Machine Learning for Network Automation: Overview, Architecture, and Applications [Invited Tutorial]

Danish Rafique and Luis Velasco

***Abstract*—Networks are complex interacting systems involving cloud operations, core and metro transport, and mobile connectivity all the way to video streaming and similar user applications. With localized and highly engineered operational tools, it is typical of these networks to take days to weeks for any changes, upgrades, or service deployments to take effect. Machine learning, a sub-domain of artificial intelligence, is highly suitable for complex system representation. In this tutorial paper, we review several machine learning concepts tailored to the optical networking industry and discuss algorithm choices, data and model management strategies, and integration into existing network control and management tools. We then describe four networking case studies in detail, covering predictive maintenance, virtual network topology management, capacity optimization, and optical spectral analysis.**

***Index Terms*—Analytics; Artificial intelligence; Autonomous networking; Big data; Communication networks; Machine learning; Optical fiber communication; Telemetry.**

I. Introduction

Optical communication networks may very well be regarded as the cornerstone of modern society. Internet-based applications and classical enterprise facilities both rely on a complex mesh of optical networking infrastructure to address connectivity requirements. In order to support such differentiated service offerings, optical networks have incorporated a series of innovations over recent decades, including the development of lasers, amplifiers, fibers, coherent detection, and digital signal processing, to name a few. With recent advances in mobile communication systems—5G networking [1], together with extensive service-oriented cloud platforms like Uber, Amazon, etc.—optical transport stakeholders face tremendous challenges in terms of harmonizing stagnating revenue streams and growing networking demands [2,3].

The industry has traditionally relied on hardware-centric innovations and continues to do so successfully—for instance, via photonic integration, graphene, space division multiplexing, etc. Figure 1 depicts a typical architecture of a multi-domain optical network, encompassing core, metro, and access networks. The various network segments typically need to work together to support multi-layer services and applications, either directly or through a hybrid wireless/wireline infrastructure. This diverse, dynamic, and complex mesh of networking stacks, together with future mobility constraints, necessitates smart and end-to-end service-oriented software frameworks to augment conventional hardware advancements. To this end, software-defined networking (SDN)-triggered control and orchestration has been introduced in the past few years, allowing for separation of control and data planes in various degrees of centralization [4,5]. Furthermore, network function virtualization (NFV) has been used in tandem to abstract physical device functionalities. The challenge, however, is that of augmenting network management and control tools with adaptive learning and decision-making across multi-layer and multi-domain network architectures in a cost- and energy- efficient manner, facilitating end-to-end network automation.

Artificial intelligence (AI) is the science of creating intelligent machines capable of autonomously making decisions based on their perceived environment [6]. Machine learning (ML), a branch of AI, enables this learning paradigm (see [7,8]). ML may be used to achieve network-domain goals ranging from laser characterization [9] to erbium-doped fiber amplifier (EDFA) equalization [10], predictive maintenance [11], and failure localization [12,13], as well as related capital expenditure (CAPEX) and operational expenditure (OPEX) savings. ML algorithms are characterized by a unique ability to learn system behavior from past data and estimate future responses based on the learned system model. For a comprehensive survey of AI methods in optical networks, we refer the reader to [14,15].

With recent improvements in computational hardware and parallel computing, such as the commercialization of big data monitoring, storage, and processing frameworks, maturity of ML algorithms, and introduction of SDN/NFV platforms, several optical networking challenges may be partially or fully addressed using ML paradigms. The key motivations together with underlying application scenarios are listed below:

– *Heterogeneity*: Optical networks are a diverse and dynamic medium. Service allocations, transmission performance, optimum configurations, etc. continuously evolve over

Manuscript received March 28, 2018; revised June 30, 2018; accepted June 30, 2018; published August 23, 2018 (Doc. ID 327126).

D. Rafique (e-mail: DRafique@ADVAoptical.com) is with ADVA Optical Networking SE, Fraunhoeferstr. 9a, 82152 Munich, Germany.

L. Velasco is with Universitat Politecnica de Catalunya, 08034 Barcelona, Spain.

https://doi.org/10.1364/JOCN.10.00D126

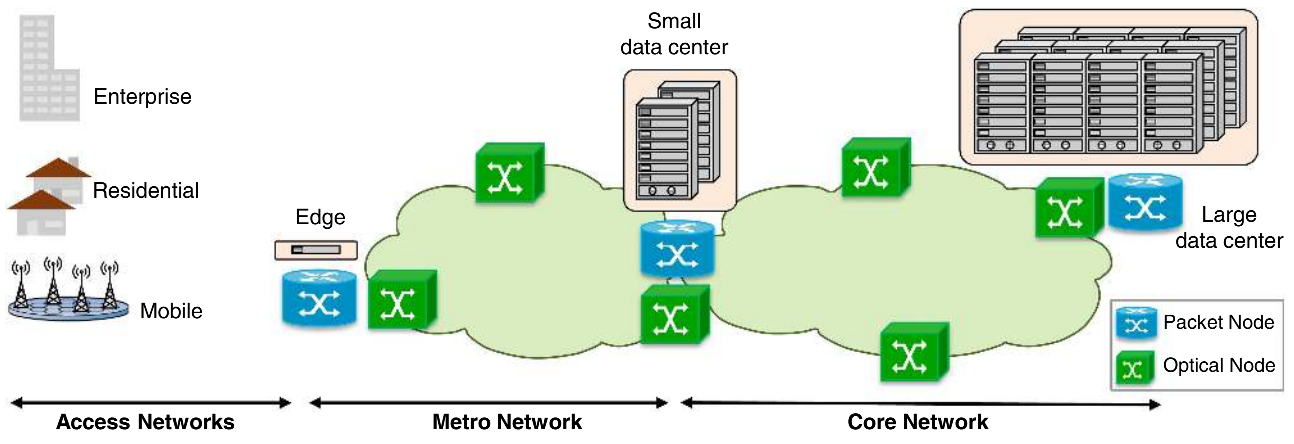


Fig. 1. Heterogeneous optical network architecture.

time. While traditionally these were handled based on static design principles, this approach no longer scales owing to differentiated and often contrasting service and operational requirements.

– *Reliability*: Optical communication infrastructure is typically built to last. This is achieved via fail-safe designs, incorporating various engineering and operational margins, etc. While this approach worked well for traditional network design and operation, the scale, complexity, and sheer combinations of equipment types, part numbers, software releases, etc., especially in the context of open line systems (OLS), render this approach practically infeasible.
– *Capacity*: Link and network capacity throughout optimization is a typical metric of network design. In fact, it is one of the most important features in terms of a solution's commercial viability. A typical differentiator in service-provider-issued requests for proposals is if a certain configuration can achieve a particular optical reach with high spectral efficiency. Conventionally, precise engineering rules are devised in order to fulfill such tasks. However, the number of configurations supported by the upcoming generation of optical transceivers does not play well with this approach.
– *Complexity*: Optical networks are often built in complicated meshed architectures. In several scenarios, optimum principles are either difficult to model (e.g., network planning), or it is impossible to come up with closed-form analytical models or fast heuristics. This leads to severe under-utilization of system resources, resulting in both OPEX and CAPEX overheads.
– *Quality Assurance*: With growing complexity, the task of network testing and verification is becoming increasingly prohibitive. Network operators are neither comfortable nor ready to deploy live traffic on untested configurations, and the concepts of self-optimization are highly suited for such problems.
– *Data Aspects*: An optical network is a sensor pool in itself, with data ranging from service configurations to maintenance logs. Most of this treasure is largely untapped in commercial systems. Discovering known and unknown patterns in a network allowing intelligent and autonomous operations can lead to a wealth of optimization possibilities. Here, ML can help to abstract various functionalities, enabling data-driven tasks with limited manual interventions and/or interruptions.

Despite the promise and scale of ML paradigms, the goal of learning-based optimization— considering the extent of services, infrastructure, and operational requirements—is extremely challenging. A few realistic challenges are listed below:

– There exists no blueprint for how to design and operate learning-based networks at scale. While ML promises self-regulated autonomous operation, exploiting and modeling intrinsic network complexities, its fundamental advantages are far from clear.
– ML has been successfully used in several domains, and the choice of data, algorithms, architectures, etc. are somewhat clear for problems such as image recognition. However, the selection, complexity, and optimization of ML algorithms for optical networking problems requires substantial research efforts.
– Typically, ML requires huge amounts of data. How such data can be collected, processed (e.g., denoising, sampling, etc.), and transferred is an open problem.
– The concept of ML in networking is a multi-layer multi-domain problem, involving several entities and stakeholders. Furthermore, the toolchain to integrate ML frameworks with network orchestration and SDN/NFV at scale is missing, and substantial work needs to be done on network control, management integration, and evaluation efforts.

While addressing all these facets is an enormous undertaking, in this tutorial, we take the initial steps towards this overall objective and introduce the major concepts and applications of ML in optical networking. In Section II we introduce various aspects of ML, including the general framework, algorithms, and evaluation techniques. We follow this up with data management considerations in Section III and highlight relevant challenges. Section IV introduces network management architectures incorporating ML-driven building blocks for network automation. In Section V we propose several ML use cases, together with concrete real-world studies. Finally, Sections VI and VII give an outlook for furthering ML research in optical networking and draw conclusions, respectively.

II. Machine Learning

A. Introduction and Workflow

ML is typically thought of as a universal toolbox, ready to be used for *classification* problems, identifying a suitable category for a new set of observations, and *regression* tasks, estimating the relationship among given data samples. In fact, it is a diverse field comprised of various constituents and necessitates: a software ecosystem including data monitoring and transformation, model selection and optimization, performance evaluation, visualization, and model integration, to name a few. Explicitly, ML refers to computational representation of a phenomenon aimed at execution of a task, given a certain performance and based on a given environment.

Figure 2 depicts a typical workflow of a ML framework—sometimes referred to as knowledge discovery in databases (KDD)—focusing on algorithm development and test cycle. Initially, ML models are constructed in a *training phase*, where data is retrieved from historical databases and pre-processed to remove or normalize outliers, handle missing data, filter and aggregate parameters, etc. The next step relates to data transformation, selecting relevant data features, minimizing redundancies, data format adaptation to the given task, etc. This is a crucial step in the ML workflow, as not all data at hand is necessarily useful in terms of algorithm performance and complexity—the curse of dimensionality. The most important steps of model selection and behavior learning are typically carried out in tandem. The process involves high-dimensional parameter optimizations and corresponding evaluations to trade off performance, complexity, and computational effort. Typically, some data is sampled from the original data set, termed as validation data, and is used together with the training data to independently verify the rules and patterns constructed in trained models. The goal of the validation stage is to ensure that ML models are not over- or underfitting the observed data. Note that, depending on the ML family, labeled

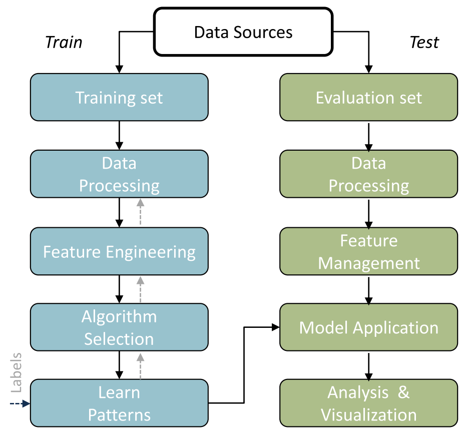


Fig. 2. Machine learning model construction and test workflow.

data may or may not be required (see the next section for further details). Finally, the models are exposed to test data, outcomes are mapped to representative knowledge (e.g., a particular pattern), and insights are delivered either to a dashboard or other related software components. Note that the presented workflow represents a typical procedure; however, practical constraints may enforce a different order or amalgamation of some steps, among other variations.

B. Algorithms

ML approaches may be categorized based on objectives of the learning task, where these objectives may target pattern identification for classification and prediction, learning for action, or inductive learning methods. The algorithms may be further classified into three distinct learning families [16], i.e., *supervised learning*, *unsupervised learning*, and *reinforcement learning*. *Semi-supervised learning*—or hybrid learning—is sometimes considered a fourth branch, borrowing features from the supervised and unsupervised categories [17].

In this subsection, we introduce the ML families, as depicted in Fig. 3. The main goal is to introduce the reader to typical ML algorithms, together with their most commonly associated applications, e.g., predictive maintenance based on supervised learning [18]. Note that the list of algorithms is not exhaustive and the interested reader can find many other algorithms in the references provided.

(i) Supervised Learning: Supervised learning (SL) makes use of known output feature(s), named *labels*, to derive a computational relationship between input and output data. An algorithm iteratively constructs a ML model by updating its weights, based on the mapping of a set of inputs to their corresponding output features. SL may be further categorized into classification and regression tasks, depending on whether discrete or continuous output features are used. In the following, we discuss a few SL algorithms [19].

1) *K-Nearest Neighbors*: A non-parametric ML algorithm based on dissimilarity between the samples. For classification problems, suppose there are n samples, $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ in space $\mathbb{R}^d$, where X and Y represent input samples and their class labels, respectively. For a new data point (X, Y), using a distance measurement, all the samples would be ordered by the distances, e.g., $|X_1 - X| \leq \ldots \leq |X_n - X|$. The class that owns the most samples among the k-nearest samples of X, where k is a user-defined parameter, is considered the class of the new data point. In a regression application, the algorithm is used to predict the value of a continuous variable, where the predicted outcome is the average or weighted-distance average of the k-nearest neighbors.
2) *Artificial Neural Networks (ANN)*: A ML approach comprised of one input and one output layer and one or more hidden layers in between, where each layer could be composed of several neurons [20]. Features (X) are fed into the network through the input layer, where the neurons in the input layer are connected with the

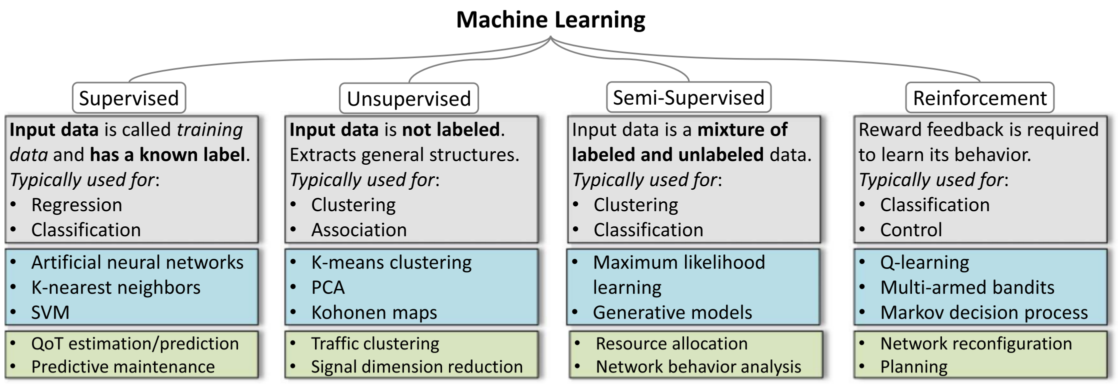


Fig. 3. ML families. The first box in each column identifies the main characterization of the ML approach; the second box identifies examples of algorithms used in the approach; and the third box indicates examples of applications that can take advantage of the approach.

neurons in the next layer, i.e., the first hidden layer, and so on. The neurons on the output layer are directly connected with the outputs (Y), as shown in Fig. 4.

In a typical fully connected ANN, the nonlinear mapping between layers could be given by relu($X \times W + b$), where W represents the weights, b represents the biases of the connections, and relu is the activation function, $\max(0, x)$. Further ANN parameter optimizations are carried out using algorithms like gradient descent, etc. Other types of neural networks include convolutional neural networks, recurrent neural networks, etc. Typical challenges associated with ANNs relate to choice of number of hidden layers and neurons, computational overhead, multidimensional parameter optimizations, etc.

3) *Support Vector Machine (SVM)*: A classification technique targeted at separating samples of different classes in a given feature space. The space is divided by maximizing margins (i.e., gaps) between the classes, where new data points are classified based on which side of the gaps they fall on. SVMs could be categorized into linear and nonlinear SVMs. For linear SVMs, the inputs are linearly transformed, whereas in the nonlinear case, the inputs are transformed into another space by a kernel mapping. A common kernel mapping is the Gaussian radial basis function, $k(x_i, x_j) = \exp(-\gamma |x_i - x_j|^2)$, where $\gamma > 0$, and x_i and x_j are two samples.

The algorithm is also classified by soft and hard margins. In a binary classification case, the margins are considered hard when they are represented by 1 and -1. When the loss function $\max(0, 1 - y_i(\omega \times x_i - b))$ is maximized, the margins are considered as soft.

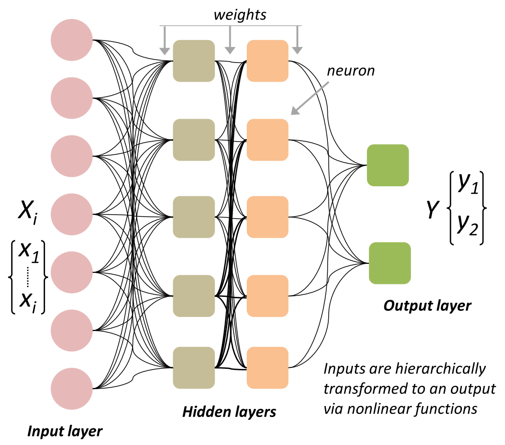

weights
neuron
X_i
$\begin{Bmatrix} x_1 \\ \vdots \\ x_i \end{Bmatrix}$
$Y \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$
Output layer
Inputs are hierarchically transformed to an output via nonlinear functions
Hidden layers
Input layer

Fig. 4. Illustrative example of a dual-layer ANN architecture.

Optical networking applications of SL algorithms include resource optimization by estimation and eventual prediction of network state parameters for a given set of configurations (e.g., symbol rates, optimum launch power, etc.) [21]. Another application is ML-driven fault identification, based on historical traffic or network function patterns [11].

(ii) Unsupervised Learning: While SL provides a clean-slate approach to ML model construction, in practice, labeled data is neither easily accessible nor abundantly available. Unsupervised learning (USL) aims to build representation of a given data set without any label-driven feedback mechanisms. USL may be further classified into clustering of data into similar groups, or association rule discovery, identifying relationships among features. A few USL algorithms are discussed below.

1) *K-Mean Clustering*: An unsupervised ML algorithm, which partitions all of the samples into k clusters based on dissimilarity metrics. Considering n samples in space $\mathbb{R}^d$, $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, k-mean clustering tries to cluster all the n samples into k classes, centered by the set $S = S_1, S_2, \ldots, S_k$. The objective function could be defined as $\arg\max_S \sum_{i=1}^{k} \sum_{x \in S_i} |x - \mu_i|^2$, where μ_i is the mean of all the points in S_i.

2) *Principal Component Analysis (PCA)*: Transforms the original variables into linear uncorrelated variables based on singular value/eigenvalue decomposition. Typically used as a dimension reduction approach. Consider a data matrix X, with zero-mean columns, where each row in X represents one observation and each column represents a single attribute. PCA transformation aims to reduce the data matrix X from n dimensions into p dimensions by multiplying a weights matrix $W = (w_1, \ldots, w_p)$, where W is obtained by the

objective function arg max W^TX^TXW. The value of X^TX is the largest eigenvalue of X, and W is the corresponding eigenvector of X.

3) *Self-Organizing Maps (SOM)*: This is a method for dimensionality reduction, where an ANN (the map) is trained using unsupervised learning to approximate samples in a high dimensional space. Typical use includes feature identification in large datasets. In the training phase, the weights within the network are updated by competitive learning, described by the pseudocode in Algorithm 1. s is the current iteration number, t is the index of the target input data vector in the input data set D, $D(t)$ is a target input data vector, v is the index of the node in the map, W_v is the current weight vector of node v, u is the index of the best matching unit (BMU) in the map, $\theta(u, v, s)$ is a penalizing parameter based on the distance from the BMU, and $\alpha(s)$ is a learning coefficient.

Algorithm 1: SOM

```
1: Randomize the node weight vectors in a map
2: **repeat** (**for each** episode)
3:    Randomly pick an input vector $D(t)$
4:    **repeat** (**for each** node in the map)
5:          Use the Euclidean distance formula to find the
6:          similarity between the input vector and
7:          the node's weight vector
8:          Track the node with the smallest distance
9:          (termed as the best matching unit, BMU)
10:     **until** all the nodes are traversed
11:     Update the weight vectors of the nodes
12:     (in the neighborhood of the BMU)
13:     by pulling them closer to the input vector
14:     $W_v(s+1) = W_v(s) + \theta(u, v, s) \cdot \alpha(s) \cdot (D(t) - W_v(s))$
15: **until** $s$ is terminal
```

USL models may be naturally used for clustering of transport channels, nodes, or devices, based on their temporal and spatial similarities. Applications include traffic migration, spectral slot identification, etc. (see, e.g., [22]).

(iii) Reinforcement Learning: Reinforcement learning (RL) refers to ML mechanisms without an explicit training phase [23]. RL aims to build and update a ML model based on an agent's interaction with its own environment. The key difference with respect to SL techniques is that labeled input–output features are not provided, but the relationship is rather learned via application of the initial model to test data. The most well-known reinforcement learning technique is Q-learning [24], described below.

1) *Q-Learning*: This algorithm tries to find the best policies under specific agent and environmental Q values. The basic elements of Q-Learning include states $S = s_1, s_2, \ldots, s_n$ and actions $A = a_1, a_2, \ldots, a_m$. A policy π is a rule chosen by the agent, and it consists of (s_i, a_j), where at state s_i, action a_j is executed. The agent chooses policies based on the value function $Q(s, a)$.

The pseudo-code for Q-learning is shown in Algorithm 2. Here, r means rewards, which are defined by the agent, γ is a discount factor, and α is the learning rate. The ε-greedy policy means that with a given probability ε, the agent will choose a random action; otherwise, it will choose the action with maximal Q value.

Algorithm 2: Q-Learning

```
1:   Initialize $Q(s, a)$ arbitrarily
2:   **repeat** (**for each** episode)
3:        Initialize $s$
4:        **repeat** (**for each** step of episode)
5:           Choose $a$ from $s$ using policy derived
6:           from $Q$ (e.g., $\epsilon$- greedy)
7:           Take action $a$, observe $r, s'$
8:           $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$
9:           $s \leftarrow s'$
10:      **until** $s$ is terminal
11:   **until** end episodes
```

One of the core applications of RL algorithms in optical networks is network self-configuration, including resource allocation and service (re)configurations—both for physical and virtual infrastructure (see, e.g., [25]). Network management frameworks may be extended with RL to come up with cognitive actions.

C. Evaluations

The ML algorithm evaluation approach impacts the way a model is constructed and eventually selected among several competing options. A poorly defined evaluation criterion may result in an unoptimized model selection procedure, resulting in erroneous conclusions when comparing classifier performances. The key questions to be answered are how to build models, how to meaningfully evaluate their performances, and how to determine the best configurations. In this subsection we focus on the data and performance aspects of ML model construction and introduce several evaluation strategies.

(i) Data Aspects: The fundamental aspect of building a ML model is to separate the available data set into training, validation, and test sets. The training data is used to build the model, whereas the validation set is used to independently validate the model constructed during the training phase. Finally, the test data, which is never observed by the model during its construction process, is used for performance evaluations.

The reason to divide data into these sets is to avoid overfitting the training data as depicted in Fig. 5. It can be seen that as the model approaches convergence—underfitting, which refers to an overly simplistic model—the training and validation data sets show similar results; however, beyond this phase, training errors continue to improve, whereas validation errors start deteriorating—overfitting, which refers to an unnecessarily complex model. The model that enables the best performance for the validation set is selected as the optimum model. Based on data characteristics, various cross-validation approaches may be used to split the training and validation set for model construction. For instance, k-fold cross-validation includes splitting the

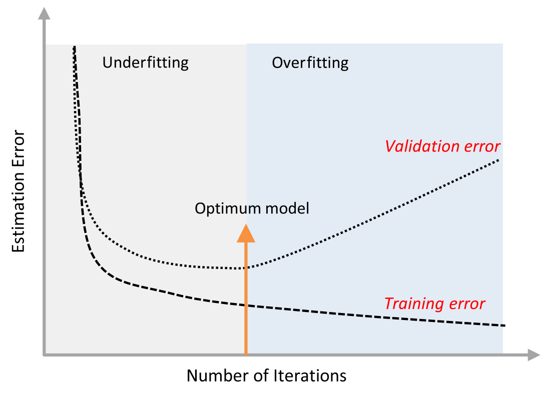


Fig. 5. Underfitting versus overfitting.

data set into k separate sub-sets—as shown in Fig. 6—then training from $k-1$ sub-sets, evaluating the remaining set, and repeating the procedure until all sub-sets are covered. Finally, the error metrics from each iteration are averaged to give overall performance estimation. Typically, k is set to 5 or 10 but may also take other values.

Another caveat is class imbalance, a common problem in a wide range of ML application areas. This problem usually appears in the form of dominant negative outcomes and may be partially resolved via data re-sampling methods.

(ii) Performance Aspects: There exist several evaluation techniques suited to a diverse set of ML models. For the sake of simplicity, let us assume a two-class classifier, where the classes represent a set of positive and negative outcomes.

– Confusion matrix: Table I lists the most common evaluation factors, i.e., true negative (TN), false negative (FN), false positive (FP), and true negative (TN). Such tabular representation is typically used as an evaluation approach for binary classifiers. The individual metrics are then used to calculate various sub-metrics.
– Sensitivity or True positive rate (TPR): TPR = TP/(TP + FN) and gives the probability that a true outcome is actually true.
– Specificity (SPC) or True negative rate: SPC = TN/(TN + FP) and gives the probability that a false outcome is actually false.
– Precision or Positive predictive value (PPV): PPV = TP/(TP + FP) and refers to the proportion of true outcomes given a set of outcomes classified as true.
– False omission rate (FOR): FOR = FN/(FN + TN) and refers to the proportion of false negatives given a set of outcomes classified as false.
– ROC and AUC: The receiver operating characteristic (ROC) curve is obtained by plotting TPR as a function of FPR and varying the class decision thresholds. An ideal classifier would entirely separate the two classes, resulting in a ROC curve that passes through the top left (100% sensitivity and specificity), as shown in Fig. 7.

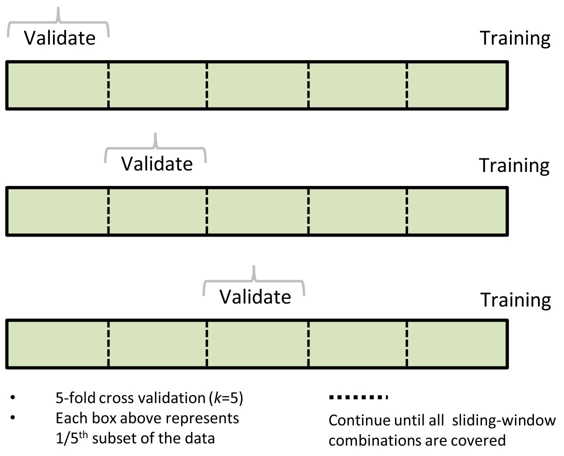

Validate
Training
Validate
Training
Validate
Training
5-fold cross validation (k=5)
Each box above represents 1/5th subset of the data
Continue until all sliding-window combinations are covered

Fig. 6. k-fold cross-validation.

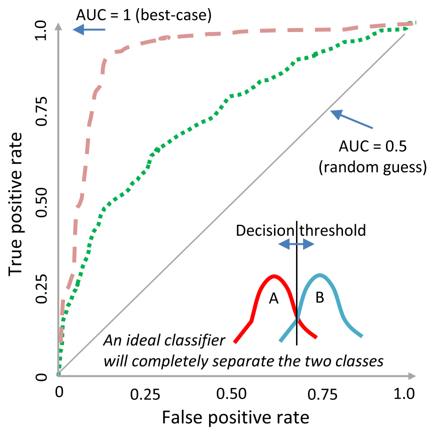


Fig. 7. ROC- and AUC-based classifier evaluation. The dotted and dashed lines are examples of typical ROC curves; the classifier corresponding to the dashed line provides better performance.

TABLE I
Confusion Matrix for a Binary Classifier

Total Population	Positive (True label)	Negative (True label)	
Predicted Positive	TP	FP	Positive Predictive value (PPV), Precision = TP/Predicted Positive
Predicted Negative	FN	TN	False Omission Rate (FOR) = FN/Predicted Negative
	True Positive Rate (TPR), Recall/Sensitivity, probability of detection = TP/Positive (True label)	False Positive Rate (FPR), Fall-out, probability of false alarm = FP/Negative (True label)	

The area under the curve (AUC) is the area under the ROC curve, and it represents the precise quality of the binary classifier, regardless of the decision threshold.

In the case of multi-class problems, binary classification approaches may not be used to evaluate performance. Below we describe several evaluation metrics, where y represents true values, $\hat{y}$ represents predicted values, and $\bar{y}$ depicts the average of the true values. The error e is obtained by $y - \hat{y}$.

– Accuracy: The rate of correct predictions made by the model over a test data set, typically via simple network management protocol.
– MAE and MAPE: Mean absolute error, mean($|e|$), and mean absolute percentage error, mean($100 \times \mathrm{abs}(e)/y$).
– MSE: Mean square error, mean(e^2).
– R^2: The ratio between predicted variation and true variations, $R^2 = SS_E/SS_T$, where $SS_T = \sum_i (y_i - \bar{y})^2$, and $SS_E = \sum_i (y_i - \hat{y}_i)^2$. It indicates how close the predicted values are to the real values.

Note that the presented techniques are not exhaustive by any means but rather an introduction to standard evaluation measures. There are several other approaches presented in the literature and are typically used based on the data and the problem at hand.

III. Data Management

An optical network generates a large amount of heterogeneous data streams, which must be fetched, processed, and analyzed in a timely manner to enable everyday network operations. In order to enable data-driven ML analysis, it is important to explore several aspects of network data including its extent, monitoring, query mechanisms, storage, and representation attributes. In this section, we address these data management features.

A. Sources

The first step in enabling ML-driven network operations is to understand the variety of network data sources, ranging from physical layer channel information all the way to applications and services. In the following, we discuss the five most common sources of ON data collection.

– *Network probes* include both intrusive and non-intrusive data access, e.g., measured optical power at an oscilloscope and deep packet inspection for filtering packets or collecting statistics, respectively.
– *Sensors* are the core data sources of optical networks, whose goal is to measure a physical quantity (device, subsystems, systems, and environment). These sensors may be explicit, e.g., a temperature measure, or implicit to network equipment, e.g., timing error. Typical examples of sensor data include received bit error rate, aggregate traffic rate, virtual network function (VNF) flow information, amplifier gain, etc.
– *Network logs* include alarm and event data sets from different network devices. Service and support teams examine these logs to initiate network diagnosis and action recommendations.
– *Control signaling* refers to supervisory channel, header data, etc., typically used for path initialization, link control, channel setup, etc.
– *Network management data* is primarily focused on two types of data sources: first, configuration, topology, and connectivity data at different network layers, and second, monitored data (typically via simple network management protocol [SNMP] MIBs) [26] obtained from the network sensors described above.

The network data may be further categorized based on their type and form, as given below.

– *Static data* refers to condition monitoring at a given time instant and may include both performance and configuration data.
– *Dynamic data* refers to time variable quantitative parameters, for instance, temperature, packet loss rate, received and transmit power levels, etc.
– *Text data* includes log files, manuals, design specifications, device models, etc.
– *Multi-dimensional data* typically represents image data, including device snapshots, locations, etc.

Table II lists several optical network data variables and their corresponding device, subsystem, and system. Note that here we exclusively focus on network data metrics, excluding end-user information like transmission control protocol, user datagram protocol, etc. Different subsets of presented variables may be used for different ML use cases. For instance, optical signal-to-noise ratio (OSNR), bit error rate (BER), and optical power may be used for physical layer performance optimization, whereas packet count, traffic load, and flow count may be used for network reconfiguration applications.

B. Monitoring

Network monitoring comprises accessing the aforementioned diverse data types. The challenge, however, is that the network devices are typically provided by a diverse set of equipment suppliers with unique interfaces, models, descriptions, etc. Moreover, the traffic itself is extremely heterogeneous with different formats, wavelength ranges, etc. Consequently, next-generation networks will represent a wide variety of rapidly evolving application stacks, consuming both physical and virtual resources. Traditional network management tools are unable to efficiently tap into this data goldmine because they lack the capabilities to probe network states in real time, among other issues related to scalability, vendor lock-in, etc. For instance, SNMP-based monitoring largely relies on data access at fixed intervals, managed using trap-based alarms. While SNMP has served the industry long and well, network monitoring needs to rise to the new visibility requirements.

On the other hand, model-driven streaming telemetry is defined by operational needs and requirements set by

TABLE II
EXAMPLES OF OPTICAL NETWORK DATA SOURCES

Parameter	Transmitter	Receiver	Amplifier	ROADM	NMS	Scope	Shelf	Switch
Optical power	X	X	X	X				
BER		X						
OSNR	X	X	X	X		X		
Amplifier gain			X					
Fiber type					X			
Distance					X			
Line rate	X	X			X			
Client rate	X	X			X			
Pluggable type					X			
Traffic load					X			X
Temperature	X	X	X	X			X	X
Main supply voltage							X	
Frequency resolution						X		
Flow count								X
Packet count								X
Topology					X			
Header length								X

telecom network operators (e.g., OpenConfig, TAPI). Figure 8 illustrates the core components of model-driven telemetry. It enables vendor-agnostic network state monitoring on a continuous basis using time series data streams and abstracts data modeling from data transport [27], leveraging advanced open-source initiatives, like YANG—model—, Google remote procedure call (GRPC)—transport, etc. Note that the northbound and southbound communication incorporates data and configuration exchange, e.g., to transport data to an SDN controller, and triggered configuration from the same controller. Furthermore, one of the key differentiators in moving from legacy monitoring (e.g., SNMP) to model-driven telemetry is the use of subscription-based data access, as opposed to request- or trap-based desired data selection and retrieval.

Table III lists a few key differences between legacy SNMP and state-of-the-art streaming telemetry solutions. On one hand, most telemetry solutions are based on participation-based standards organizations, whereas GRPC follows its own open-source development structure. The key differences arise from the multi-vendor operability and scale of the projects, e.g., sFlow being a generalized solution, compared to GRPC focusing on the management and control planes, etc.

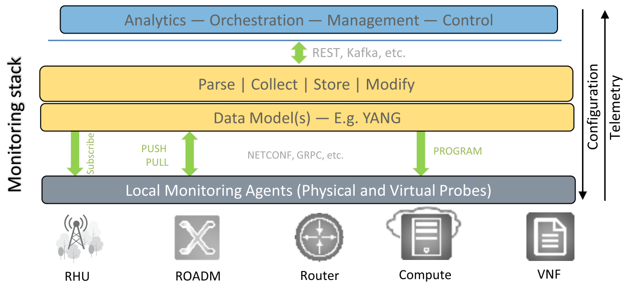

Analytics — Orchestration — Management — Control
REST, Kafka, etc.
Parse | Collect | Store | Modify
Data Model(s) — E.g. YANG
Subscribe
PUSH
PULL
NETCONF, GRPC, etc.
PROGRAM
Local Monitoring Agents (Physical and Virtual Probes)
Monitoring stack
Configuration
Telemetry
RHU
ROADM
Router
Compute
VNF

Fig. 8. Model-driven telemetry stack. REST, representational state transfer; RHU, remote hub unit; VNF, virtual network function.

TABLE III
LEGACY VERSUS MODEL-DRIVEN NETWORK TELEMETRY

Technology	SNMP	sFlow	NETCONF	GRPC
Models	MIBs	Structures	YANG	YANG
Transport	UDP	UDP	SSH	HTTP
Encoding	ASN.1	XDR	XML	GPB
Standard	IETF	sFlow	IETF	–
Type	Pull	Push	Push	Push
Application	Generic	Generic	Large-scale	Large-scale

C. Data Storage and Representation

Having discussed data sources and access mechanisms, data storage and representation also play an important part in data management frameworks. Due to the growing size of the network and complexity of the information sources, the data sources are often scattered across multiple storage and management systems with their own peculiar features. For example, there are storage technologies such as relational databases that require a dedicated data model or schema, whereas non-structured query language (NoSQL) databases serve the purpose without any relational model and have better scalability. However, both these technologies often fail when the volume and complexity of data becomes tremendously large. Recent advancements have demonstrated the utility of big-data technologies; for example, using Hadoop clusters, Spark, or Teradata to store and process huge amounts of data in a distributed fashion.

Nonetheless, it is not unusual for a typical network infrastructure to have its data scattered over these different heterogeneous data sources or systems that have been adapted over time to fulfill the requirements of the application they serve. This often leads to situations where data retrieval becomes a bottleneck due to the different representation schemes, constraints, naming conventions of the schema elements, formats used within the data models, etc.

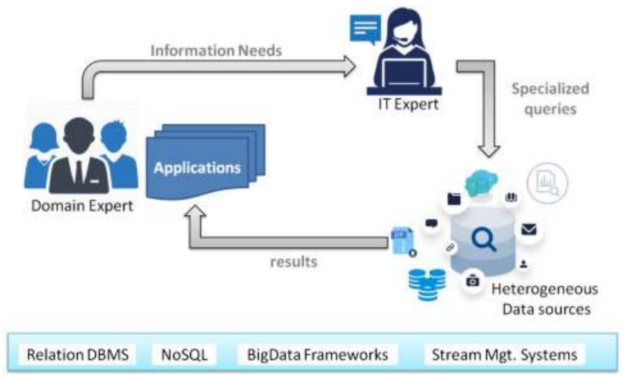


Fig. 9. Data storage and representation workflow and technologies.

Most enterprises today have dedicated teams of IT specialists to develop database queries for the domain experts. This means that the domain experts always have to pass their information needs to IT specialists (see Fig. 9), and this can drastically affect the efficiency of finding the right data in the right format that can be used for timely decision-making. From the modeling point of view, the data model is not the same as the domain model, leading to inefficiencies in data access protocols (schema, vocabularies, etc.). Clearly, it would be much easier to understand, analyze, and benefit from the data if it was uniformly accessible and represented in a domain-specific language. Recently, ML and graph technologies have been adopted to create a domain-specific abstraction over the data sources and make query answering easier for experts [28].

IV. Network and Model Management

In order to make use of advanced ML models, these models need to be integrated into the existing network software stack. On the other hand, multi-layer and multi-vendor network control and management is a complex task in itself, involving services incorporating cloud operations, core and metro transport, and mobile front-haul and back-haul connectivity all the way to heterogeneous user applications [29]. With localized and highly engineered operational tools, it is typical of these networks to take several weeks to months for any changes, upgrades, or service deployments to take effect. In this context, SDN is considered a game changer by many, owing to its agility, flexibility, and scalability, in contrast to traditional control and management platforms [30]. SDN distances itself from proprietary, device-specific operation to open, resource-driven control, enabling centralized, programmable, and automated services across multiple domains. The key issue is that while SDN is progressively getting adopted in the cloud, the ecosystem around centralized SDN platforms is ossified to legacy static and hardware-centric operations. The bottleneck does not necessitate introduction of entirely new tools, but rather an upgrade of legacy technologies—service provisioning, activation, fault management, etc.

In order to attain true network automation, centralized SDN control needs to be augmented with instantaneous data-driven decision-making using advanced monitoring and ML tools, feeding management and control planes alike. In the following we discuss the architecture of such an evolved platform.

A. Architecture

(i) Closed Loop Operation: The discussions regarding SDN have almost exclusively focused on separation of data and control planes, with little to no attention on the overall operational feedback loop, including monitoring, intelligence, and management functionalities. Figure 10 captures this theme and presents a high-level network architecture, where central offices (CO) consist of intra- and inter-data-center infrastructure. The intra-data-center resources comprise storage, computation, and networking, whereas inter-data center connectivity is provided by a transport network. Resources—physical or virtual—are continuously monitored, exposing real-time network states to the analytics stage, which in turn feeds into the control, orchestration, and management (COM) system.

This holistic platform not only caters to centralized and programmable control, but also makes ML-driven decisions to trigger actions, essentially connecting data-driven automation with policy-based orchestration and management. To this end, the COM architecture includes the NFV orchestrator providing network services, the virtual infrastructure manager (VIM) coordinating and automating data center workflows, the network orchestrator adopting hierarchical control architectures with a parent SDN controller abstracting the underlying complexity, and a monitoring and data analytics (MDA) controller that collates monitoring data records from network, cloud, and applications and contains ML algorithms.

(ii) Centralized and Distributed: Regarding MDA, a hybrid architecture may be envisioned, where every CO includes a distributed MDA agent that collates monitoring data from the network, cloud, and applications as well as a centralized MDA controller [31]. The MDA agent exposes two interfaces toward the MDA controller for collection

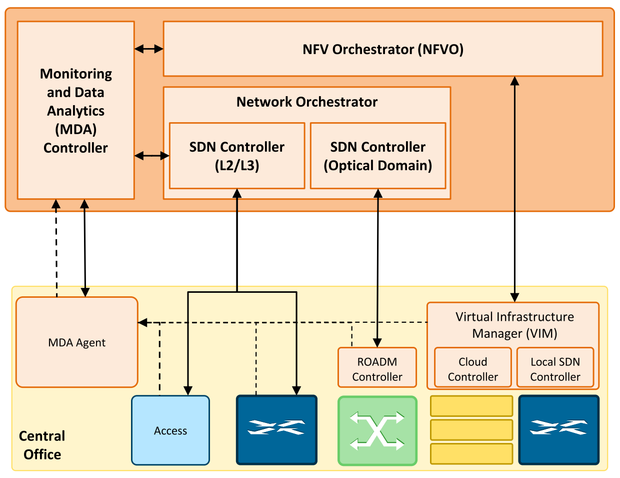


Fig. 10. Self-driven networking architecture.

of monitoring and telemetry data. In addition, specific interfaces for monitoring control allow the MDA agent to connect with the network nodes. The MDA agent includes a local module containing data analytics applications for handling and processing data records. The data analytics capabilities deployed close to the network nodes enable local control loops, i.e., localized data analysis, and consequent updated configurations.

The centralized MDA controller abstracts monitored data via suitable interfaces and implements a ML-based learning engine, where ML algorithms analyze monitoring data to discover patterns. Such knowledge can be used to make predictions and detect anomalies before they negatively impact the network. Such events can be notified in advance to the corresponding COM module (SDN controller or orchestrator), together with a recommended action. Note that a recommended action is a suggestion that the COM module can follow or just ignore and apply its own policies. The notification might trigger a network re-configuration, hence closing the loop and adapting the network to the new conditions.

(iii) Control Loop Examples: It is worth highlighting the importance of the control loops for network automation, as they fundamentally change the way networks are operated today—empowering truly dynamic and autonomous operation. As examples of control loops, we analyze the following, in the context of the presented architecture: i) soft-failure processing and ii) autonomic virtual network topology (VNT) management.

– *Soft-failure processing*: Let us focus on the optical layer where lightpaths might support virtual links (vlinks) in packet-over-optical multilayer networks. Soft failures can degrade a lightpath's quality of transmission (QoT) and introduce errors in the optical layer that might impact the quality of the services deployed on top of such networks. Such soft failures affecting optical signals can be detected by the MDA agents at intermediated nodes analyzing the optical spectrum acquired by local optical spectrum analyzers (OSA). Note that the acquired optical spectra entail large amounts of data [e.g., 6400 frequency-power ($\langle f, p \rangle$) pairs for the C-band for OSAs with 625 MHz resolution], so local analysis carried out at the MDA agents greatly reduces the amount of data to be conveyed to the MDA controller. Upon detection of a soft failure, the MDA agent notifies the MDA controller, which is able to correlate notifications received from several MDA agents and for several lightpaths to localize the element that is causing the failure [13]. Once the failure has been localized, e.g., in an optical link, a notification is issued to the COM module responsible for the resources (e.g., SDN controller) together with the recommended action of re-routing the affected lightpath excluding the failed resource.
– *Autonomic virtual network topology management*: VNTs are commonly created to adapt demand granularity to the huge capacity of lightpaths. Because demands vary throughout the day, defining a static VNT where vlinks are dimensioned with the capacity for traffic peaks leads to huge capacity over-provisioning; thus, VNT adaptation to follow demand requirements greatly reduces costs for network operators. One approach to dynamic management is to monitor capacity usage in the vlinks and to configure thresholds so that when a threshold is exceeded, the capacity of the vlink can be (reactively) reconfigured by adding or releasing lightpaths supporting this vlink. Another option is to monitor the origin-destination (OD) and reconfigure the VNT accordingly in a proactive manner [21]. For such proactive VNT adaptation to work, OD traffic prediction is needed to anticipate demand changes. OD traffic prediction is based on fitting models using ML algorithms that use historical OD traffic data collected by the MDA agents and stored in the MDA controller. Those OD traffic models are also stored in the MDA controller and are ready to be used for prediction. Periodically, the SDN controller can request the predicted OD traffic for the next period, e.g., the next hour, and use such prediction to create a traffic matrix that is the input of an optimization problem in charge of finding the best VNT configuration that meets both current and predicted capacity requirements.

B. ML Model Life Cycle

Once a ML model is constructed and deployed, the next step is to maintain this model over its lifetime. This process largely involves questions related to when and how a model should be updated. In particular, it is critical to determine the update cycle of a ML model considering the computational load and also ascertain the model validity while performing updates. In the following, we detail both of these aspects.

(i) Data Profile: A rather standard approach is to perform model updates at regular intervals, termed *constant update*, regularly adapting to evolving network states. While this simplifies model maintenance, it either suffers from unnecessary updates—in case network behavior has not changed significantly—or not an accurate enough model due to fast-paced changes in underlying network conditions. An orthogonal methodology is to trigger model updates when the data profile has changed significantly, termed *adaptive update*. This allows for efficient use of computational resources and adapts to real-time network behavior. Figure 11 shows an example of a normal and abnormal distribution for received optical power levels, triggering a model update.

(ii) Reconstruction: Model reconstruction involves incorporating new training data into the ML model. Once a model update decision has been made, the next step is

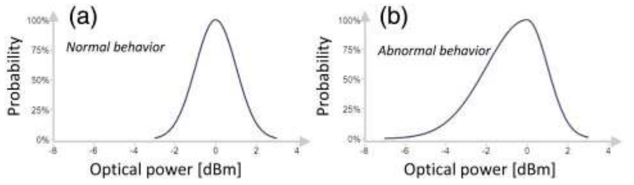


Fig. 11. Adaptive model update based on changing data profile, as opposed to fixed duration periodic updates. (a) Normal distribution of optical power levels. (b) Abnormal distribution of optical power levels.

to decide whether a batch-based or real-time update will be performed. Batch-based model reconstruction rebuilds the model with every refresh cycle, discarding previous data. This can be performed either in a sliding window or a weighted sliding window mode. Conversely, a real-time update refers to an incremental update of a preexisting model based on new data. This enables computational savings by not reconstructing the entire model, where resources have already been used. Furthermore, a real-time update allows for efficient data handling, as once the model is reconstructed there is no need to store the data for future model development.

V. Application Scenarios

In this section, we introduce several applications of machine learning in optical networks. Broadly, we consider network failure, reconfiguration, and performance-related use cases, discuss the background and relevant prior art, and follow up with proof-of-concept demonstrations. In particular, the control loop examples described in the previous section are detailed, together with two other sample applications, i.e., physical layer capacity optimization and optical spectrum analysis. The ML algorithms used for these applications include regression, SVM, and ANN, implemented using TensorFlow [32] for our proof-of-concept algorithm development.

A. Predictive Maintenance

With increasing network complexity and heterogeneity, business economics dictate the need for improved asset management to reduce, if not eliminate, downtime and improved resource usage [33]. Typical maintenance inefficiencies include the following.

– Network assurance agreements are signed, requiring maintenance cycles and periodic support, without considering current equipment or network health status.
– Network operators need to allocate effort to monitor and raise issues with support teams. This is typically done after a failure has occurred. Furthermore, in case of early support requests, repairs are quick fixes due to lack of a global operational perspective.
– Even in the case of data gathering and processing, most of the analysis is manual and only used for post-failure diagnostic purposes.
– The entire cycle is based on a few highly experienced individuals, with issues related to single-point-of-failure, non-transferable skills, product variations, etc.

Typical examples of network faults may include cooling unit failure, laser degradation, subsystem control unit failure, etc. Early detection of equipment failure states and consequent remedial actions can prevent network downtime and enable scheduled preventive maintenance. The general functional hierarchy of a fault management system is given in Fig. 12, ranging from detection of a potential failure, all the way to resolution processes [12]. Most

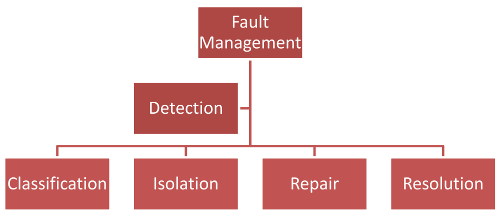


Fig. 12. Fault management functional hierarchy.

commercial equipment tolerates some errors until automatically tearing down the connection when some system thresholds are exceeded. While a restoration procedure could be initiated to recover the affected traffic, it would be desirable to anticipate such events and re-route the lightpath before it is disrupted. In case rerouting is necessitated, failure localization is required so as to exclude the failed resources from path computation. In addition, proactive failure detection would also allow time to plan the re-routing procedure, e.g., during off-peak hours.

In this use case, we address failure detection and localization blocks and demonstrate an analytics-enabled fault discovery and diagnosis (FDD) architecture, capable of proactively detecting and localizing potential faults (anomalies) as well as determining the likely root cause—based on SDN-integrated knowledge discovery [34]. The network segment consisted of ADVA FSP 3000 modules, carrying a (monitored) 100 Gb/s transport service. Here we follow a two-stage fault detection approach, where, in the first stage, the optical power levels are monitored as level-I data, and, in the second stage, localized level-II subscription-based monitoring consists of amplifier gain, shelf temperature, current draw, and internal optical power. The dual-stage approach allows for reduced monitoring and processing overhead compared to the all-at-once monitoring approach. The ensemble fault discovery and diagnosis framework is initiated within the SDN controller, as shown in Fig. 13(a), eventually triggering distributed analysis [Fig. 13(b)].

After initial data acquisition (aggregated in hourly bins), the algorithm is executed in two phases. In the first phase,

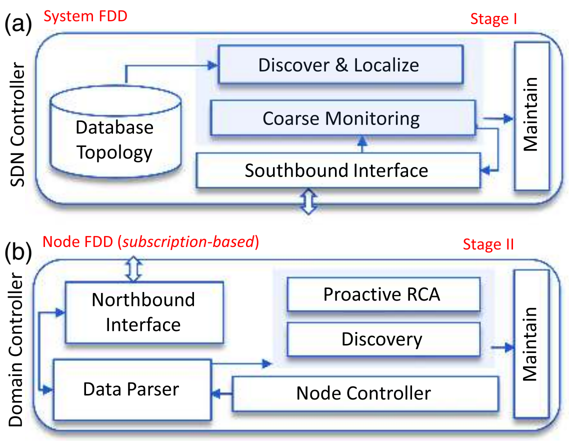


Fig. 13. SDN-integrated fault discovery (detection) and diagnosis. (a) System-level FDD. (b) Node-level FDD.

the records are partitioned, and we execute the generalized extreme studentized deviation test (GESD), a sequential-processing-based modification of Grubb's test [35], for deviation and classification tests (memory of 11 units), and neural network (backpropagation trained, in a 7(input) × 5(hidden) × 1(output) neuron configuration using tanh activation) based true fault detection. The ANN inputs represent positive and negative fault label values (a sequence of zeros and ones), including the indicated fault identified by GESD. The role of the ANN network is to make true abnormality decisions based on these fault label patterns.

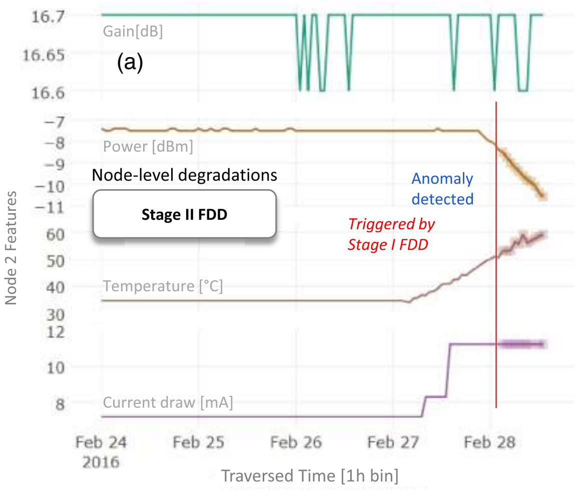
The engine performs first-level layer 0 monitoring via SDN abstraction through the southbound interface (SBI). It detects potential faults and localizes them to relevant network ports. The outcome is then distributed to a network maintenance application (RESTful), and also to the domain controller via SBI (Netconf). Second-level functional blocks include local fault discovery, followed by the root cause analysis, which maps metric fault measures to available node topology and lists potential root causes in a priority log.

After first-level detection of power level anomalies (not shown for the sake of conciseness), the second-level localized analysis is triggered. Figure 14(a) shows sample test points for these features, including temperature, amplifier gain, intermediate stage power, and current draw profiles, where the associated changes in a feature indicate the potential root cause. Figure 14(b) shows the distance metric, based on shape-based clustering [36], amongst the probable fault root causes, identifying the response of other features with respect to each others' changes. The shorter the length along an ellipse's minor axis, the higher the similarity between the two features; the rightwards tilt represents positive correlation, and vice versa. Note that the diagonal shows a line segment (minor axis of length zero), representing a perfect match, as the same feature is present on both the x axis and y axis. A high association is found between temperature and current draw as increasing temperature necessitates higher fan speed, whereas on the other hand, a decreasing draw may increase the temperature. Furthermore, a strong anti-correlation is identified between temperature and power, followed by relatively weaker dependency between current draw and power. Power may be ruled out as a potential fault because it starts deteriorating after the other two features; current and temperature are the two remaining causes, which may be processed further.

B. Autonomic Virtual Network Topology Management

Figure 15(a) shows an initial VNT where every vlink is supported by a 100 Gb/s lightpath in the underlying optical layer; the MPLS path for OD 6→7 is also shown. Let us assume that a 90% capacity threshold is configured in the vlinks, and in the event of a threshold violation, the capacity of some of the vlinks is increased. In our example, as OD traffic 6→7 increases, two threshold violations—for vlinks 1→6 and 1→7—will be issued, resulting in increased VNT capacity [Fig. 15(b)]. It is worth noting that the MPLS path for OD 6→7 is not affected by the VNT reconfiguration. As shown, the threshold-based reconfiguration can adapt the VNT capacity to traffic changes, such that the resources in the optical layer are allocated only when vlinks need to increase their capacity. However, the same number of transponders are required as in the static VNT approach; for instance, two transponders are installed in routers 6 and 7 and another four in router 1 reserved for vlinks 1→6 and 1→7.

Let us now assume that instead of monitoring vlink capacity usage, OD traffic is monitored in the nodes. By analyzing such monitoring data it could be observed that OD 6→7 is responsible for the registered traffic increment. In this case, let us assume that new vlinks can be created/removed in addition to increasing the capacity of the existing ones such that the VNT is actually changed. We propose an approach where OD traffic is periodically analyzed and the current VNT is reconfigured accordingly. An example that follows this approach is illustrated in Fig. 15(c), where the OD traffic 6→7 is predicted for the next



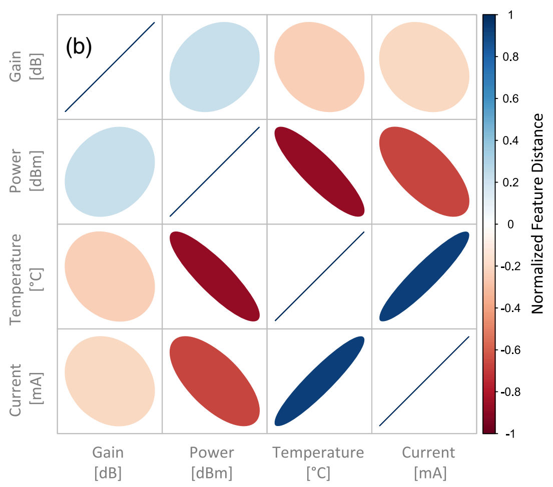


Fig. 14. (a) Localized fault discovery at the node. (b) Local feature similarity analysis for root cause analysis.

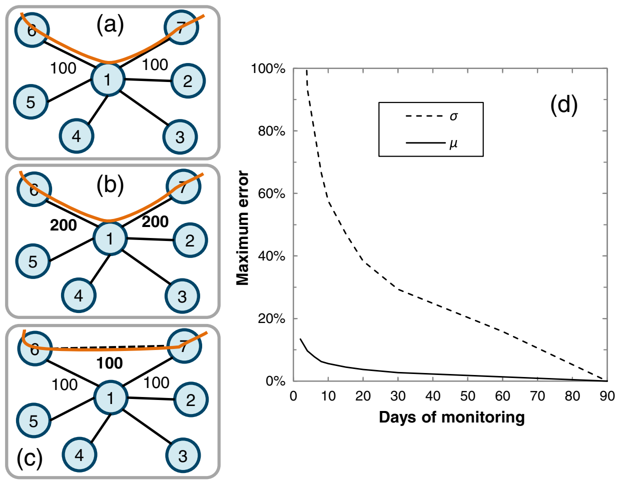


Fig. 15. (a)–(c) Reactive and proactive adaptation. Numbers in the inset represent link capacity in gigabits per second (Gb/s). (d) Maximum prediction error versus days of monitoring.

hour and a new vlink between nodes 6 and 7 can be created by establishing a lightpath, rerouting 6→7 traffic. Note that with this solution there are two fewer required transponders in router 1 compared to the previous approach.

(i) Proactive VNT Reconfiguration: Proactive VNT adaptation anticipates demand changes, consequently enabling better resource management with respect to reactive strategies. The VNT reconfiguration problem can be defined based on traffic prediction (VENTURE) [21], which can be formally stated as the following:

Given:

- The current VNT represented by a graph $G(N, E')$, N being the set of routers and $E' \subseteq E$ the set of current vlinks. Set E is the set of all possible vlinks connecting two routers,
- the set P with the optical transponders available in the routers; every transponder with capacity B,
- the current traffic matrix D,
- the predicted traffic matrix OD.

Output: The reconfigured VNT $G^{\star}(N, E^{\star})$, where $E^{\star} \subseteq E$, and the paths for the traffic are on $G^{\star}$.

Objective: Maximize current and predicted traffic matrices, while minimizing the total number of transponders used.

Since we are targeting VNT adaption to current and future traffic conditions, predictive traffic models that can accurately anticipate OD traffic are needed to define traffic matrices.

(ii) OD Traffic Estimation Models: The model estimation approach that we follow here is fitting ANN models. Different ANNs need to be fitted to separately predict the bitrate of each OD traffic flow. Each ANN receives as input p previous data values from the monitoring data repository and returns the expected average bitrate μ and the bitrate variance σ^2 at time t. Other variables, like the confidence interval at 95%, can be obtained by combining μ and σ^2 predictions.

One interesting analysis is the required training data (denoted as Y), as monitoring traffic during the right period of time is crucial to produce quality models while minimizing the time for new model availability. To evaluate this, we conducted experiments where μ and σ^2 are estimated and evaluated varying $|Y|$. Figure 15(d) shows the maximum error for σ^2 and μ estimations for different values of $|Y|$ between 2 days and 3 months. Although μ can be estimated with less than 5% maximum error in about 10 days, a maximum error of 60% is observed for σ^2 for the same duration. To decrease the maximum error, $|Y|$ needs to be increased up to 2 months to keep maximum prediction errors under 20%.

Figures 16(a) and 16(b) show one day of monitoring traffic data for two different traffic profiles, as well as the prediction of the μ model and the confidence interval at 95% obtained by combining the μ and σ^2 models (dashed lines). In addition, it is interesting to investigate the performance of an ANN model to smooth changes in the data. Figure 16(c) shows such adaptation to an evolutionary traffic scenario, where daily traffic pattern changes smoothly day by day. One can observe how the ANN model adapts from the initial (red) to the final (blue) traffic pattern while keeping its original accuracy without refitting.

(iii) Illustrative Results: Let us now apply the ANN traffic modeling approach to predict the OD bitrate of a core VNT, used as the input of the VENTURE optimization problem. Let us assume that VENTURE is triggered periodically, so per-OD prediction of maximum traffic for the next period is needed.

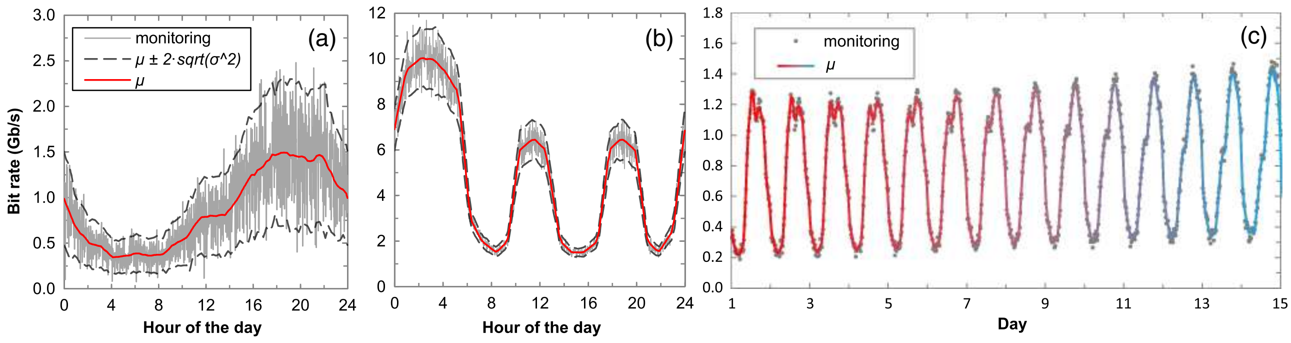


Fig. 16. (a) and (b) Prediction of min/max/avg for two different traffic profiles, and (c) ANN adaptation to smooth evolutionary bit rate.

For evaluation purposes, we compare the number of optical transponders required for static VNT management that is configured for peak traffic versus the number required with dynamic VNT reconfiguration. When no predictions are available, the optimization problem is triggered when a capacity threshold configured in the vlinks is exceeded, named *threshold-based*. All approaches were applied to a full-mesh 14-node VNT, where the initial capacity of each vlink ranges from 100 to 200 Gb/s. We ran simulations in an event-driven simulator based on OMNeT++. To measure the effect of volumetric and directional changes in traffic, we implemented traffic generators that inject traffic following the profiles in Figs. 16(a) and 16(b). The threshold-based model was configured with 90% threshold. Regarding OD traffic prediction, ANN models for μ and σ^2 were trained on a dataset with monitoring data belonging to the previous weeks. Maximum OD traffic prediction was considered targeting zero blocking probability.

Figure 17 plots, for each approach, the maximum transponder usage as a function of the load. Both the static- and the threshold-based approach show constant transponder usage for loads lower than 0.5, incrementing as a function of increasing load. For low loads, the capacity of vlinks in the fully meshed VNT is 100 Gb/s—in both cases—and is increased to 200 Gb/s for high loads under the static approach. The threshold-based approach, however, is able to manage the use of transponders by flexibly using available transponders to increment the capacity of vlinks running out of capacity. This allows it to achieve transponder savings up to 11% with respect to the static VNT approach. Interestingly, transponder usage scales linearly with the load with the predictive approach. Compared to the threshold-based approach, the predictive approach enables savings between 8% and 42%.

C. Physical Layer Capacity Optimization

An optical network is a mesh of individual entities contributing to the ensemble network behavior. One of the fundamental tasks in network operation is physical layer capacity planning. Traditionally, this has been achieved by precise engineering rules devised by subject experts, and the outcomes were configured for the lifetime of the network. While this approach made sense considering network size and limited configuration possibilities, it is restrictive in terms of data cooperation across the network for global optimization opportunities and suffers from limited scalability.

Recently published work has focused on predicting QoT based on measured BER and Q factor [37–39]. In this use case we instead aim to predict the optimum modulation format based on features such as symbol rate, channel load, number of spans, etc. In particular, we consider various multi-layer perceptron (MLP) architectures and show the performance in terms of classification accuracy and training time [40].

Figure 18 shows the setup considered in this work. At the transmit side, we considered typical features like channel symbol rate, pulse shaping filter roll-off, optimum launch power, and channel load on a given point-to-point optical link. The optical transmission channel was modeled using the Gaussian noise (GN) approach described in Ref. [41]. The back-to-back OSNR penalties were considered to be 0.5, 1, 2, and 3 dB for dual polarization quadrature amplitude modulation (DP-mQAM), where $m = 2, 4, 16$, and 64, respectively, accounting for optical and electrical component limitations. All of the considered features are listed in Fig. 18. We used GN simulations to generate our well-characterized data set, where the training data consisted of 6×10^4 unique records, validation data consisted of 2×10^4 records, and the test data set consisted of 2×10^4 records. The ANN classifier was a feedforward model with supervised learning based on the backpropagation algorithm, termed MLP, and was trained on the 11 input features and 1 output feature listed in Fig. 18(a). The output feature was a multi-level categorical variable representing the four modulation formats mentioned above. The different MLP architectures considered were a single layer with 5 neurons (MLP_A), a single layer with 10 neurons (MLP_B), a single layer with

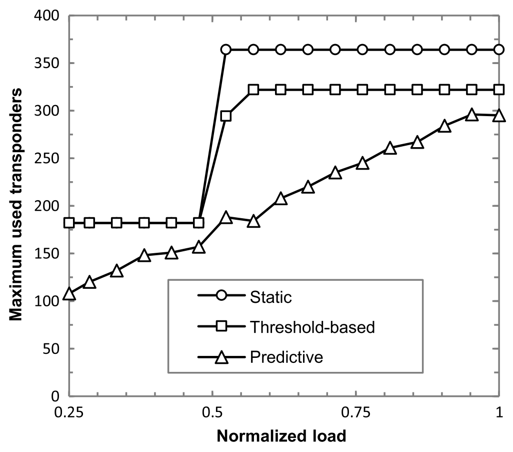


Fig. 17. Maximum used transponders versus load.

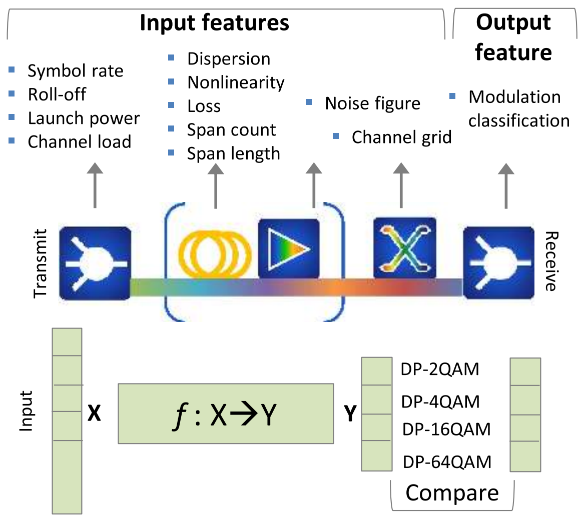


Fig. 18. Network setup and ML input and output parameters.

100 neurons (MLP_C), and two layers with 100 and 10 neurons, respectively (MLP_D).

Performance evaluation was carried out as a function of epochs, as shown in Fig. 19(a). The curves represent optimization behavior on validation data, as the test-data model was chosen based on convergence obtained on this validation set. It can be seen that all of the MLP architectures follow a similar saturation behavior after a certain number of epochs. MLP_A achieves 93.6% accuracy after 6000 epochs, followed by MLP_B with 95.8% accuracy and 5000 epochs, MLP_C with 97.1% accuracy after 3000 epochs, and MLP_D with 98.36% accuracy after only 2000 epochs. Clearly, MLP_D enables a significant 6% accuracy gain with three times fewer required number of epochs. Figure 19(b) shows the corresponding elapsed time to obtain the optimum model for the four classifiers. Interestingly, the least complex to the most complex configuration follows a linearly decreasing model training time, ranging from ~75 seconds to ~20 seconds. This behavior may be attributed to the lower number of required epochs for the more complex models, ascertaining that the more complex architecture does not significantly contribute to the training time. Nonetheless, MLP_D achieves an impressive ~98% classification accuracy, making it—or even a more complex architecture—a suitable choice for the modulation classification problem in optical networks.

The current use case may be extended to cooperative physical layer capacity optimization, consuming multi-layer traffic characteristics and performance metrics. For instance, in case of traffic congestion due to network load or link failures, the infrastructure layer may automatically adapt to the change, rather than leaving the remedial actions only to higher layers. On the other hand, traditional conservative capacity planning may be replaced with ML-driven solutions, self-adapting the capacity to a given network state.

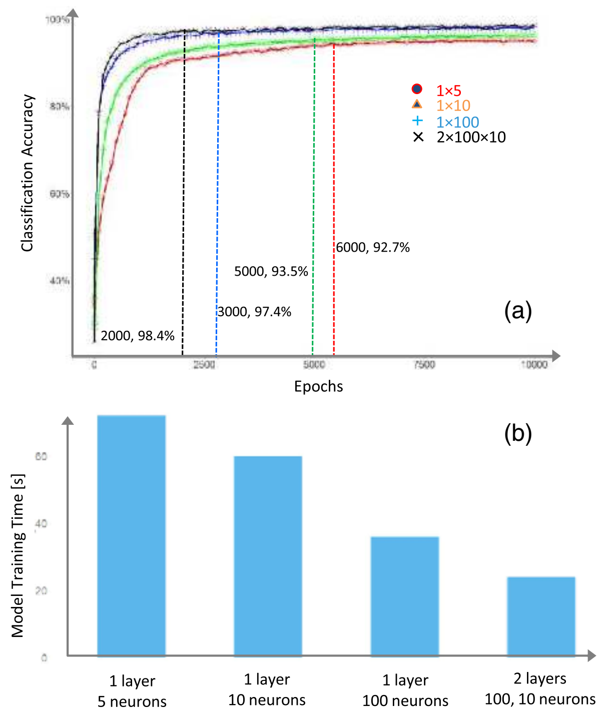


Fig. 19. (a) Classifiers' performance. (b) Training time.

D. Optical Spectrum Analysis

Optical signals typically undergo filtering effects due to laser drifts, narrow spectral grids, etc., eventually leading to QoT degradation. The optical signal spectrum is a useful indicator of such behavior, as these deteriorations result in spectral narrowing or clipping effects [42,43]. In this subsection, we identify degrading QoT due to filtering effects using ML, exploiting the baseline optical signal behavior of its central frequency being symmetric around the center of the assigned spectrum slot.

(i) Filter Effects: Figure 20 shows an example of the optical spectrum of a 100 Gb/s DP-QPSK modulated signal. By inspection, we can observe that a signal is properly configured when *i)* its central frequency is around the center of the allocated frequency slot; *ii)* its spectrum is symmetrical with respect to its central frequency; and *iii)* the effect of filter cascading is limited to a value given by the number of filters that the signal has traversed. However, when a filter failure occurs, the spectrum is distorted, and the distortion can fall into two categories: *i)* the optical spectrum is asymmetrical as a result of one or more filters being misaligned with respect to the central frequency of the slot allocated for the signal (filter shift, FS) and *ii)* the edges of the optical spectrum look excessively rounded for the number of filters, as a consequence of the bandwidth of a filter being narrower than the frequency slot width allocated for the signal (filter tightening, FT).

To detect the above-mentioned distortions, an optical spectrum (represented by an ordered list of "frequency, power" $\langle f, p \rangle$ pairs) can be processed to compute relevant signal points that facilitate its diagnosis. Before processing an optical spectrum, it is normalized to 0 dBm. Next, a number of signal features are computed as follows [13]:

- equalized noise level, denoted as sig (e.g., −60 dB + equalization level),

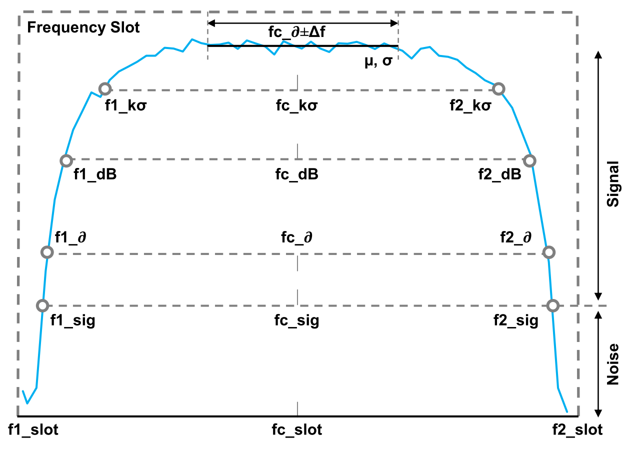


Fig. 20. Example of optical spectrum and signal features.

- edges of the signal, computed using the derivative of the power with respect to the frequency, denoted as ∂,
- the mean μ and the standard deviation σ of the central part of the signal, computed using the edges from the derivative ($f_{c}\_\partial \pm \Delta f$),
- family of power levels computed with respect to $\mu - k\sigma$, denoted as $k\sigma$,
- a family of power levels computed with respect to $\mu - k$dB, denoted as dB.

Using these power levels, two cutoff points can be generated and denoted as $f1(\cdot)$ and $f2(\cdot)$ (e.g., $f_{1\text{sig}}, f_{1\partial}, f_{1\text{dB}}, f_{1k\sigma}$). Additionally, the assigned frequency slot is denoted as $f_{1\text{slot}}, f_{2\text{slot}}$. Other features are also computed as linear combinations of the ones mentioned above.

These features are used as input for the subsequent failure detection and identification modules. Although relevant metrics are computed from an equalized signal, signal distortions due to filter cascading have not been corrected yet. As mentioned above, this effect might result in an incorrect diagnosis of a potential filter problem. To overcome this, we apply a correction mask to the measured signal. Such correction masks can be easily obtained by means of theoretical signal filtering effects or experimental measurements.

The two considered filter failure scenarios are illustrated in Fig. 21, where the solid line represents the optical spectrum of the normal signal expected at the measurement point, and the solid area represents the optical spectrum of the signal with failure. In case of filter shift, a 10 GHz shift to the right was applied [Fig. 21(a)], whereas the signal is affected by a 20 GHz FT in Fig. 21(b).

(ii) ML Algorithms for Failure Detection and Identification: In this application we make use of classification and regression algorithms. In case of classification, the objective is to classify unknown received data, e.g., an optical signal, and decide whether the signal belongs to the normal class, the FS class, or the FT class, whereas regression is used to estimate the magnitude of a failure. Once the optical spectrum of a signal has been acquired, and processed as described above, failure analysis is carried out. Figure 22 summarizes the workflow that returns the detected class of the failure (if any) and its magnitude. While ML algorithms are suitable for this task, we selected SVM for classification and linear regression for prediction.

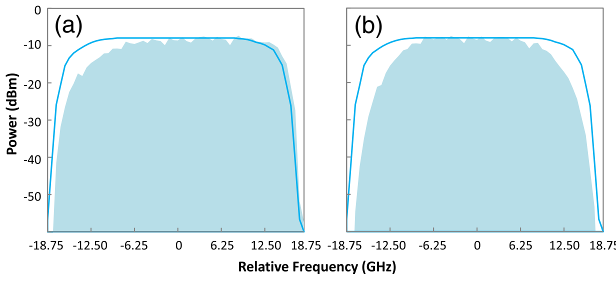


Fig. 21. Example of filter failures considered in this paper: (a) FS and (b) FT.

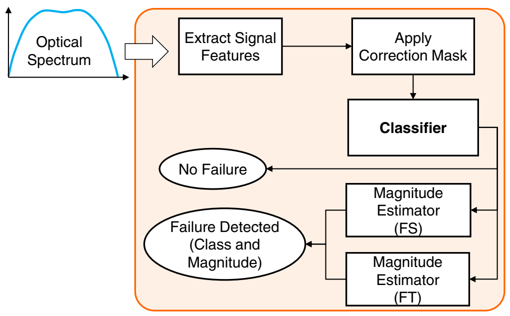


Fig. 22. Workflow for filter failure detection and identification.

(iii) Illustrative Results: In this section, we numerically study the proposed workflow using a testbed modeled in VPI Photonics, where the optical spectrum database was generated for training and testing the proposed algorithms. In this study, we focus on the cases where failure is limited to the first node. A large database of failure scenarios with different magnitudes (magnitude of 1 to 8 GHz for FS and 1 to 15 GHz for FT, both with 0.25 GHz step size) was collected.

Figures 23(a) and 23(b) show the accuracy of identifying FS and FT, respectively, in terms of the failure magnitude. Every point in Figs. 23(a) and 23(b) is obtained by considering all of the observations belonging to that particular failure magnitude, and above. This representation reveals the true accuracy of the classifier while considering failures with magnitude above certain thresholds. For instance, the accuracy of detecting FS in a data set comprising observations larger than 1 GHz (it comprises failures up to 8 GHz in which there are equal number of observations per each magnitude) is around 96%. On the other hand, the accuracy of the classifier becomes 100% for failures larger than 5 GHz.

Once the failures are detected, the filter shift estimator (FSE) and filter tightening estimator (FTE) can be launched to return the magnitude of the failures. In our case, the estimators were able to predict the magnitude of failures with very high accuracy, with mean square error (MSE) equal to 0.09091 and 0.00583 for FSE and FTE, respectively.

VI. Future Work and Industrial Perspective

While we presented several aspects of application of ML in optical networking, numerous challenges remain unaddressed. In particular, future work should consider the

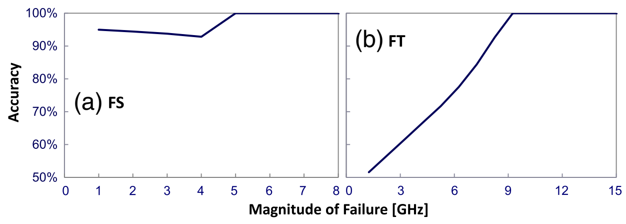


Fig. 23. Accuracy of the proposed method for (a) FS and (b) FT identification.

following research and practical issues to enable industry-wide adoption:

– As network hardware is slowly evolving towards an open and interoperable system, it is imperative that ML-driven solutions, regardless of vendors, products, and services, follow standardization processes.
– One of the key issues with applications of ML is the lack of available data sets. ML models using low-quality data may result in delayed technology introduction or complete abandonment. In particular, industry players should make an effort to produce anonymized data sets available to the larger community.
– While it is tempting to apply a series of ML algorithms to a given problem, care must be taken to trade off complexity and computational effort with performance. ML methods are still evolving, and a ready-made optimized solution may not be available at this point in time.
– When designing new frameworks and software architectures, care must be practiced to offer scalable solutions. This should range from applications, services, etc. to back-end storage facilities.
– The applications discussed may broadly be categorized as either network- or performance-related classification and pattern detection/prediction. As networks evolve towards data-dependent self-driven architectures, securing access to network data, authentication mechanisms, attack detection, and containment, etc. also need to evolve beyond conventional methods. As such, network security defines an orthogonal application of ML-related solutions.

VII. Conclusion

Traditional networks suffer from largely static operational and optimization practices that limit their scalability and efficiency. Machine learning provides a collection of techniques to fundamentally adapt to the dynamic network behavior. This tutorial has aimed to establish a reference for the practical application of machine learning in the optical networking industry. We have discussed several aspects, as summarized below:

• *Machine Learning Paradigms*: We introduced the fundamental concepts of ML, ranging from simple workflows to algorithm families and their evaluation methods. Several optical networking applications were tied to standard algorithms, where some were detailed in later sections.
• *Data Management*: Network data sources were elaborated, together with advanced monitoring and telemetry framework discussions. Furthermore, storage and representation aspects were highlighted with respective challenges.
• *Network and Model Management*: Novel architecture, including monitoring and data analytics, was presented, extending the conventional SDN control and management framework. We also discussed the operational life cycle of a deployed ML model, together with update strategies.
• *Case Studies*: Based on the discussed use cases, it becomes apparent that ML techniques can enable substantial benefits for optical network design, operation, and maintenance. Proactively detecting soft failures by monitoring multi-domain signaling or looking at unique optical signal properties would help increase network reliability. Traffic and network states may be used to improve resource utilization at the physical layer or trigger virtual link creation at higher layers.

While the application of ML for optical networks is still in its infancy, these learning-based techniques provide a promising platform for end-to-end network automation. We hope that this paper will contribute to the understanding of multi-disciplinary concepts and help improve network resource utilization and operational reliability.

Acknowledgment

The work has been partially funded from the EU project METRO-HAUL (G.A. 761727), from the AEI/FEDER TWINS project (TEC2017-90097-R), from CELTIC EUREKA project SENDATE-Secure-DCI (Project ID C2015/3-4), by the German BMBF (Project ID 16KIS0477K), and from the Catalan Institution for Research and Advanced Studies (ICREA).

References

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

[2] V. Vusirikala, "A decade of software defined networking at Google," in *European Conf. Optical Communications*, Sept. 2018.

[3] D. Rafique, H. Griesser, and J. P. Elbers, "Enabling 64Gbaud coherent optical transceivers," in *Optical Fiber Communications Conf. and Exhibition (OFC)*, Mar. 2017, pp. 1–3.

[4] V. Lopez, J. M. Gran, R. Jimenez, J. P. Fernandez-Palacios, D. Siracusa, F. Pederzolli, O. Gerstel, Y. Shikhmanter, J. Mårtensson, P. Sköldström, T. Szyrkowiec, M. Chamania, A. Autenrieth, I. Tomkos, and D. Klonidis, "The role of SDN in application centric IP and optical networks," in *European Conf. Networks and Communications (EuCNC)*, June 2016, pp. 138–142.

[5] M. Ruiz and L. Velasco, *Provisioning, Recovery, and In Operation Planning in Elastic Optical Networks*, Wiley, 2017.

[6] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2009.

[7] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, NY: Wiley, 2000.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed. Springer, 2009.

[9] D. Zibar, L. H. H. de Carvalho, M. Piels, A. Doberstein, J. Diniz, B. Nebendahl, C. Franciscangelis, J. Estaran, H. Haisch, N. G. Gonzalez, J. C. R. F. de Oliveira, and I. T. Monroy, "Application of machine learning techniques for amplitude and phase noise characterization," *J. Lightwave Technol.*, vol. 33, no. 7, pp. 1333–1343, Apr. 2015.

[10] Y. Huang, P. B. Cho, P. Samadi, and K. Bergman, "Dynamic power pre-adjustments with machine learning that mitigate

EDFA excursions during defragmentation," in *Optical Fiber Communications Conf. and Exhibition (OFC)*, Mar. 2017, pp. 1–3.

[11] D. Rafique, T. Szyrkowiec, H. Grießer, A. Autenrieth, and J.-P. Elbers, "TSDN-enabled network assurance: a cognitive fault detection architecture," in *43rd European Conf. Optical Communication (ECOC)*, 2017.

[12] D. Rafique, T. Szyrkowiec, H. Grießer, A. Autenrieth, and J.-P. Elbers, "Cognitive assurance architecture for optical network fault management," *J. Lightwave Technol.*, vol. 36, no. 7, pp. 1443–1450, Apr. 2018.

[13] A. P. Vela, B. Shariati, M. Ruiz, F. Cugini, A. Castro, H. Lu, R. Proietti, J. Comellas, P. Castoldi, S. J. B. Yoo, and L. Velasco, "Soft failure localization during commissioning testing and lightpath operation," *J. Opt. Commun. Netw.*, vol. 10, no. 1, pp. A27–A36, Jan. 2018.

[14] J. Mata, I. de Miguel, R. J. Durán, N. Merayo, S. K. Singh, A. Jukan, and M. Chamania, "Artificial intelligence (AI) methods in optical networks: a comprehensive survey," *Opt. Switching Netw.*, vol. 28, pp. 43–57, 2018.

[15] F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore, "A survey on application of machine learning techniques in optical networks," arXiv:1803.07976, Mar. 2018.

[16] S. Marsland, *Machine Learning: An Algorithmic Perspective*, CRC Press, 2011.

[17] O. Chapelle, B. Schlkopf, and A. Zien, *Semi-Supervised Learning*, 1st ed. MIT, 2010.

[18] S. Shahkarami, F. Musumeci, F. Cugini, and M. Tornatore, "Machine-learning-based soft-failure detection and identification in optical networks," in *Optical Fiber Communication Conf.*, Optical Society of America, 2018, paper M3A.5.

[19] S. B. Kotsiantis, "Supervised machine learning: a review of classification techniques," in *Emerging Artificial Intelligence Applications in Computer Engineering—Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, IOS, 2007, pp. 3–24.

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

[21] F. Morales, M. Ruiz, L. Gifre, L. M. Contreras, V. López, and L. Velasco, "Virtual network topology adaptability based on data analytics for traffic prediction," *J. Opt. Commun. Netw.*, vol. 9, no. 1, pp. A35–A45, Jan. 2017.

[22] S. G. Petridou, P. G. Sarigiannidis, G. I. Papadimitriou, and A. S. Pomportsis, "On the use of clustering algorithms for message scheduling in WDM star networks," *J. Lightwave Technol.*, vol. 26, no. 17, pp. 2999–3010, Sept. 2008.

[23] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA: MIT, 1998.

[24] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, May 1992.

[25] Y. V. Kiran, T. Venkatesh, and C. S. R. Murthy, "A reinforcement learning framework for path selection and wavelength selection in optical burst switched networks," *IEEE J. Select Areas Commun.*, vol. 25, no. 9, pp. 18–26, Dec. 2007.

[26] S. Pandey, M.-J. Choi, Y. J. Won, and J. W.-K. Hong, "SNMP-based enterprise IP network topology discovery," *Int. J. Netw. Manage.*, vol. 21, no. 3, pp. 169–184, May 2011.

[27] "gRPC—An RPC library and framework," GitHub. Inc, 2015 [Online]. Available: https://github.com/grpc/grpc.

[28] E. Wittern, J. Laredo, M. Vukovic, V. Muthusamy, and A. Slominski, "A graph-based data model for API ecosystem insights," in *IEEE Int. Conf. Web Services*, June 2014, pp. 41–48.

[29] L. Velasco, L. M. Contreras, G. Ferraris, A. Stavdas, F. Cugini, M. Wiegand, and J. P. Fernandez-Palacios, "A service-oriented hybrid access network and clouds architecture," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 159–165, Apr. 2015.

[30] R. Casellas, R. Martínez, R. Vilalta, and R. Muñoz, "Control, management, and orchestration of optical networks: evolution, trends, and challenges," *J. Lightwave Technol.*, vol. 36, no. 7, pp. 1390–1402, Apr. 2018.

[31] L. Gifre, J.-L. Izquierdo-Zaragoza, M. Ruiz, and L. Velasco, "Autonomic disaggregated multilayer networking," *J. Opt. Commun. Netw.*, vol. 10, no. 5, pp. 482–492, May 2018.

[32] "TensorFlow–An open source machine learning framework for everyone," GitHub. Inc, 2018 [Online]. Available: https://www.tensorflow.org/.

[33] S. R. Tembo, S. Vaton, J. L. Courant, and S. Gosselin, "A tutorial on the EM algorithm for Bayesian networks: application to self-diagnosis of GPON-FTTH networks," in *Int. Wireless Communications and Mobile Computing Conf. (IWCMC)*, Sept. 2016, pp. 369–376.

[34] D. Rafique, T. Szyrkowiec, A. Autenrieth, and J.-P. Elbers, "Analytics-driven fault discovery and diagnosis for cognitive root cause analysis," in *Optical Fiber Communication Conf.*, Optical Society of America, 2018, paper W4F.6.

[35] F. E. Grubbs, "Sample criteria for testing outlying observations," *Annu. Math. Stat.*, vol. 21, no. 1, pp. 27–58, Mar. 1950.

[36] J. Paparrizos and L. Gravano, "k-shape: efficient and accurate clustering of time series," *SIGMOD Rec.*, vol. 45, no. 1, pp. 69–76, June 2016.

[37] A. P. Vela, M. Ruiz, F. Fresi, N. Sambo, F. Cugini, G. Meloni, L. Potì, L. Velasco, and P. Castoldi, "BER degradation detection and failure identification in elastic optical networks," *J. Lightwave Technol.*, vol. 35, no. 21, pp. 4595–4604, Nov. 2017.

[38] L. Barletta, A. Giusti, C. Rottondi, and M. Tornatore, "QoT estimation for unestablished lighpaths using machine learning," in *Optical Fiber Communications Conf. and Exhibition (OFC)*, Mar. 2017, pp. 1–3.

[39] T. Tanimura, T. Hoshida, T. Kato, S. Watanabe, and H. Morikawa, "Data analytics based optical performance monitoring technique for optical transport networks," in *Optical Fiber Communication Conf.*, Optical Society of America, 2018, paper Tu3E.3.

[40] D. Rafique, "Machine learning based optimal modulation format prediction for physical layer network planning," in *Int. Conf. Transparent Optical Networks*, IEEE, 2018, paper Tu.A3.5.

[41] P. Poggiolini, "The GN model of non-linear propagation in uncompensated coherent optical systems," *J. Lightwave Technol.*, vol. 30, no. 24, pp. 3857–3879, Dec. 2012.

[42] T. Rahman, A. Napoli, D. Rafique, B. Spinnler, M. Kuschnerov, I. Lobato, B. Clouet, M. Bohn, C. Okonkwo, and H. de Waardt, "On the mitigation of optical filtering penalties originating from ROADM cascade," *IEEE Photon. Technol. Lett.*, vol. 26, no. 2, pp. 154–157, Jan. 2014.

[43] D. Rafique and A. D. Ellis, "Nonlinear and ROADM induced penalties in 28 Gbaud dynamic optical mesh networks employing electronic signal processing," *Opt. Express*, vol. 19, no. 18, pp. 16739–16748, Aug. 2011.