This paper is a postprint of a paper submitted to and accepted for publication in *ECOC 2019: 45th European Conference on Optical Communications: proceedings: Dublin, Ireland: 23-26 September, 2019* and is subject to Institution of Engineering and Technology Copyright. The copy of record is available at the IET Digital Library. DOI 10.1049/cp.2019.0985

COLLECTIVE SELF-LEARNING BY EXCHANGING ML MODELS

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

[1] Universitat Politècnica de Catalunya, Barcelona, Spain
[2] Nokia Bell Labs, Nozay, France
** lvelasco@ac.upc.edu*

Keywords: SELF-LEARNING, MACHINE LEARNING, AUTONOMIC TRANSMISSION

Abstract

Collective self-learning based on Machine Learning (ML) model sharing and combination is proposed to accelerate ML-based algorithm deployment. The considered architecture is presented, together with different alternatives for combining ML models. Performance analysis is carried out on an illustrative use case for autonomic optical transmission.

1. Introduction

Autonomic operation of optical transmission and networking requires from algorithms relying on Machine Learning (ML) models [1]. Those ML models have to be trained with datasets covering the whole features space to produce accurate ML models. However, the availability of enough data is rarely ensured, and ML training could be carried out with datasets partially covering the features space, hence reducing ML models accuracy. To facilitate ML deployment, in our previous paper in [2], we proposed to initially populate datasets for ML training and, once ML models are generated, ML retraining can be carried out to improve their precision once inaccuracies are detected. In this regard, the concept of *collective self-learning* was introduced, where agents running on top of every transmission and networking device, spread knowledge among them to speed-up the learning curve. ML models are initially trained with reduced datasets; data samples not yet considered (detected as model inaccuracies by the agents) are used to augment training datasets to consider those patterns. Those data samples are shared by the detecting agent for either centralized or distributed ML retraining thus, improving ML model accuracy. It was demonstrated that collaborative self-learning outperforms individual strategies. However, because the size of training dataset might be large to reach high-accuracy and robustness, collaborative self-learning increases both data to be exchanged among agents, as well as those stored in the nodes.

Aiming at improving *data-based* collective self-learning, in this paper, we present an alternative strategy based on sharing and combining ML models instead of data; as ML models consist of a set of parameters of moderate size compared to the size of training datasets and capture knowledge behind them, the amount of data to be shared/exchanged can be remarkably reduced. Note that this might be at the cost of adding complexity in the subsequent ML model combination process.

2. Collective Self-Learning

We propose the architecture in Fig. 1 to enable *model-based* collective self-learning based on model sharing and combination. This architecture allows a wide range of ML model combination alternatives commonly used in computer science [3].

We assume that the devices are able to measure several performance parameters and generate monitoring data [4]. Then, ML algorithms can be trained using such data; we can consider that the ML model training process can be executed either in the node agent or in the controller and once trained, ML models can be deployed to the device agents. Under some specific conditions, ML models can be shared with other agents and those models can be used together with the initial ML models aiming at increasing their accuracy; alternative approaches on how ML models are combined are presented in Section 3. ML models are key components of self-configuration processes. In particular, we can assume that such autonomous process might consist of a set of ML problems generating different outputs to a decision maker module that finds the best configuration for the underlying device. Any ML problem might require a specific procedure to combine several ML models and being able to generate their outputs. We can assume that the combination process can take place either in the node agent/controller or inside the device agent, as part of the algorithm capabilities [5].

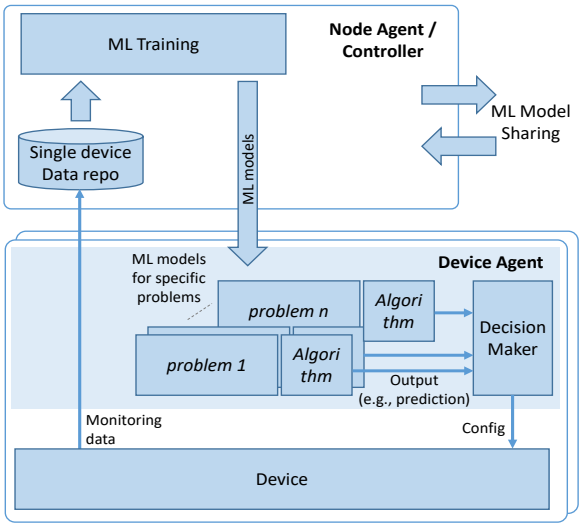

Fig. 1 Architecture for model-based collective self-learning

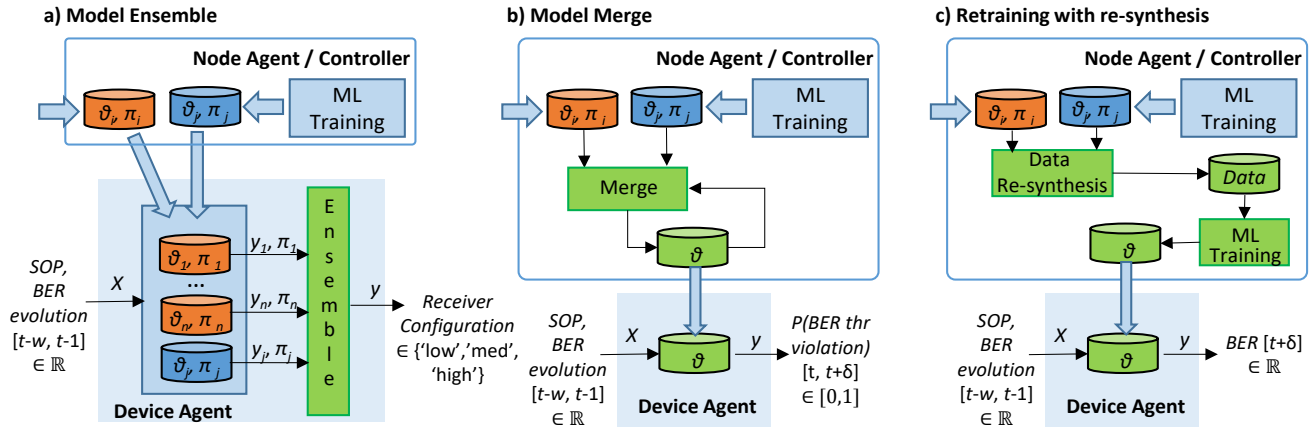

Fig. 2 ML model combination options: a) model ensemble, b) model merge, and c) model retraining.

To illustrate the co-existence of several ML algorithms working together, let us focus on a use case derived from our previous work presented in [2] under scenarios where low-resolution Analog-to-Digital converters are used [6]. Specifically, the evolution of the State-of-Polarization (SOP) (the Stoke parameters) and the pre- Forward Error Correction (FEC) Bit Error Rate (BER) in the last w time units are used for both dynamic receiver configuration and pre-FEC BER degradation anticipation in the next δ time units; three different ML-based problems need to be continuously solved:

1) a *regression* model that estimates future pre-FEC BER;
2) a *probabilistic estimator* of the chance of violating a given pre-FEC BER threshold;
3) a *classifier* to determine the proper configuration of the receiver based on the predicted class, e.g., the number of iterations of soft-decision FEC [7] to run.

3. Model combination

In this section, we will focus on three different options for model combination (see Fig. 2) that can be applied depending on the ML technique used; we consider three ML techniques Artificial Neural Networks (ANN), Support Vector Machines (SVM), and Decision Trees (DT).

For generalization purposes, let us assume that a ML model is characterized by a set of parameters ϑ (including the coefficients of the trained model, e.g., weights of an ANN). The model receives a set of input features X (SOP, pre-FEC BER evolution [t-w, t-1]) and returns the estimation of the response variable y, which can be either numerical (regression model) or categorical (classification model). Each trained ML model is shared along with additional *meta-data* π, which contains some relevant inputs required for model combination.

The first combination option, called *model ensemble* (Fig. 2a), consists in combining the prediction of several individual ML models by means of a procedure that returns one single output. Several alternatives can be considered for this procedure, e.g., a weighted average of the individual responses. The meta-data of a ML model can include the weight that need to be applied to its predictions and/or the features range observed during model training, just to mention a few.

Model ensemble is the option requiring less computational effort to apply collective self-learning, while only small additional storage capacity is required in the device for individual ML model's persistency. Moreover, it can be applied to any ML technique and even different type of ML models could be combined. On the other hand, the definition of the ensemble procedure and the weights are crucial for the sake of producing good quality predictions. In our use case, we will apply this methodology for the local receiver configuration classifier; note that in this problem, the individual ML models can be seen as weak classifiers that are combined into a strong (accurate) classifier.

The second model combination option consists in *merging* individual ML models with the aim of obtaining a single enhanced model that is used in the device agent as a single predictor (Fig. 2b). Parameters of the enhanced model are modified by the merging procedure as soon as a new individual model is available. This methodology can provide potential benefits for those cases where model parameters can be partially updated without affecting the robustness and accuracy of the non-updated part. In our case, we will use model merging in the probabilistic pre-FEC BER threshold violation estimator that it is based on a DT. In the event of a new individual model capturing a previously un-observed pattern, the merging procedure can enhance the ML model in several ways, e.g., updating probabilities of leaf nodes and/or extending a new sub-tree (branching) from a leaf node.

Finally, the third model combination option consists in generating the response from the shared individual ML models in the given features range to obtain a *synthetic* training dataset from which a new ML model is trained (Fig. 2c). The local data re-synthesis from ML models avoids exchanging large amounts of monitored data among nodes and/or to the controller. Note that some of the shared models and/or part of the synthetic data could need to be kept for future retraining cycles. In our case, we will use retraining with data re-synthesis for the ANN-based pre-FEC BER estimator.

4. Illustrative results

For performance evaluation purposes, we configured a four-node setup where optical nodes consist of one single optical receiver. Both the node agent and the device agent were implemented in Python 3.0 and multiple configurations were enabled to reproduce different collective self-learning options. For comparison purposes, *individual* self-learning, where generated knowledge is used for training and updating only the ML models of the local device, was also implemented.

Emulated receivers generate synthetic monitoring samples at a

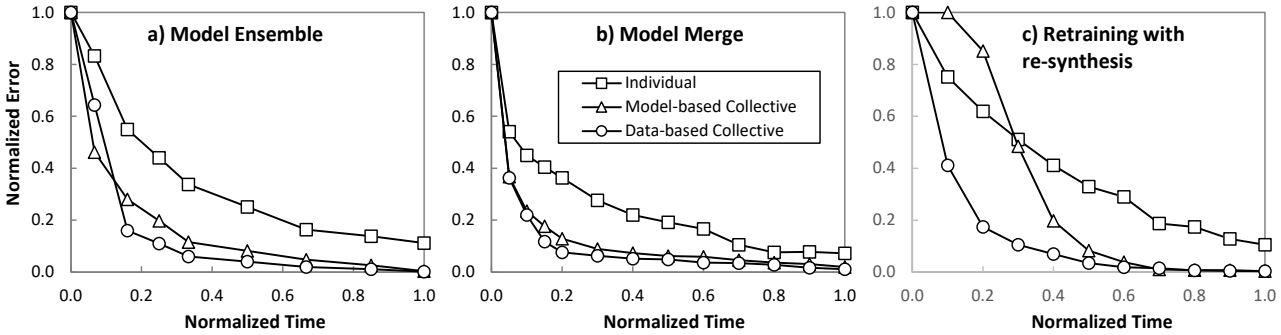

Fig. 3 Normalized prediction error vs time

rate of 278 µs (3600 samples/s). Each sample consists of: *i*) time stamp, *ii*) the values of the three Stokes parameters, and *iii*) a pre-FEC BER measurement. Based on the experimental measurements carried out in [2], realistic fiber stressing events causing correlated SOP and pre-FEC BER fluctuations were randomly generated. Regardless of the model combination option and ML problem, each ML model in the device was initially trained with 5000 samples collected from the same device. Once initial models were available, operation was emulated by sequentially generating samples, comparing the predicted response with the real one observed and, in the case of detecting inaccuracies, storing the meaningful data for further model improvement. With a given periodicity, e.g., every hour, detected inaccuracies are used to train one or more specific ML models, which are shared for model combination. For the receiver configuration classifier, we considered SVMs with 5th-degree polynomial kernels. Here, the high misclassification cost leads to consider diverse ML models and use the model ensemble option, where the number of training samples is used for weighting the combined response. For the probabilistic BER threshold violation anticipation, we considered a DT where the topology of the trees was pre-defined. Here, the merge option consisted in updating all the numerical parameters in the tree nodes by weighted averaging based on node size. Finally, for pre-FEC BER forecasting, ANNs were considered and configured with 90 inputs (i.e., last 30 values of each Stokes parameter) and 45 hidden neurons. Here, data re-synthesis was adopted, where every model was used for data re-synthesis only in the observed range of the input features; such information is shared as meta-data.

Fig. 3 shows the evolution of prediction error against the emulated operation time. For convenience, prediction error was normalized to the error of the initial trained models of the individual self-learning approach, whereas time was normalized to the time where the most accurate approach reached negligible error (<0.5%). As it can be observed, model-based collective self-learning provides an excellent performance in terms of convergence time as it clearly over-performs the individual approach and gets noticeably close to that of the data-based collective one for all the considered problems. The exception is during the early stage of operation in the case of data re-synthesis; note that as models were trained with a small data set, data synthetized from them would easily diverge from the real data, even though they are generated in the observed range. In view of this, we suggest start sharing ML models for data re-synthesis once they provide moderated accuracy.

We have demonstrated that collective self-learning performs similarly regardless of sharing data or ML models. The main benefit of model-based collective self-learning compared to the data-based one is presented in Fig. 4, where the data exchange ratio is plotted in logarithmic scale as a function of time; the ratio is computed as the amount of data exchanged under the data-based approach (i.e., ML training data) over the amount of data exchanged under the model-based one (i.e., model parameters and meta-data). As it can be observed, the model-based approach requires far less data exchange during the operation time needed to converge to accurate models (normalized error <1%). Once reached such accuracy, the data-based approach can be used to reduce even more exchanged data. The embedded table provides absolute numbers of the estimated size of an example data exchange during the very first stage of operation of a single device. By scaling these figures to a deployment with hundreds of devices exchanging data, one can realize that the model-based collective self-learning approach reduces the requirements of the control plane communication network.

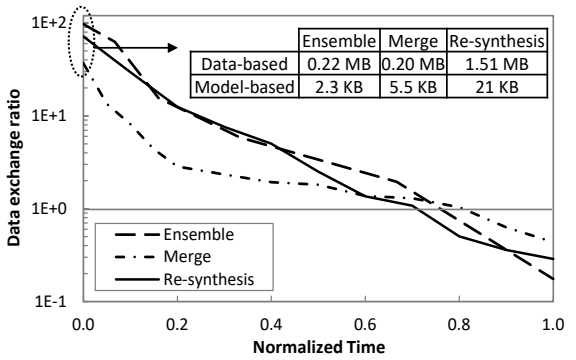

Fig. 4 Data exchanged comparison.

5. Conclusions

Collective self-learning based on ML model sharing have been proposed and different model combination options have been explored (ensemble, merge, and re-training with data re-synthesis). A use case for autonomic transmission have been used for evaluation purposes, showing that they reach similar performance with much better scalability than a collective approach based on training data sharing.

6. Acknowledgements

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

7. References

[1] D. Rafique and L. Velasco: 'Machine Learning for Network Automation: Overview, Architecture, and Applications,' IEEE/OSA Journal of Optical Communications and Networking (JOCN), 2018, 10, (10), pp. D126-D143.

[2] L. Velasco, B. Shariati, F. Boitier, P. Layec, and M. Ruiz: 'A Learning Life-Cycle to Speed-up Autonomic Optical Transmission and Networking Adoption,' IEEE/OSA Journal of Optical Communications and Networking (JOCN), 2019, 11, (5), pp. 226-237.

[3] Bishop, C.: 'Combining Models', in Bishop, C.: 'Pattern Recognition and Machine Learning' (Springer Verlag, 2006, 1st edn.), pp. 653-676.

[4] N. Sambo, A. Giorgetti, F. Cugini, P. Castoldi: 'Sliceable transponders: pre-programmed OAM, control, and management,' IEEE/OSA Journal of Lightwave Technology (JLT), 2018, 36, (7), pp. 1403-1410.

[5] L. Velasco et al: '"Building Autonomic Optical Whitebox-Based Networks,' IEEE/OSA Journal of Lightwave Technology (JLT), 2018, 36, (15), pp. 3097-3104.

[6] X. Chen, S. Chandrasekhar, S. Randel, W. Gu, and P. Winzer: 'Experimental Quantification of Implementation Penalties from Limited ADC Resolution for Nyquist Shaped Higher-Order QAM,' Proc. Optical Fiber Communications Conference (OFC), Anaheim, CA, 2016.

[7] C. Dorize, O. Rival, and C. Costantini: 'Power scaling of LDPC decoder stage in long haul networks,' Proc. International Conference on Photonics in Switching (PS), Ajaccio, France, 2012.