Learning Life Cycle to Speed Up Autonomic Optical Transmission and Networking Adoption

Luis Velasco, Behnam Shariati, Fabien Boitier, Patricia Layec, and Marc Ruiz

***Abstract*—Autonomic optical transmission and networking requires machine learning (ML) models to be trained with large datasets. However, the availability of enough real data to produce accurate ML models is rarely ensured since new optical equipment and techniques are continuously being deployed in the network. One option is to generate data from simulations and lab experiments, but such data could not cover the whole features space and would translate into inaccuracies in the ML models. In this paper, we propose an ML-based algorithm life cycle to facilitate ML deployment in real operator networks. The dataset for ML training can be initially populated based on the results from simulations and lab experiments. Once ML models are generated, ML retraining can be performed after inaccuracies are detected to improve their precision. Illustrative numerical results show the benefits of the proposed learning cycle for general use cases. In addition, two specific use cases are proposed and demonstrated that implement different learning strategies: (i) a two-phase strategy performing out-of-field training using data from simulations and lab experiments with generic equipment, followed by an in-field adaptation to support heterogeneous equipment (the accuracy of this strategy is shown for a use case of failure detection and identification), and (ii) in-field retraining, where ML models are retrained after detecting model inaccuracies. Different approaches are analyzed and evaluated for a use case of autonomic transmission, where results show the significant benefits of collective learning.**

***Index Terms*—Autonomic optical transmission and networking; Machine learning; Training function placement.**

I. Introduction

The revolution brought by 5G technology requires profound changes, not only in the way optical networks are built but also in the way they are fundamentally managed. Specifically, agile control and management tools must replace typical slow operation procedures that take days or even weeks to implement service deployments or network reconfigurations. In this regard, the software-defined networking (SDN) paradigm must be complemented with monitoring and data analytics (MDA) capabilities to enable *autonomic networking* [1,2]. Behind the autonomic concept, machine learning (ML) plays an essential role for a wide range of use cases in optical networks (see [3–7]). Examples include use cases from self-configuration to predictive maintenance and, at several levels, from transmission to single and multilayer network [8–14].

Assuming that ML is going to be used, one of the main problems that arises when it is applied to telecom scenarios is the lack of data required to train typical ML models, such as the artificial neural network (ANN) or the support vector machine (SVM). Note that ML training makes use of known output feature(s) to derive a model relating input and output data. However, existing legal and regulatory context limits the availability of real network performance measurement, and few research studies can be found where the data came from real measurements (see [12,14]). Moreover, obtaining training datasets belonging to specific pre-commercial and commercial technologies, as well as current and forecasted scenarios is a complex and extremely difficult task due to the vast combination of potential cases for the application of ML models. A possible solution is to use simulation (e.g., VPIphotonics or tools like the one in [15]) and lab or test-bed experiments to produce a sufficient amount of valid data for ML training and testing. Note that large datasets are needed to produce accurate ML models, as shown in [16]. Nonetheless, it is not clear whether this solution produces accurate ML models, as the data might not cover real deployments.

In fact, because optical transmission and networking is complex, wide ranged, and dynamic there are many examples where one just cannot generate all relevant examples of training and testing data, as data generation depends on the combination of elements. For instance, imagine the case of optical connection (lightpath) signal analysis using the spectrum acquired by an optical spectrum analyzer (OSA) installed in intermediate reconfigurable optical add-drop multiplexers (ROADMs) in the network [17]. Because the transfer function of optical filters that an optical signal has traversed affects the shape of its spectrum differently, we should be able to produce all the combinations of filter types for the different number of intermediate ROADMs to train accurate ML algorithms capable of identifying patterns, such as signal degradation. Even though we were hypothetically able to produce this amount of data, if new releases of ROADMs with different filter types were introduced into the network, new data would need to be

Manuscript received January 2, 2019; revised March 6, 2019; accepted March 11, 2019; published April 5, 2019 (Doc. ID 356788).

L. Velasco (e-mail: lvelasco@ac.upc.edu), B. Shariati, and M. Ruiz are with the Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.

F. Boitier and P. Layec are with Nokia Bell Labs, Nozay, France.

https://doi.org/10.1364/JOCN.11.000226

generated for the incremental number of possible combinations. Another example would be the case of real-time optical parameters analysis, like the state of polarization (SOP) [18] to implement autonomic transmission. It is clear that one just cannot produce data for every change in the parameters since such change might be the effect of physical perturbations with a different intensity, time duration, etc.

Regarding their location in the network, contrarily to the centralized architecture of SDN, ML algorithms might be executed as close as possible to data sources for particular use cases attempting to minimize the amount of data conveyed from the observation points that measure performance parameters to the ML algorithms, as well as at minimizing the response time [19] to allow control loop implementation, from subsystem to network. Continuing with our previous examples, the ML algorithm designed by the authors of [17] runs in every ROADM in the network to detect and localize soft failures degrading the quality of optical signals; once detected and localized, lightpaths can be rerouted excluding the failed element. As for the autonomic transmission example, the agent proposed in [18] runs inside transponders to enable local control loop implementation for fast device reconfiguration according to metered and forecasted data.

In this paper, we concentrate on a specific scenario: ML applied to lightpath analysis and generalize our previous work in [20,21] facing the problem of how to deploy highly accurate ML algorithms reducing the need to generate complete training datasets. Note that ML models can be retrained to enhance accuracy when model inaccuracies are detected, thus creating a *learning life cycle*. Specifically, the contribution of this paper is threefold:

1. A learning life cycle specifically designed for the deployment of ML-based algorithms in operators' networks is proposed in Section II. Starting from the motivation of ML retraining to improve the accuracy of ML models, the general options for ML training within such a learning life cycle are explored, including the out-of-field and in-field generation of datasets and ML training.
2. A two-phase strategy to facilitate ML algorithm deployment in operator networks is proposed in Section III. It consists of (i) training accurate models for a reference equipment scenario based on simulations and/or experiments carried out in laboratory or test-bed facilities and (ii) devising a proper *adaptation* mechanism that makes adjustments on the data for the specific lightpath being analyzed, which might have traversed different ROADM types along its route from the transmitter. Note that this strategy also facilitates the introduction of ROADMs with new filter types, as current vendors deploy new equipment releases in the network. This strategy is applied to a use case of filter failure detection and identification [20].
3. In-field retraining procedures are explored in Section IV, where starting from initially trained ML models, they are retrained with augmented datasets that include not-yet-considered patterns, added as soon as they are detected. Strategies for its practical implementation in optical networks are discussed: from typical individual learning, where each agent detects new patterns from their local sources and uses them for retraining, to collaborative learning, where agents spread knowledge among themselves to speed up the learning curve [22]. Because ML training is a hard task and requires large computation capabilities, analysis of distributed and centralized options reveals their pros and cons. In-field retraining alternatives are applied to an illustrative use case for autonomic transmission [21].

The discussion is supported by the results presented in Section V.

II. Proposed ML-Based Algorithm Life Cycle

Data availability is one of the main obstacles for the generalized deployment of ML-based algorithms in operators' networks. Ideally, one should start from a dataset with real data samples properly covering the considered features space. However, in many cases this is not possible as it was argued in the introduction section.

An alternative approach for ML training is to build an *out-of-field* training dataset with data generated from simulations and/or lab experiments. This approach might work provided that one can reproduce in the lab a reasonably large number of patterns to generate the training dataset. Once trained, the ML models can be deployed in the network and used for tasks such as prediction and classification. However, it is not easy and, depending on the case, virtually impossible to reproduce the large number of patterns that can appear once the ML-based algorithms are deployed in the network. Consequently, the ML models will present inaccuracies (e.g., in the form of prediction or classification errors). Such inaccuracies can be reduced by retraining the ML models with augmented training datasets that include new samples in areas of the features space not yet covered. Hence, the accuracy of the ML models must be monitored in the field by comparing the results of applying the ML models against the real measurements from the network. Once patterns for which the ML models do not meet the accuracy requirements are detected, the training dataset can be augmented and ML retraining can be triggered.

For illustrative purposes, Fig. 1 shows how ML models are improved when retraining is done after a prediction or a classification error is detected. Figure 1(a) represents a training dataset, where it can be observed how samples in the training dataset are not uniformly distributed and appear grouped in some areas (or subranges) of the features space, while few samples are available in other areas. This is a representation of a scenario where the training dataset comes from samples obtained by lab experiments and/or simulations, as well as for a scenario where the probability to observe certain patterns is low, whereas for other patterns the probability is higher. Figure 1(a) also shows the regression model obtained with the training dataset, as well as some confidence interval. Note that although a prediction can be done with the current regression model, there are some areas in the features space for which no training samples exist, so the prediction error

around those areas could be potentially high. In fact, let us imagine that a prediction is needed for a value in one such area [represented by the blue point in Fig. 1(a)]. Once the prediction error is detected after observing the real value measured in the network some time later, retraining can be carried out using the original training dataset augmented with the new pattern. Figure 1(b) represents the new regression model with improved accuracy.

Figures 1(c) and 1(d) presents an example for binary classification using SVMs; for the sake of simplicity, we consider linearly separable classes in a two-dimensional features space. Figure 1(c) illustrates that the current hyperplane perfectly separates the two classes of samples identified in the training dataset. After a classification error is identified by comparing the classification obtained by the current SVM classifier and the real data measured in the network, retraining is triggered and a more accurate SVM classifier is obtained [Fig. 1(d)].

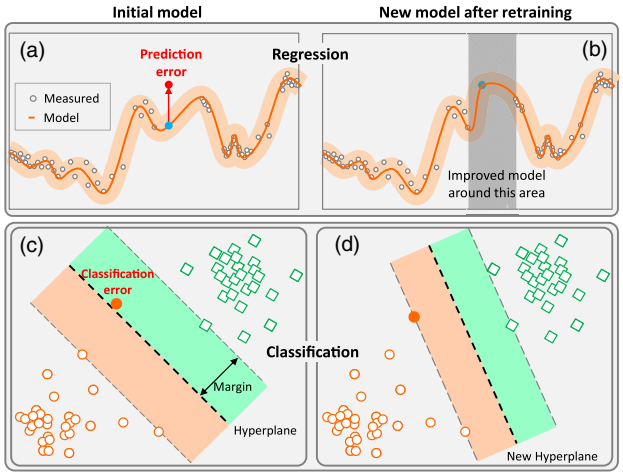

Fig. 1. Different models: (a), (b) Retraining regression and (c), (d) classification.

In view of this reality and trying to speed up the deployment of ML-based algorithms, the general ML-based algorithm life cycle presented in Fig. 2 can be followed. Two different activities and elements are defined in Fig. 2 as a function of whether they are carried out/exist out of the field (in blue color) or in the field (in green color). In particular, based on results obtained by simulations or lab experiments (labeled 1 in Fig. 2) an *out-of-field* dataset can be built (2). Since simulation and experiments might include certain assumptions and particularities that might not be true or present in the network, samples in the dataset must be generalized (3) to create a new dataset (4). The aforementioned process that reproduces patterns in the lab can be triggered as soon as new equipment and techniques are deployed in the network (0), as well as when inaccuracies are detected in the current ML models.

Assuming that ML models are already deployed in the network as a part of some algorithms (A), they are applied to real-time monitoring data (B) that has been conveniently generalized (C) to produce information such as predictions and classification. (D) Note that data samples can be stored and used for retraining purposes in an *in-field* data repository (E). The ML results are stored and later compared against real measurements to find inaccuracies, check the distribution of the samples in the dataset, and other tasks. (F). Note that ML training can be carried out considering any mix of samples in the *out-of-field* and *in-field* dataset repositories.

We next present two different use cases that exploit the proposed generic ML-based algorithm life cycle with the addition of specific extensions.

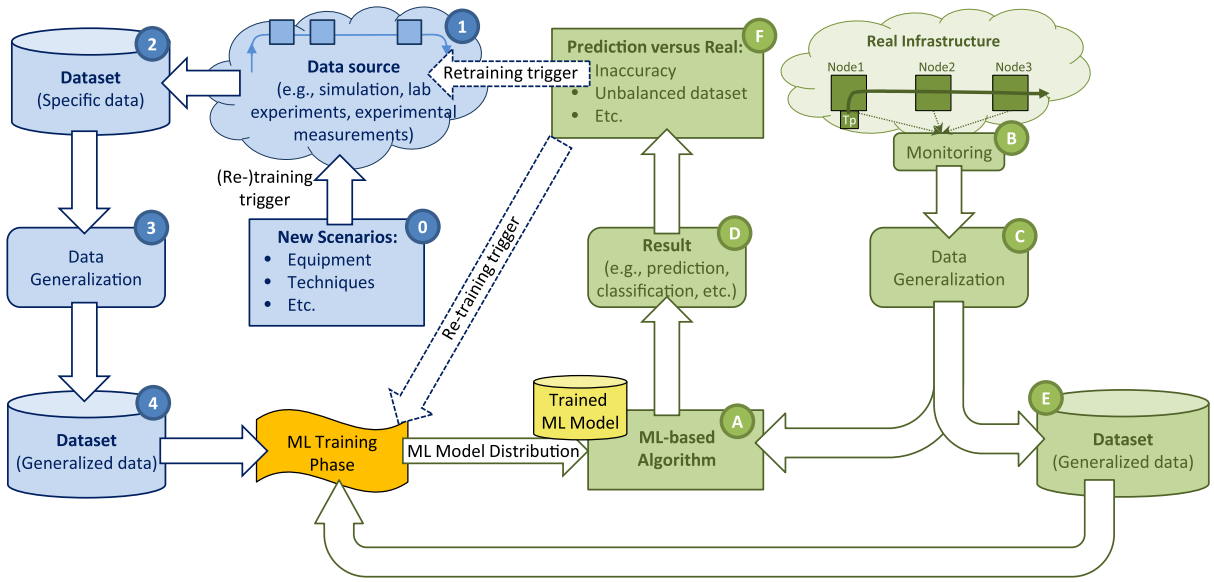

Fig. 2. Generic proposed ML-based algorithm life cycle.

A. Out-of-Field ML Training With In-Field Model Adaptation

This use case consists of two phases: (i) training accurate models for a reference infrastructure (i.e., transponders, ROADMs, and optical amplifiers) and (ii) devising a proper adaptation mechanism that makes adjustments on the data for the specific lightpath being analyzed, which might have traversed different equipment from those considered in the training phase.

Figure 3 gives an overview of the considered strategy. Scenarios with generic equipment or just one single type of equipment are used for simulation and/or lab experiments to produce a large dataset used for ML training and testing purposes [Fig. 3(a)]. Imagine that the ML model(s) are used for classification to detect and identify failures in optical connections. In the case where just one single equipment model from one single vendor is deployed in the field, the accuracy (defined in terms of failure detection and identification) of the ML algorithm will be equivalent to that obtained during out-of-field ML testing phase.

When new types of equipment are deployed in the network, new simulation and/or lab experiments must be carried out. However, this time the amount of simulation or experiments would exponentially increase to augment the dataset to consider all possible combinations of equipment that a lightpath could traverse. To avoid such an explosion of experiments, an adaptation function that converts the considered features to the specific equipment characteristics must be found so the resulting adapted features are equivalent to those used during the training phase [Fig. 3(b)]. Doing this would allow the trained ML models leading to *out-of-field* model training and *in-field* adaptation to stay unaltered, which makes it a robust feasible solution for networks with heterogeneous equipment.

B. In-Field ML Retraining

Similar to the previous use case, the in-field ML training case starts from an initial ML model trained out of the field [Fig. 4(a)]. Let us assume that the ML-based algorithm deployed in the field produces periodical estimations of near future bit errors based on features extracted from the receiver in the transponder. In this case, it is easy to keep track of the accuracy of the estimations because the real bit errors are also measured by the transponder. When model inaccuracy is detected, a learning loop can be triggered, where training data can be generated and used to feed *in-the-field* ML retraining to improve the accuracy of the current model. When a new model is produced, it can replace the existing one and the ML-based algorithm can continue to make predictions [Fig. 4(b)].

Although this option can be applied with no restriction to ML-based analysis, ML training is in general a

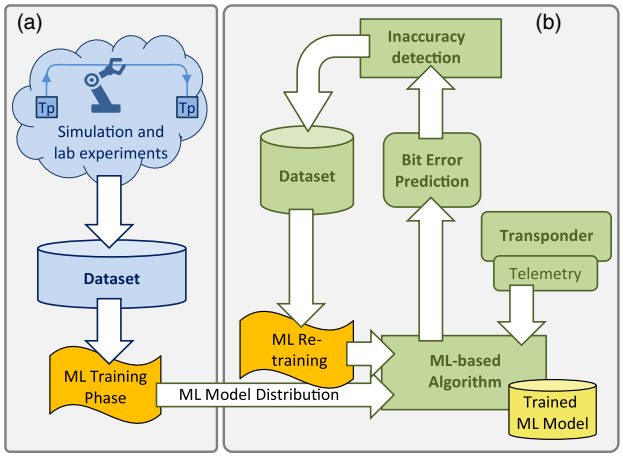

Fig. 4. Examples of in-field ML retraining: (a) out-of-field and (b) in-field.

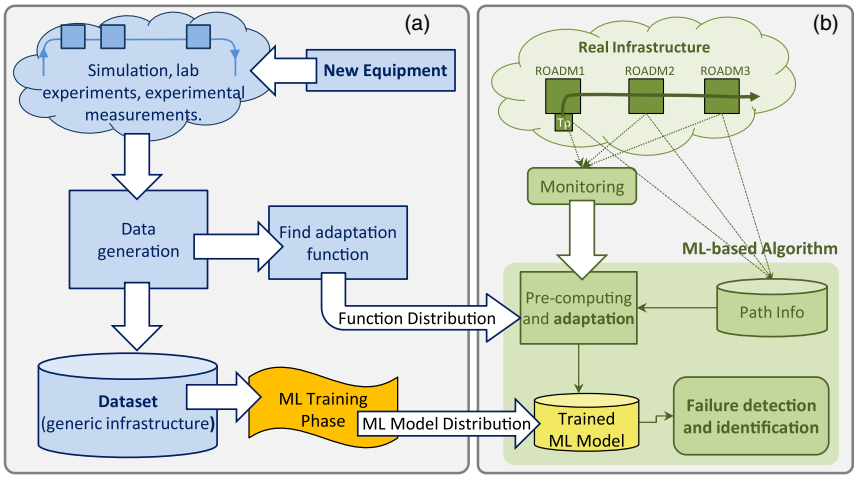

Fig. 3. Examples of out-of-field ML training and in-field model adaptation: (a) out-of-field and (b) in-field.

computationally demanding task. Therefore, the right place for its training must be studied, as many agents run in environments where computational resources are scarce.

The next sections focus on these two illustrative use cases implementing learning life cycles.

III. Out-of-Field ML Training With In-Field Adaptation

The out-of-field ML training option is studied through a lightpath failure detection and identification use case, where the spectra acquired by OSAs installed in intermediate nodes is analyzed for filter-related failure detection and identification. Note that filter-related failures [e.g., filter shift (FS) and filter tightening (FT)], noticeably deform the shape of the optical spectrum. Here, the *residual-based* approach developed in [23] is selected due to its potential to be adapted to different types of filters because of its dependency on the synthetic behavior of the filter responses. Note that one single filter type was considered in [17,23], which limits the deployment of ML approaches in real operator networks, which usually consist of equipment from different vendors. The most straightforward solution to overcome this limitation is to have different models trained about various types of filters that might be available in the network. Nonetheless, it makes the training phase very complex and data hungry. Yet, it will not be easy to comprehend the sequence of filters a priori and the responses of slightly nonidentical filters in the network might not be very well detected, necessitating even more combinations of models to have an appropriate generic model. In this section, we present an enhanced version of such an approach with the ability to be adapted to new filter types in the network. The application of *out-of-field* model training and *in-field* adaptation leads to a robust, yet feasible, solution for networks with heterogeneous filtering.

The residual-based approach lies in preprocessing the acquired optical spectrum by comparing it to the one that would be expected after passing the same number of filters that the signal passes. This comparison produces a residual representing the differential deformation that is used as input for a classifier that detects soft failures [Fig. 5(a)]. Two modules are required to compute the residual signal: (i) the expected signal calculator (ESC) [Fig. 5(b)] and (ii) the residual computation and adaptation [Fig. 5(c)]. The ESC module generates a theoretically calculated optical spectrum emulating a properly operating lightpath. The aim of ESC is to synthetically reproduce an averaged noise-free version of the measured optical signal. Then, the residual signal is easily obtained by subtracting the OSA-acquired signal from the signal generated by the ESC module. However, further elaboration on the residual signal is required to make it suitable for decision-making and training the classifiers. The elaborated residual signals can ultimately be used to train SVM-based classifiers to detect (i.e., decide whether a signal is normal or affected by a soft failure) and identify filter failures (i.e., decide between FT and FS).

The in-field adaptation is performed in (i) the ESC module by considering the specific filters that the signal has passed through [see three filter transfer functions in Fig. 6(a)] and (ii) the residual computation module that normalizes and adapts the residuals for the signal being analyzed. Following the procedure presented in [23], the calculated residual is normalized with respect to the mean value of the central part of the residual, so the mean becomes 0. This normalization approach is operational when the same type of filter exists in both out-of-field training

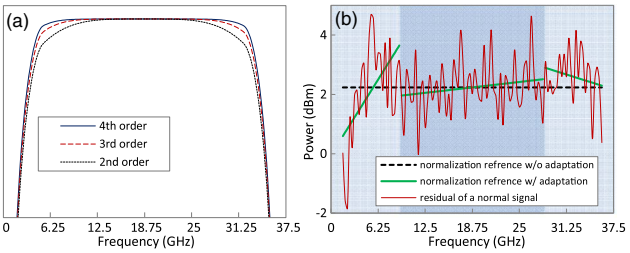

Fig. 6. (a) Three filter types and (b) 4th order Gaussian normalization.

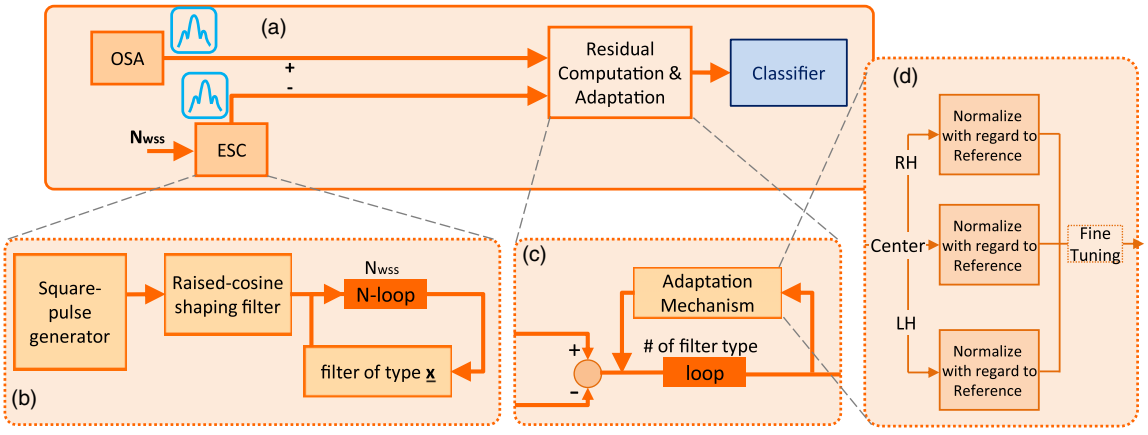

Fig. 5. Soft-failure detection and identification based on residuals analysis and in-field adaptation. (a) Residual-based approach, (b) expected signal calculator (ESC), (c) residual computation and adaptation, and (d) adaptation mechanism.

and in-the-field operation of the ML algorithm. However, it produces distorted results in the presence of other filter types. Considering this issue, we propose an adaptation procedure [Fig. 5(d)] that consists of dividing the residual signal into three segments [see Fig. 6(b)] and applying different normalization methods to each segment, reflecting the filter characteristics. The normalization reference of every segment is obtained by applying linear regression to the unnormalized version of the residual signal obtained for that segment. Then, the residual computation and adaptation module receives the signal, as well as the linear regression coefficients modeling three different normalization references that consider the filter characteristics. For this stage, the number of adaptation mechanism loops equals the number of filter types that the lightpath has passed through. By subtracting every segment of the unnormalized residuals from the corresponding normalization reference, a filter-type-agnostic residual signal is obtained. Note that, as the amount of filter cascading effect depends on the transfer function of the filter, there might be an undesirable deviation in the residual signals when the lightpath traverses different filter types; this deviation is compensated in the fine-tuning step. The amount of deviation can be computed locally, assuming that the mean value of the residual remains zero when the signal is in proper operation mode. Ultimately, a single classifier trained with the measurements collected in the lab based on a reference filter type can be used for optical spectra experiencing filtering effects from different types of filters.

IV. In-Field ML Retraining

To illustrate the in-field retraining option, let us consider a use case assuming the architecture in Fig. 7, where optical nodes (e.g., transponders, ROADMs in disaggregated scenarios) generate monitoring and/or telemetry data with performance measurements. Controlling subsystems, *device agents* can be designed to collect metered data from the device, analyze them by means of ML models, and send back specific device configurations to enhance transmission, thus resulting in a closed *device-wide* control loop [18]. On top of that, *node agents* expose a single interface to the SDN controller and enable local control loops to affect several subsystems. Finally, the centralized *MDA controller* running beside the SDN controller enables *network-wide* autonomic operations.

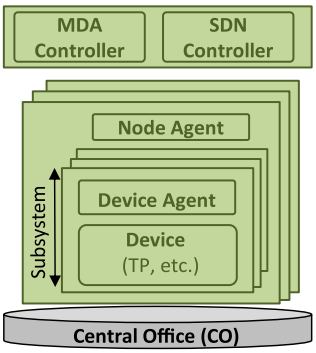

Fig. 7. Reference architecture to illustrate in-field retraining.

In line with Fig. 4, in-field retraining can be immediately triggered when model inaccuracies (in this use case, an inaccuracy is defined as an incorrectly detected case) are locally detected by device agents. In fact, depending on *how* the knowledge generated by inaccuracies is used to improve models and *where* the training task is carried out, several alternatives can be implemented. Attending to how knowledge is used, we have (i) *individual learning*, where generated knowledge is used for training and updating just the ML model of the detecting device, and (ii) *collective learning*, where the generated knowledge is spread and used for training and updating the ML models for every device. This alternative will speed up the learning curve, especially for *rare* patterns, as ML models are updated in every device when just one of them detects an unknown pattern. However, this alternative entails higher complexity than individual learning because the training data may require previous *normalization* to fit models with different characteristics. As to where training is performed, we have (i) *distributed training*, where training is executed locally (e.g., in the node agent), and (ii) *centralized training*, where training is implemented in a centralized element (e.g., the MDA controller).

Figure 8 illustrates the four how–where combinations, where labels help to identify how data flows. Individual learning is the most straightforward alternative, where the distributed (local) training does not require any data to be conveyed to the MDA controller at the expense of requiring extra computational resources in the node agents for ML training, whereas, in the centralized training, data must be sent to the MDA controller where more computational resources are usually available. In the case of collective learning with distributed training, even though training is performed locally, the MDA controller has the role of distributing training data after normalization to node agents, as such data normalization tasks require network-wide knowledge. Finally, collective learning with a centralized approach uses computational resources from the MDA controller for training the ML models of every device using normalized training data.

The above-described approaches are evaluated through the autonomic transmission use case (Fig. 9), where an optical receiver is dynamically configured in response to predicted short-term pre-forward error correction (FEC) bit error rate (BER) degradation [18]; the device agent using ML models is called the *autonomic transmission agent* (ATA). These ML models use the evolution within a time window of measured Stokes parameters representing the SOP as input parameters to return the expected BER for a target short-term interval (e.g., 100 ms). ANN was the selected ML algorithm due to its inherent capability to admit complex correlation between input and output variables while adding negligible overhead to subsystem operation. ANN requires one input for each of the last Stokes parameters in the analyzed window and produces a single output with the BER prediction for the target interval. Finally, BER prediction is used to increase or reduce the number of FEC iterations.

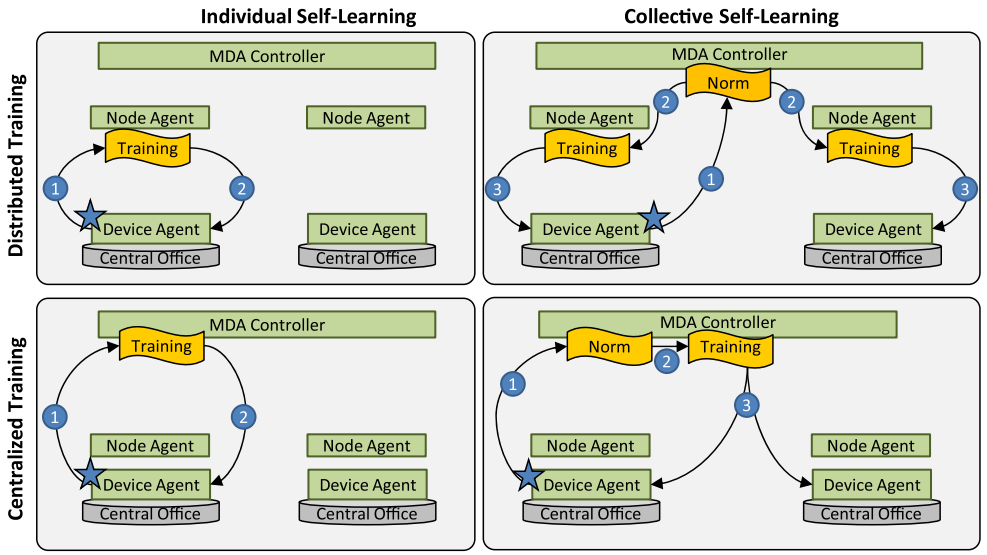

Fig. 8. Individual versus collective learning under centralized and distributed training.

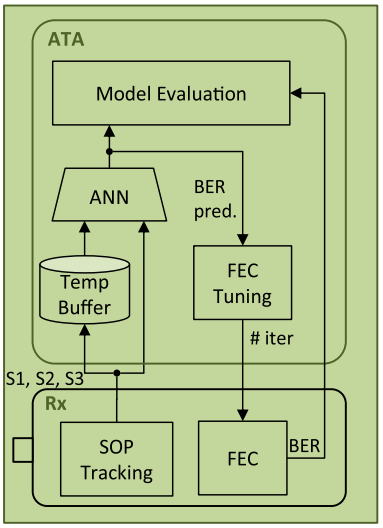

Fig. 9. In-field retraining use case.

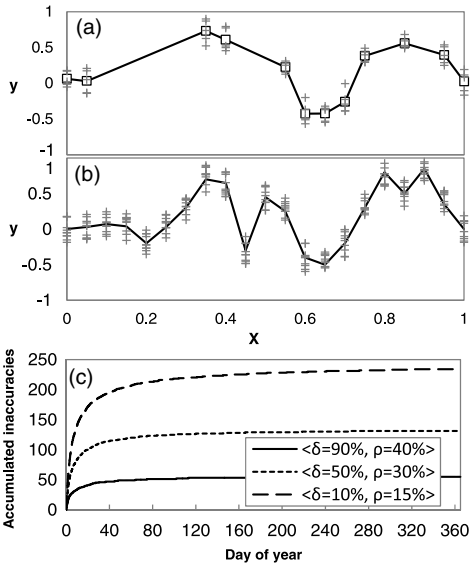

Fig. 10. (a) Initial and (b) steady regression models, and (c) accuracy evolution.

V. Illustrative Results

In this section, we first analyze the performance of retraining for regression and classification for two different general examples that capture the general characteristics of specific use cases in optical networks related to regression [9,12,18] and classification [17,20,23,24]. Next, we present illustrative results for the above-described use cases aiming at validating the feasibility of the considered ML training options.

A. Retraining for Regression and Classification

Let us start analyzing the performance of retraining for a regression use case. For simplicity, let us consider that a response variable $y \in \mathbb{R}$ is to be predicted as a function of a single feature $x \in \mathbb{R}$. A model $f(x)$ is defined in the whole x range so that $|f(x) - y| \leq \varepsilon$; however, no simple correlation (e.g., linear) between x and y can be assumed. In addition, let us assume that during the training phase only some subranges within the whole x range could be observed. Thus, when an inaccuracy for a sample x' (i.e., $|f(x') - y| > \varepsilon$) is detected, retraining must be triggered to improve the model.

For numerical evaluation, a large dataset of pairs $\langle x, y \rangle$ was synthetically generated so that x and y are highly correlated within given subranges and randomly correlated in other subranges. Next, an initial model obtained with a fraction of subranges was trained [Fig. 10(a)]. For the sake of simplicity, we assume a piece-wise linear model connecting averaged values [25]. A one year in-field operation is

emulated by randomly selecting samples from the whole data set. When inaccuracies are detected, the model is retrained with the new data, so that the model is continuously improved until reaching a steady model [Fig. 10(b)] for which no significant further improvement is needed.

For the sake of a complete study, different scenarios have been considered, according to the characteristics of the data used for training and the complexity of the relationship between x and y. To this end, let density δ be the proportion of x subranges contained in the initial training dataset and ρ be the measure of correlation defined as the cubic correlation between x and y [26] that can be used as an estimator of the relationship complexity between both variables. Figure 10(c) illustrates the accumulated number of inaccuracies detected along the operation time for three different scenarios, assuming that new samples arrive every minute. Scenario $\langle \delta = 90\%, \rho = 40\% \rangle$ mimics a situation where a realistic behavior can be likely reproduced during training and, consequently, few inaccuracies are detected in operation. In contrast, scenario $\langle \delta = 10\%, \rho = 15\% \rangle$ reproduces a more challenging situation, where most of the behavior is learned during operation. Nevertheless, as shown in Fig. 10(c), accuracy improves quickly in all the cases and the number of inaccuracies drops until reaching the steady model ($\ll$0.1% of model inaccuracies). Based on these results, we can conclude that retraining cycles allow accurate models to be obtained, speeding up ML-based algorithm deployment, whatever the characteristics of the scenario.

Let us now focus on classification using SVMs; in this case, let us consider an example where the categorical variable y with classes c_0 and c_1 is classified as a function of two features $x_1, x_2 \in \mathbb{R}$. Similar to the regression use case, we generated synthetic data according to two different scenarios: (i) the *balanced* scenario assumed that the probability of generating both classes is similar [i.e., $P(c_0) \approx P(c_0)$] and that features x_1 and x_2 can be synthetically reproduced with high likelihood, and (ii) the *unbalanced* scenario, where $P(c_0) \gg P(c_1)$, assumes that class c_1 is only partially reproducible for training through simulations and lab experiments. This unbalanced scenario mimics an anomaly detection use case, where c_0 represents the *normal* class and c_1 the anomaly [24]. Examples of initial SVM classifiers for both scenarios are illustrated in Fig. 11(a).

Three different strategies are compared: (i) *no retraining* (i.e., the SVM classifier is never updated); (ii) *periodic retraining* (i.e., retraining is triggered periodically; in this case, once a month), augmenting the training dataset with the data generated by the inaccuracies detected during the last month, if any; and (iii) *continuous retraining* (i.e., retraining is triggered every time an inaccuracy is detected). The steady-state SVM classifier after one year of operation and the continuous retraining is illustrated in Fig. 11(b). Note how the SVM classifier was strongly modified in the unbalanced scenario after processing real anomaly-like measurements. Figure 11(c) plots the evolution of the accumulated inaccuracies for one year. In light of the results, it is clear that retraining is crucial to keep the number of inaccuracies low and descending. In particular, a periodic strategy can be used to reduce the amount of retraining loops while keeping a similar performance to the continuous one. Nevertheless, in balanced scenarios, the continuous strategy allows speeding up even more, obtaining the steady ML model.

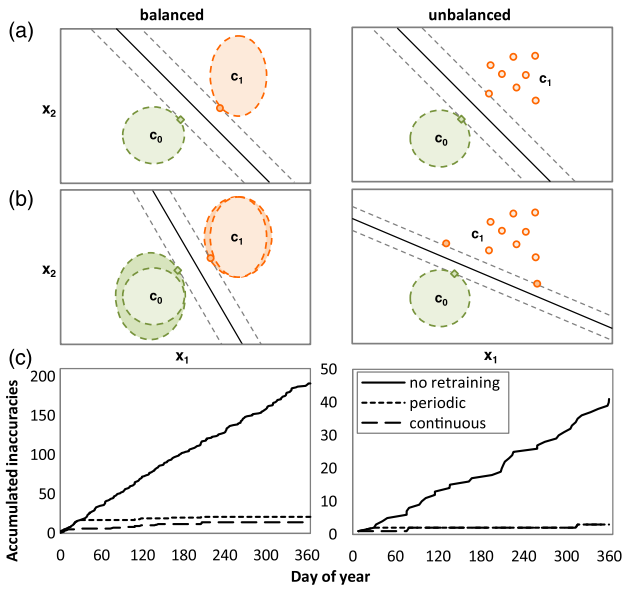

Fig. 11. (a) Initial and (b) steady training data, and SVM classifiers. (c) Accuracy evolution.

B. *Out-of-Field ML Training With In-Field Adaptation*

In this subsection, we discuss the obtained results and demonstrate how the proposed adaptation mechanism enables the residual-based approach to be applied to the optical spectrum of a signal after passing through different types of filters in the network. For the experiments, we configured a VPIphotonics scenario where a 100 Gb/s DP-QPSK modulated signal was emulated [23]. After the transmitter, the optical signal passes through eight optical nodes (from N1 to N8); after every span, an optical amplifier compensates for the accumulated attenuation of the fiber. Every optical node consists of two wavelength selective switches (WSS), each one modeled as a single optical filter with a 2nd order Gaussian transfer function for the training phase. The filter's bandwidth is set to 37.5 GHz, leaving 7.5 GHz as a guard band for the lightpath. Finally, the optical signal ends in a coherent receiver that compensates for the impairments introduced throughout the transmission. In addition, OSAs with 312.5 MHz resolution are placed after every optical node to acquire the optical spectrum of every optical link.

The efficiency of the proposed adaptation method is illustrated in Fig. 12. The residual signals of a lightpath passing through three different types of filters with a Gaussian transfer function of order 2, 3, and 4 are illustrated in Fig. 12(a). Normalization shifts the residuals so its mean is 0 [Fig. 12(b)]. Note that the differences among residuals are clearly seen at the edges, whereas they are virtually identical in the central part before and after normalization.

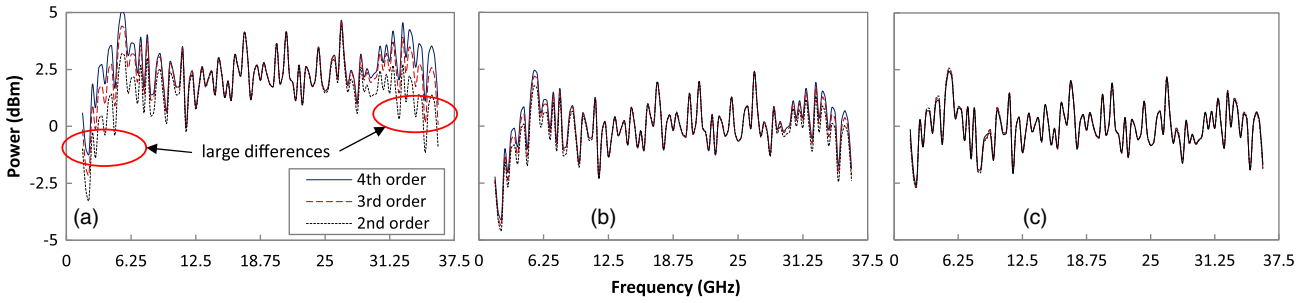

Fig. 12. (a) Unnormalized residual, (b) normalized w/o adaptation, and (c) normalized with adaptation.

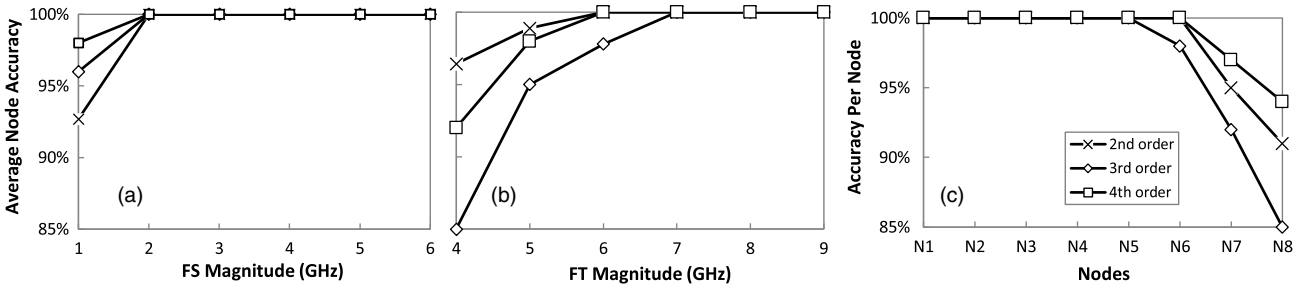

Fig. 13. Average node accuracy with respect to failure magnitudes for (a) FS and (b) FT. (c) Accuracy per node with respect to the sequence of cascaded nodes.

Adaptation focuses on compensating the effects of the different filters and the results are clearly visible at the edges [Fig. 12(c)]. Note that the most relevant parts of the residuals to detect filter-related soft failures are that of the edges. As shown, even though the signals pass through different types of filters, they result in an identical residual signal, removing the filter-dependent characteristics of the residual signal.

To emulate failure scenarios, we modified the characteristics of the 2nd WSS of every node in the setup; its bandwidth and central frequency were modified to model FT and FS failures, respectively. A large dataset of failures was collected by inducing failures of magnitude in the range [1–8] GHz for FS and in the range [1–15] GHz for FT. We configured optical filters to be 2nd order Gaussian for training and reconfigured them to become 3rd and 4th order Gaussian for testing, where the same failure scenarios were simulated.

We looked first at the benefits of applying the adaptation mechanism to identify normal cases. We found that accuracy is very poor (<20%) when no adaptation is applied and becomes perfect with residual adaptation. Next, we looked at the benefits of applying residual adaptation to detect failures. Three cases were studied: (i) 2nd order for both out-of-field training and in-field testing [note that no adaptation is needed (the case in [23])]; (ii) 3rd order; and (iii) 4th order, in which 2nd order filters were used for training, and 3rd and 4th order, respectively, with adaptation were used for testing. The results are reported in Fig. 13, where Figs. 13(a)–13(b) show the average node accuracy of identifying FS and FT, respectively, for failures in all eight nodes and varying levels of failure magnitudes. The accuracy is promising for all the cases under study, even though it degrades for very small magnitudes in which the spectrum looks like normal cases; in fact, failure detection is 100% in all cases when the failure identification step is the cause of the reduced accuracy (Table I). To highlight the impact of cascaded nodes, Fig. 13(c) presents the average accuracy for FS and FT with respect to the node where the failure occurs; failure magnitudes in the range of [1–4] GHz for FS and [4–7] GHz for FT were considered. As shown, the accuracy drops at the very last nodes because the accumulated filter cascading effects make it very challenging to distinguish between different cases.

Ultimately, the efficiency of the algorithm for transmission systems with two different filter types was evaluated. To this end, we modified the above-described setup to have 2nd order Gaussian filters in the first four nodes and 4th order Gaussian filters in the last four. As reported in Table I, failure detection accuracy is 100% while failure identification shows perfect accuracy for magnitudes above some values. Specifically, the minimum failure magnitude

TABLE I
RESULTS COMPARISON

Scenario	Failure Detection	Failure Type Identification: Min FS Magnitude	Failure Type Identification: Min FT Magnitude
Only 2nd or 4th order	100%	2 GHz	6 GHz
Only 3rd order	100%	2 GHz	7 GHz
Mix of 2nd and 4th order	100%	5 GHz	7 GHz

to be detected with 100% accuracy is 5 and 7 GHz, for FS and FT, respectively, just a bit higher than in the case of one single filter type. These results validate the performance of our residual adaptation method.

C. In-Field Retraining

In this subsection, we present the obtained results for the autonomic transmission use case. For evaluation purposes, we configured a setup with eight emulated optical nodes consisting of one node agent and one ATA with an MDA controller in the control plane. Software modules were implemented as independent Python 3.0 processes enabling multiple configurations to reproduce both individual and collective learning approaches with centralized or distributed training.

An initial training dataset with 10,000 samples from lab experiments ([18,27]) was used to train ANNs; specifically, ANNs were configured with 90 inputs (i.e., 30 last values of each Stokes parameter) and 45 hidden neurons. Then, operation started and continuously generated synthetic random samples at a rate of 278 μs (3.6 kHz), emulating real events that included some unobserved during lab experiments, causing SOP and BER fluctuation according to [18]. Figure 14(a) shows an example of SOP measurements and estimated and measured BER, where an example of the model inaccuracy can be observed. Such model inaccuracy detection triggers the learning loop.

The performance of individual and collective approaches was evaluated in terms of convergence time. As the number of ATAs significantly affect the convergence time, we started with a setup with just four of them. Figure 14(b) plots the prediction error normalized to the error of the initial trained models versus time normalized with respect to the time when all events are observed (~500 in total) in the collective approach. When inaccuracies are detected, models are improved and the prediction error decreases. Such prediction error reduction is remarkable under the collective approach, as ATA modules share knowledge among each other as soon as it is discovered; in fact, ~3.5 speed up is observed compared to the individual learning approach. This result suggests that the speed-up ratio and the number of ATAs are somehow related. To analyze such a relation, we reproduced the previous experiment and configure a number of ATAs from one to eight; the results are reported in the embedded chart inside Fig. 14(b), where an almost linear relation between speed up and the number of device agents can be observed.

Let us now evaluate distributed and centralized retraining in terms of (i) the amount of data exchanged between node agents and the MDA controller and (ii) the amount of data to be stored locally in node agents. Figure 14(c) presents accumulated data volumes at the end of executions for every ATA in the network. Under the individual learning approach, data for every detected model inaccuracy is either stored in the local node or sent to the MDA controller to augment the training dataset; moreover, model updates after every retraining are sent back to nodes in the centralized training. A slightly lower amount of data is exchanged in distributed training under the collective learning approach, as model inaccuracies are detected among all ATAs. Finally, regarding computational resources, retraining an ANN takes several minutes in a medium-sized computer (i.e., Intel Core i7-4790 with 16 GB RAM), which would convert into hours considering that computing resources in nodes are much more limited. This fact unfortunately greatly limits the applicability of distributed retraining.

Table II summarizes the analysis of the proposed alternatives and the results of the illustrative use case.

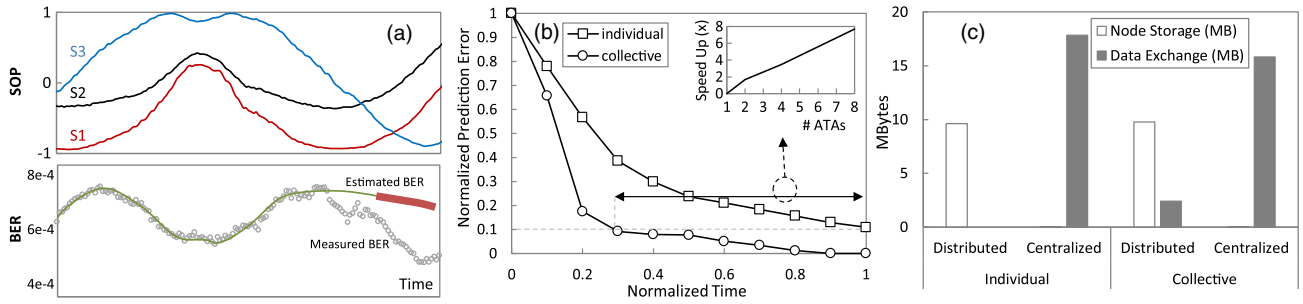

Fig. 14. (a) Inaccuracy example, (b) individual versus collective, and (c) centralized versus distributed.

TABLE II
SUMMARY OF IN-FIELD RETRAINING

		Features		Suitable Applicability Scenarios	
Learning	Training	Learning Speed	Complexity	Correlation of the Observed Patterns	Availability of CPU and Storage Resources
Individual	Distributed Centralized	Slow	Low Medium	Negligible/Low	Need extra resources at nodes Available
Collective	Distributed Centralized	Fastest	Highest High	Medium/High	Need extra resources at nodes Available

The characteristics of each approach, in terms of learning speed and complexity (elements involved, data exchange, computing and storage needs, etc.) are reviewed. Collective learning was shown to be a more suitable option compared to individual learning for those cases where a significant correlation between observations from different agents exists. In addition, a brief analysis of proper use cases and scenarios for each approach is presented. Regarding the need for computational and storage resources at the nodes, centralized training uses already available resources in the centralized MDA controller, while they are usually scarce in the node agents to support distributed training.

VI. Concluding Remarks

A learning life cycle has been proposed looking at deploying highly accurate ML models, thus implementing the *autonomic optical transmission system and networking* paradigm. The proposed framework includes the steps of the traditional ML model construction and test workflow (i.e., data generation, ML training, and model evaluation), and adds adaptation and generalization, both of which can be implemented out-of-field and in-field. The learning cycle has been designed to overcome critical obstacles, such as the lack of real measurements used for ML training or the extremely large complexity of reproducing realistic heterogeneous scenarios, thus paving the way toward smart optical networks.

Two relevant use cases have been used to show practical applications in a comprehensive way. First, out-of-field training with in-field model adaptation was proposed as a scalable option to obtain accurate models and algorithms for filter-related failure detection and classification in optical networks with heterogeneous transmission and switching devices. Second, in-field training was presented as an approach to achieve autonomic transmission for a use case to predict pre-FEC BER for intelligent receiver configuration purposes. This approach takes advantage of the capabilities available in advanced transponders to push data analytics at the subsystem level and opens the possibility to implement different in-field collaborative learning strategies.

Numerical evaluation was carried out to highlight the main benefits of the proposed learning cycle. As a general conclusion, the learning cycle was validated as an effective way to dynamically and continuously improve the accuracy of ML models for typical regression and classification purposes. Regardless of the uncertainty of the data to model and data availability for ML training during the initial out-field training, in-field retraining showed a fast convergence to accurate models.

In the out-of-field ML training with an in-field model adaptation use case, the model adaptation phase helped to achieve similar in-field detection and classification accuracy on heterogeneous networks as that observed on out-of-field homogeneous networks. As for the in-field ML retraining, the learning speed was remarkably increased by implementing collaborative learning, regardless of whether the centralized or distributed training was implemented.

Acknowledgment

The research leading to these results has received funding from the European Commission for the H2020-ICT-2016-2 METRO-HAUL project (G.A. 761727), from the AEI/FEDER TWINS project (TEC2017-90097-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

References

[1] L. Gifre, J.-L. Izquierdo-Zaragoza, M. Ruiz, and L. Velasco, "Autonomic disaggregated multilayer networking," *J. Opt. Commun. Netw.*, vol. 10, pp. 482–492, 2018.

[2] L. Velasco, L. Gifre, J.-L. Izquierdo-Zaragoza, F. Paolucci, A. P. Vela, A. Sgambelluri, M. Ruiz, and F. Cugini, "An architecture to support autonomic slice networking," *J. Lightwave Technol.*, vol. 36, pp. 135–141, 2018.

[3] D. Rafique and L. Velasco, "Machine learning for optical network automation: Overview, architecture, and applications [Invited Tutorial]," *J. Opt. Commun. Netw.*, vol. 10, pp. D126–D143, 2018.

[4] 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.

[5] F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore, "An overview on application of machine learning techniques in optical networks," *IEEE Commun. Surv. Tutorials*, to be published.

[6] Y. Zhao, B. Yan, D. Liu, Y. He, D. Wang, and J. Zhang, "SOON: Self-optimizing optical networks with machine learning," *Opt. Express*, vol. 26, pp. 28713–28726, 2018.

[7] D. Côté, "Using machine learning in communication networks [Invited]," *J. Opt. Commun. Netw.*, vol. 10, pp. D100–D109, 2018.

[8] R. Morais and J. Pedro, "Machine learning models for estimating quality of transmission in DWDM networks," *J. Opt. Commun. Netw.*, vol. 10, pp. D84–D99, 2018.

[9] F. Morales, L. Gifre, F. Paolucci, M. Ruiz, F. Cugini, P. Castoldi, and L. Velasco, "Dynamic core VNT adaptability based on predictive metro-flow traffic models," *J. Opt. Commun. Netw.*, vol. 9, pp. 1202–1211, 2017.

[10] G. Choudhury, D. Lynch, G. Thakur, and S. Tse, "Two use cases of machine learning for SDN-enabled IP/optical networks: Traffic matrix prediction and optical path performance prediction [Invited]," *J. Opt. Commun. Netw.*, vol. 10, pp. D52–D62, 2018.

[11] L. Velasco, A. Sgambelluri, R. Casellas, L. Gifre, J.-L. Izquierdo-Zaragoza, F. Fresi, F. Paolucci, R. Martínez, and E. Riccardi, "Building autonomic optical whitebox-based networks," *J. Lightwave Technol.*, vol. 36, pp. 3097–3104, 2018.

[12] Z. Wang, M. Zhang, D. Wang, C. Song, M. Liu, J. Li, L. Lou, and Z. Liu, "Failure prediction using machine learning and time series in optical network," *Opt. Express*, vol. 25, pp. 18553–18565, 2017.

[13] 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, pp. 4595–4604, 2017.

[14] R. Proietti, X. Chen, K. Zhang, G. Liu, M. Shamsabardeh, A. Castro, L. Velasco, Z. Zhu, and S. J. Ben Yoo, "Experimental demonstration of machine learning-aided QoT estimation in

multi-domain elastic optical networks with alien wavelengths," *J. Opt. Commun. Netw.*, vol. 11, pp. A1–A10, 2019.

[15] M. Ruiz, F. Coltraro, and L. Velasco, "CURSA-SQ: A methodology for service-centric traffic flow analysis," *J. Opt. Commun. Netw.*, vol. 10, pp. 773–784, 2018.

[16] L. Barletta, A. Giusti, C. Rottondi, and M. Tornatore, "QoT estimation for unestablished lighpaths using machine learning," in *Optical Fiber Communication Conf. (OFC)*, 2017.

[17] 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, pp. A27–A36, 2018.

[18] B. Shariati, F. Boitier, M. Ruiz, P. Layec, and L. Velasco, "Autonomic transmission through pre-FEC BER degradation prediction based on SOP monitoring," in *European Conf. on Optical Communication (ECOC)*, 2018.

[19] A. P. Vela, M. Ruiz, and L. Velasco, "Distributing data analytics for efficient multiple traffic anomalies detection," *Comput. Commun.*, vol. 107, pp. 1–12, 2017.

[20] B. Shariati, M. Ruiz, and L. Velasco, "Out-of-field generic ML training with in-field specific adaptation to facilitate ML deployments," in *Optical Fiber Communication Conf. (OFC)*, 2019.

[21] M. Ruiz, F. Boitier, P. Layec, and L. Velasco, "Self-learning approaches for real optical networks," in *Optical Fiber Communication Conf. (OFC)*, 2019.

[22] J. Perez, A. Gutierrez-Torre, J. L. Berral, and D. Carrera, "A resilient and distributed near real-time traffic forecasting application for Fog computing environments," *Future Gener. Comput. Syst.*, vol. 87, pp. 198–212, 2018.

[23] B. Shariati, M. Ruiz, J. Comellas, and L. Velasco, "Learning from the optical spectrum: Failure detection and identification [Invited]," *J. Lightwave Technol.*, vol. 37, pp. 433–440, 2019.

[24] X. Chen, B. Li, M. Shamsabardeh, R. Proietti, Z. Zhu, and S. J. B. Yoo, "On real-time and self-taught anomaly detection in optical networks using hybrid unsupervised/supervised learning," in *European Conf. on Optical Communication (ECOC)*, 2018.

[25] D. Montgomery, E. Peck, and G. Vining, *Introduction to Linear Regression Analysis*, 4th ed., Hoboken, NJ: Wiley, 2012.

[26] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*, 3rd ed., Abingdon, UK: Routledge, 2013.

[27] F. Boitier, V. Lemaire, J. Pesic, L. Chavarría, P. Layec, S. Bigo, and E. Dutisseuil, "Proactive fiber damage detection in real-time coherent receiver," in *European Conf. on Optical Communication (ECOC)*, 2017.