0	1
Data-driven	Label I	100	0	1
Condition-based	Label II	0	0	0
Data-driven	Label II	~57	>100*	0.57
Condition-based	Label III	~45	0	0.45
Data-driven	Label III	100	10	1
Condition-based	Label IV	25	0	.25
Data-driven	Label IV	~93	96	1.25

From Table III it can be observed that our architecture outperforms conventional set-point based detection, both in terms of detection rate and associated response time (defined as time from detection to typical failure state) – for any given fault label. The first column represents performance of ESD in terms of abnormal behavior detection, whereas the last column shows the performance of neural network on the detected abnormal behavior in terms of true fault label detection. Specifically, label IV may be adequately addressed before actual failure, with a response time of 96 hrs, as opposed to no response time at all from threshold-based detection. Likewise, label III alarms may be issued at least 10 hrs prior to failure, whereas reaction time to label II depends upon its evolution behavior. While label I is detected by both methods, it allows no reaction time due to its peak response. Finally, the trade-off between accuracy of abnormality detection and true predicted faults is shown in the last column, where a ratio of 1 shows 100% detection of actual faults predictors (determined based on the neural network model). Qualitatively this means whether the decision engine incorporated in the PFD framework correctly labels significant and nonsignificant fault behaviors, based on historically identified fault patterns. It can be seen that while PFD performs exceedingly well, compared to condition-based method, label IV leads to minor over-detection or false positives due to incorrect ANN classification. The incorrect results may be attributed to model overfitting, and classification performance may be improved by using more complex neural network architectures, multi-dimensional hyper-parameter optimizations, and more training data.

Finally, the core idea to use neural networks for true label identification is to generalize and automate the operational complexity of setting multiple thresholds, which are specific to network type, service type, modules, transmission rates, etc. Once neural network model is trained, it takes a few seconds to apply it to new test cases, and consequently doesn't compete with the large available proactive reaction times.

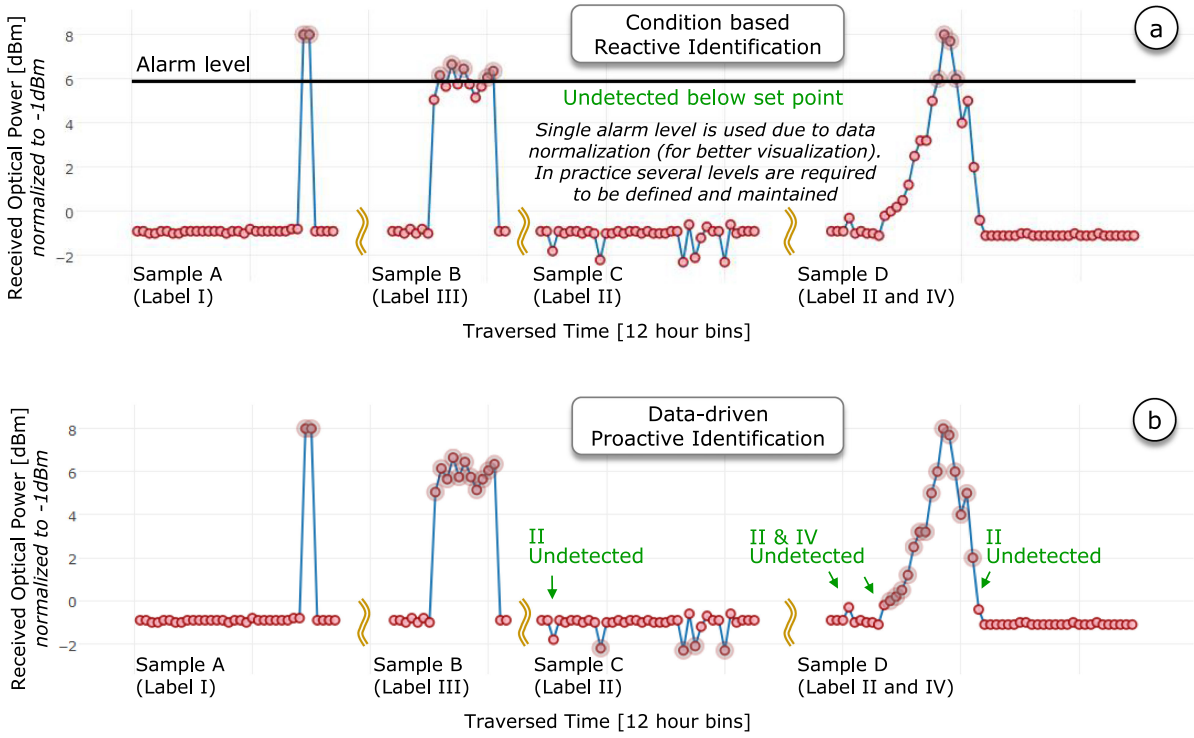

Fig. 10. Monitored layer 0 optical received power levels as a function of traversed time (aggregated to 12 hour bins). a) Condition based fault detection, b) Data based fault detection (PFD engine). Highlighted symbols (opacity) represent detected faults. Broken lines represent different data samples. For fault label definitions see Table I.

VI. Conclusion

We proposed and demonstrated a, TSDN-integrated, cognitive network assurance architecture for next-generation network management and operation. The proposed framework not only allows for simpler network management, getting rid of multiple fixed set point definition and maintenance; but also significantly improves the proactive fault response time, compared to conventional threshold-based failure detection.

While machine learning is gaining momentum in optical networking industry, most of the effort is directed towards capacity optimization and routing challenges. The goal of this work is to introduce a generic cognitive assurance architecture based on machine learning, neither limited to presented network configuration nor to the dataset, and application use cases across different network layers are currently underway.

References

[1] A. Gupta and R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, Jul. 2015.

[2] V. Vusirikala, "A decade of software defined networking at Google," in *Proc. Eur. Conf. Opt. Commun.*, Gothenburg, Sweden, Sep. 2017.

[3] A. Tzanakaki, M. Anastasopoulos, and D. Simeonidou, "Optical networking: An important enabler for 5G," in *Proc. Eur. Conf. Opt. Commun.*, Gothenburg, Sweden, 2017, Paper M.2.A.1.

[4] S. Aleksic, "Towards fifth-generation (5G) optical transport networks," in *Proc. 17th Int. Conf. Transparent Opt. Netw.*, Budapest, Hungary, 2015, Paper We.A2.2.

[5] M. Vissers, I. Busi, M. Betts, L. Ong, and G. Zhang, "SDN architecture for transport networks," *Open Netw. Forum*, Palo Alto, CA, USA, ONF TR-522, 2016. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/SDN_Architecture_for_Transport_Networks_TR522.pdf

[6] R. Casellas, "Control, management and orchestration of optical networks: An introduction, challenges and current trends," in *Proc. Eur. Conf. Opt. Commun.*, Gothenburg, Sweden, 2017, Paper Tu.1.G.1.

[7] V. Lopez *et al.*, "The role of SDN in application centric IP and optical networks," in *Proc. Eur. Conf. Netw. Commun.*, Athens, Greece, 2016, pp. 138–142.

[8] T. Szyrkowiec *et al.*, "First field demonstration of cloud datacenter workflow automation employing dynamic optical transport network resources under OpenStack & OpenFlow orchestration," in *Proc. 39th Eur. Conf. Exhib. Opt. Commun.*, London, U.K., 2013, pp. 1–3.

[9] S. Yan, A. Aguado, Y. Ou, R. Wang, R. Nejabati, and D. Simeonidou, "Multilayer network analytics with SDN-based monitoring framework," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 9, no. 2, pp. A271–A279, Feb. 2017.

[10] A. P. Vela, M. Ruiz, and L. Velasco, "Bringing data analytics to the network nodes for efficient traffic anomalies detection," in *Proc. 19th Int. Conf. Transparent Opt. Netw.*, Girona, Spain, 2017, pp. 1–4.

[11] A. H. Duffy, "The "what" and "how" of learning in design," *IEEE Expert*, vol. 12, no. 3, pp. 71–76, May/Jun. 1997.

[12] E. Alpaydm, *Introduction to Machine Learning*, 3rd ed. Cambridge, MA, USA: MIT, 2014.

[13] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, no. 2, pp. 4–22, Apr. 1987.

[14] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[15] I. T. Jolliffe, *Principal Component Analysis*. Berlin, Germany: Springer-Verlag, 2002.

[16] H. Zhang and Y. Lu, "Learning Bayesian network classifiers from data with missing values," in *Proc. IEEE Region 10 Conf. Comput., Commun., Control Power Eng.*, 2002, vol. 1, pp. 35–38.

[17] L. Gifre, M. Ruiz, A. Castro, R. Proietti, S. J. B. Yoo, and L. Velasco, "Experimental assessment of degradation-triggered reconfiguration in optically interconnected cloud-RAN," in *Proc. Opt. Fiber Commun. Conf. Exhib.*, Los Angeles, CA, USA, 2017, Paper W2A.30.

[18] X. Chen *et al.*, "Leveraging deep learning to achieve knowledge-based autonomous service provisioning in broker-based multi-domain SD-EONs with proactive and intelligent predictions of multi-domain traffic," in *Proc. Eur. Conf. Opt. Commun.*, Gothenburg, Sweden, 2017, Paper W.3.A.3.

[19] F. Morales, M. Ruiz, and L. Velasco, "Incremental capacity planning in optical transport networks based on periodic performance metrics," in *Proc. 18th Int. Conf. Transparent Opt. Netw.*, Trento, Italy, 2016, pp. 1–4.
[20] P. Samadi, D. Amar, C. Lepers, M. Lourdiane, and K. Bergman, "Quality of transmission prediction with machine learning for dynamic operation of optical WDM networks," in *Proc. Eur. Conf. Opt. Commun.*, Gothenburg, Sweden, 2017, Paper W.3.A.1.
[21] F. Meng *et al.*, "Field trial of a novel SDN enabled network restoration utilizing in-depth optical performance monitoring assisted network replanning," in *Proc. Opt. Fiber Commun. Conf. Exhib.*, Los Angeles, CA, USA, 2017, pp. 1–3.
[22] M. Bouda *et al.*, "Accurate prediction of quality of transmission with dynamically configurable optical impairment model," in *Proc. Opt. Fiber Commun. Conf., OSA Tech. Dig. (Opt. Soc. Amer.)*, 2017, Paper Th1J.4.
[23] D. Rafique, T. Szyrkowiec, H. Grießer, A. Autenrieth, and J.-P. Elber, "TSDN-enabled network assurance: A cognitive fault detection architecture," in *Proc. Eur. Conf. Opt. Commun.*, Gothenburg, Sweden, 2017, Paper W.3.A.4.
[24] Y. Huang *et al.*, "A machine learning approach for dynamic optical channel add/drop strategies that minimize EDFA power excursions," in *Proc. 42nd Eur. Conf. Opt. Commun.*, Dusseldorf, Germany, 2016, pp. 1–3.
[25] FSP3000 AgileConnect, ADVA Optical Networking. [Online]. Available: http://www.advaoptical.com/en/products/scalable-optical-transport/fsp-3000-agileconnect.aspx
[26] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[27] F. E. Grubbs, "Sample criteria for testing outlying observations," *Ann. Math. Statist.*, vol. 21, no. 1, pp. 27–58, 1950.
[28] B. Rosner, "Percentage points for a generalized ESD many-outlier procedure," *Technometrics*, vol. 25, no. 2, pp. 165–172, May 1983.
[29] P. Samadi *et al.*, "Quality of transmission prediction with machine learning for dynamic operation of optical WDM networks," in *Proc. Eur. Conf. Opt. Commun.*, 2017, Paper W3A1.
[30] D. E. Rumelhart, G. E. Hinton, and R. Williams, "Learning representations by back-propagating errors," *J. Nature,* vol. 323, pp. 533–536, 1986.
[31] B. Baldwin, "Evaluating with probabilistic truth: Log loss vs. O/1 loss," *LingPipe Blog*, 2010. [Online]. Available: https://lingpipe-blog.com/2010/11/02/evaluating-with-probabilistic-truth-log-loss-vs-0-1-loss/

Danish Rafique received the B.Sc., M.Sc., and Ph.D. degrees all in electrical engineering. He is a Senior Innovation Manager and Principal Engineer with the CTO Office, ADVA Optical Networking, where he is working on roadmap definitions for cognitive network design, control, and management and several EU and national projects. He was previously a Visiting Research Scientist with Acreo AB, Sweden (2008–2011), with Nokia Siemens Networks, Portugal as a System Architect (2012), followed by Senior R&D Eng. and Senior Lead Manager Product and Technology positions with Coriant, Germany (2013–2016).

Thomas Szyrkowiec received the M.Sc. degree in informatics from the Technical University of Munich (TUM), Munich, Germany, in 2013. He is currently working toward the Dr.-Ing. degree at the Institute for Communication Networks, TUM. He is currently with ADVA Optical Networking SE, Munich, as an Engineer with the CTO Office. His research interests include software-defined networking and network virtualization in optical networks.

Helmut Grießer received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering and information technology from Ulm University, Ulm, Germany, in 1996 and 2002, respectively. He is currently the Director Advanced Technology with the CTO Office, ADVA Optical Networking, where he is working on signal processing, coding, modulation formats, and system design for optical fiber communication systems. Before joining ADVA Optical Networking in 2011, he was with Ericsson (previously Marconi) working on high-speed fiber transmission research.

Achim Autenrieth received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering and information technology from the Munich University of Technology, Munich, Germany, in 1996 and 2003, respectively. He is currently the Director Advanced Technology with the CTO Office, ADVA Optical Networking, where he is working on multilayer network design, planning, SDN and NFV c for converged 5G access, metro, and core networks. Before joining ADVA Optical Networking in June 2010, he was with Siemens AG and Nokia Siemens Networks, where he was last working as the Head of Broadband Connectivity Solutions R&D Innovations.

Jörg-Peter Elbers received the Diploma and Dr.-Ing. degrees in electrical engineering from Dortmund University, Dortmund, Germany, in 1996 and 2000, respectively. He is a Senior Vice President Advanced Technology, Standards & IPR with ADVA Optical Networking, Munich, Germany, and responsible for technology strategy, new product concepts, standardization, and research. Prior to joining ADVA in 2007, he was the Director of technology with the Optical Product Unit of Marconi (now Ericsson). From 1999 to 2001, he was with Siemens AG, last as the Director of Network Architecture in Siemens Optical Networks.