Self-Learning Approaches for Real Optical Networks

Marc Ruiz[1*], Fabien Boitier[2], Patricia Layec[2], and Luis Velasco[1]

[1] *Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain*

[2] *Nokia Bell Labs, Nozay, France*

e-mail: mruiz@ac.upc.edu

Abstract: Self-learning approaches to facilitate the deployment of ML algorithms in real networks are analyzed and their performance evaluated through an illustrative use case. Results show large benefits of collective self-learning with centralized retraining.

OCIS codes: (060.0060) Fiber optics and optical communications, (060.1155) All optical networks

1. Introduction

The revolution brought by 5G technology requires profound changes, not only in the way optical networks are built but fundamentally, in the way they are managed. Specifically, agile control and management tools need to replace typical slow operation procedures taking days to weeks to implement service deployment or network reconfiguration. In this regard, the Software Defined Networking (SDN) paradigm bringing programmability needs to be complemented with monitoring and data analytics (MDA) capabilities to enable *autonomicity* at several levels, from network to transmission [1]. Behind the autonomic concept, machine learning (ML) plays an essential role for a wide range of use cases, from self-configuration to predictive maintenance.

Contrarily to the centralized architecture of SDN, ML algorithms might be executed as close as possible to the data sources for particular use cases, e.g., when distributed MDA architectures target at minimizing the amount of data to be conveyed, as well as minimizing the response time [2] to allow control loop implementation at any level, from subsystem to network. As an example, authors in [3] proposed an autonomic transmission agent running inside transponder subsystems that enables local control loop implementation for fast device reconfiguration according to metered and forecasted data.

Although examples of autonomic networking have been experimentally demonstrated, their application in real networks entails some challenges that are not yet solved; particularly that of the availability of complete datasets that can be used for ML training, as many datasets are collected from simulation and/or lab experiments and might not cover real deployments.

In this paper, we explore *self-learning* procedures [4], where starting from an initial training dataset, ML algorithms are retrained with augmented datasets that include not yet considered patterns added as soon as they are detected. We discuss several strategies for its practical implementation in optical networks: from typical individual learning, where each agent detects new patterns from their local sources and use such them for self-learning to collaborative learning, where agents spread knowledge among them to speed-up the learning curve. Because retraining usually requires high computation capabilities, analysis of distributed and centralized options reveals their pros and cons. Self-learning alternatives are applied to an illustrative use case for autonomic transmission, where Forward Error Correction (FEC) modules in optical transponders are configured as a function of estimated pre-FEC Bit Error Rate (BER).

2. Reference architecture and key concepts

Fig. 1a presents a compact view of the distributed MDA architecture considered in this paper. Optical nodes (e.g., transponder, ROADM 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 into a closed *device-wide* control loop [3]. On top of that, *node agents* expose a single interface to the SDN controller and enable local control loops affecting several subsystems. Finally, the centralized *MDA controller* running besides the SDN controller enables network-wide autonomic operations.

Device agents should be able to compare predicted values from ML models against measured data to promptly detect those patterns for which ML models are especially inaccurate, as they can be used to augment training datasets and trigger retraining for models enhancement. Fig. 1b illustrates an example of model inaccuracy; let us assume that a ML model in the device agent produces estimations with few hundreds of ms of anticipation to be used for predictive performance analysis and proactive subsystem configuration. During certain time interval, prediction largely underestimates real measurements. Note that such inaccuracy can only be detected after real measurements, e.g. hundreds of ms later. When such model inaccuracy is detected, a self-learning loop can be triggered (Fig. 1c). Training data chunks containing monitoring data, current model error, and subsystem configuration, among others, are generated and used to feed retraining aiming at improving the accuracy of the current model and update predictive capabilities of device agents.

The research leading to these results has received funding from the EC through the METRO-HAUL project (G.A. nº 761727), from the Spanish MINECO TWINS project (AEI/FEDER TEC2017-90097-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

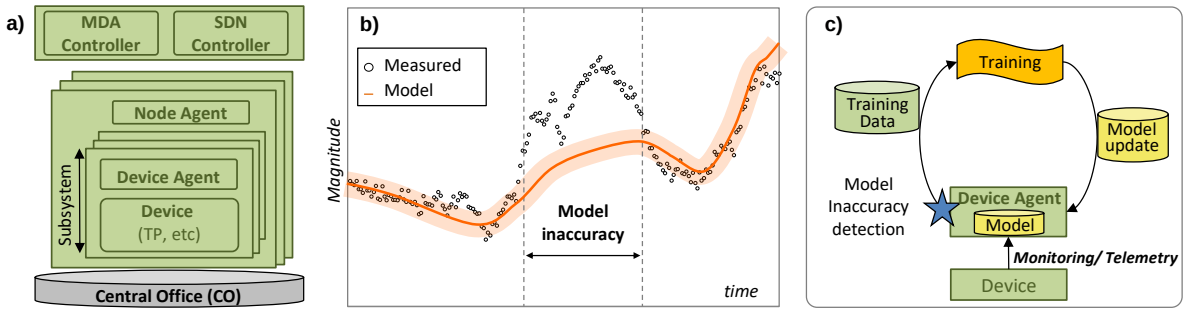

Fig. 1. Reference architecture (a), model accuracy evolution (b), and self-learning loop (c)

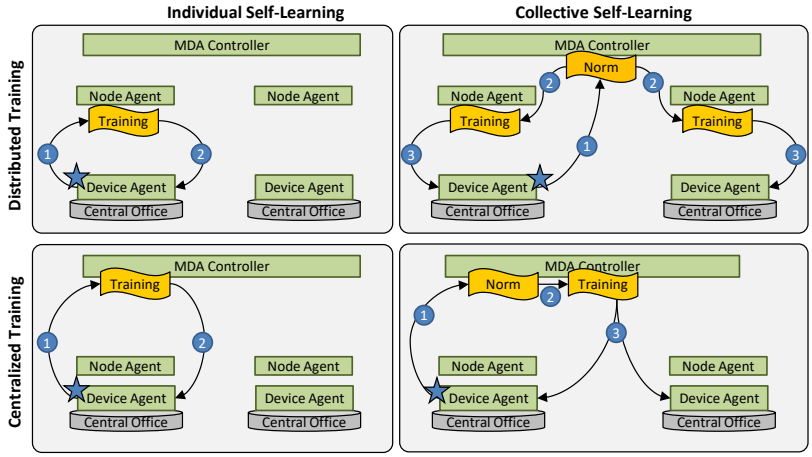

Fig. 2. Individual vs collective self-learning under centralized and distributed training

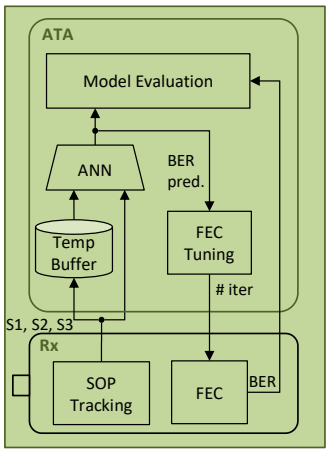

Fig. 3. Use case

3. Approaches for self-learning

As shown in Fig. 1c, self-learning can be triggered when model inaccuracies 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 approaches can be implemented. Attending to how knowledge is used, we have: *i*) *individual self-learning*, where generated knowledge is used for training and updating just the ML model of the detecting device; and *ii*) *collective self-learning*, where the generated knowledge is spread and used for training and updating the ML models of every device. This approach 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 approach entails much more complexity than individual self-learning as training data may require previous *normalization* to fit models with different characteristics. As for 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.

Fig. 2 illustrates the four how-where combinations, where labels help to identify how data flow. Individual self-learning is the most straightforward approach, where the distributed (local) training does not require any data to be conveyed to the MDA controller at the expenses of requiring extra computational resources in the node agents for ML training, whereas in the centralized training data needs to be sent to the MDA controller where more computational resources are usually available. In the case of collective self-learning with distributed training, even though training is performed locally, the MDA controller holds the role of distributing training data after normalization to node agents, as such data normalization tasks require network-wide knowledge. Finally, collective self-learning with centralized approach uses computational resources from the MDA controller for training the ML models of every device using normalized training data.

4. Illustrative use case: Autonomic Transmission

The above-described approaches are evaluated through the autonomic transmission use case (Fig. 3), where an optical receiver is dynamically configured in response to predicted short-term pre-FEC BER degradation [3]; the device agent using ML models is called *autonomic transmission agent* (ATA). These ML models use the evolution within a time-window of measured Stokes parameters representing the state of polarization (SOP) as input parameters to return the expected BER for a target short-term interval, e.g. 100 ms. As for the predictive BER model, artificial neural networks (ANN) were selected due to their inherent capability to admit complex correlation between input and output variables while adding negligible overhead to subsystem operation. The 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.

For performance evaluation purposes, we configured a setup with 8 emulated optical nodes consisting of one

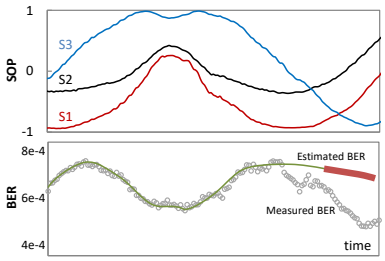

Fig. 4. Inaccuracy example

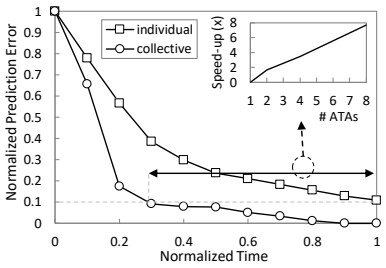

Fig. 5. Individual vs Collective

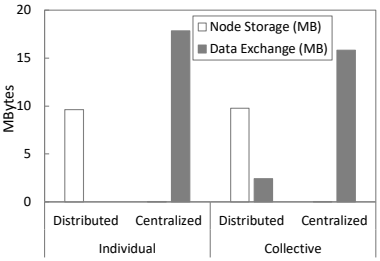

Fig. 6. Centralized vs Distributed

node agent and one ATA with a 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 self-learning approaches with centralized or distributed training.

An initial training dataset from lab experiments 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 (1/3600 sec) emulating real events including some not observed during lab experiments, causing SOP and BER fluctuation according to [3]. Fig. 4 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 self-learning loop.

The performance of individual and collective approaches was evaluated in terms of convergence time. As the number of ATAs significantly impact on the convergence time, we started with a setup with just 4 of them. Fig. 5 plots the prediction error normalized to the error of the initial trained models vs time normalized to respect the time when all events are observed (~500 in total) in the collective approach. When inaccuracies are detected, models are improved and prediction error decreases. Such prediction error reduction is remarkable under the collective approach, as ATA modules share knowledge among them as soon as it is discovered; in fact, ~x3.5 speed-up is observed compared to the individual self-learning approach. This result suggests that the speed-up ratio and the number of ATAs are somehow related. To analyze such relation, we reproduced the previous experiment and configure a number of ATAs from 1 to 8; the results are reported in the embedded chart inside Fig. 5, where an almost linear relation between speed-up and 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. Fig. 6 presents accumulated data volumes at the end of executions for every ATA in the network. Under the individual self-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. Slightly lower amount of data is exchanged in distributed training under the collective self-learning approach, as model inaccuracies are detected among all ATAs. Finally, regarding computational resources, retraining an ANN takes several minutes in a medium-size computer (i.e., Intel Core i7-4790 with 16GB RAM), which would convert into hours considering that computing resources in nodes are much more limited; this fact, greatly limits the applicability of distributed retraining.

5. Conclusions

Table 1 summarizes the analysis of the proposed approaches and the results of the illustrative use case. The characteristics of each approach, in terms of learning speed and complexity (elements involved, data exchange, computing and storage needs, etc.) are reviewed. In addition, a brief analysis of which use cases and scenarios are suitable for each approach is proposed. Specifically, collective self-learning is proposed for those cases where significant correlation between observations from different agents exists. Regarding the need of 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.

Table 1 Summary

		Features		**Suitable applicability scenarios**	
Self-learning	**Training**	**Learning Speed**	**Complexity**	**Correlation of the observed patterns**	**Availability of CPU and Storage resources**
Individual	Distributed	Slow	Low	Negligible / Low	Need extra resources at nodes
	Centralized		Medium		Available
Collective	Distributed	Fastest	Highest	Medium / High	Need extra resources at nodes
	Centralized		High		Available

References

[1] D. Rafique *et al*, "Machine Learning for Network Automation: Overview, Architecture, and Applications," IEEE/OSA JOCN, 2018.
[2] Ll. Gifre *et al*., "Autonomic Disaggregated Multilayer Networking," IEEE/OSA JOCN, 2018.
[3] B. Shariati *et al*., "Autonomic Transmission Through pre-FEC BER Degradation Prediction Based on SOP Monitoring," ECOC, 2018.
[4] J.L. Perez *et al*., "A resilient and distributed near real-time traffic forecasting application for Fog computing environments," Future Generation Computer Systems, 2018.