Using Machine Learning in Communication Networks [Invited]

David Côté

***Abstract*—In this paper, we first review how the main machine learning concepts can apply to communication networks. Then we present results from a concrete application using unsupervised machine learning in a real network. We show how the application can detect anomalies at multiple network layers, including the optical layer, how it can be trained to anticipate anomalies before they become a problem, and how it can be trained to identify the root cause of each anomaly. Finally, we elaborate on the importance of this work and speculate about the future of intelligent adaptive networks.**

***Index Terms*—Communication networks; Cross layer design; Machine learning; Prediction methods; Software defined networking.**

I. Introduction

The ability of artificial intelligence (AI) systems to acquire their own knowledge by extracting patterns from raw data is known as machine learning (ML) [1–4]. Rooted in classical linear algebra [5] and probability theory [6], this technology has been proven to work for a growing number of tasks, ranging from image recognition to natural language processing and others [4]. ML is particularly powerful in the presence of massive amounts of data, often called "big data." Increasingly large datasets enable increasingly accurate learning during the *training* of ML; these datasets can no longer be grasped by the human eye, but can be scanned by computers running ML algorithms. Searches for subatomic particles from the early 2000s [7] and 2010s [8,9] are good examples of this ML + Big Data synergy. Today, *deep* learning [10] is widely recognized as state of the art in the field. In 2012, a prestigious image recognition competition (ILSVRC, a.k.a. "ImageNet") was won by a large margin using a deep convolutional neural network [11]. In 2015, superhuman performance was achieved for the first time at the same competition [12]. In 2017, AlphaGo Zero achieved superhuman performance at the ancient Chinese game of Go, without any human input, using deep reinforcement learning [13]. Last but not least, the usage of ML applications in the past decade has been highly profitable for technology companies like Google, Microsoft, Facebook, IBM, Baidu, Apple, Netflix, and others. Hence, it is now established that ML not only works technically, but is also valuable economically.

There is no doubt that ML technologies can be applied to communication networks with equal success. Optical networks typically contain thousands of network elements (NEs). This number gets much larger for packet, IP, mobile, or "Internet of Things" networks. These NEs produce large amounts of data that can be consumed by ML. Furthermore, multi-layer, multi-vendor telecommunications networks rapidly get very complex. This complexity can be managed with ML applications.

At present, usage of ML in communication networks is in its infancy. This might be because a fairly substantial infrastructure must be put in place before network data can be effectively consumed by ML applications. Furthermore, the development of ML infrastructure and applications requires a new mix of multidisciplinary skills not necessarily present across the industry so far. Nevertheless, it is noticeable that network service providers (AT&T, Verizon, etc.), private network operators (Google, Facebook, etc.), and network equipment vendors (Ciena, Cisco, etc.) are all starting to invest in ML initiatives targeting communication networks.

Compared to classic algorithms based on subject matter expertise, ML is attractive because it produces highly reusable and automatable software, is often easier to implement, and can yield better performance. However, subject matter expertise is still required to prepare the input data and interpret the output insights of concrete ML applications.

In this paper, we will review how the main concepts of ML can be applied to communication networks in Section II. We will present results from a concrete machine learning application on a real network in Section III. Finally, we will elaborate on the importance of this work and speculate about the future of intelligent adaptive networks in Section IV.

II. Machine Learning Tasks Applied to Communication Networks

In this section, we review how the main concepts of ML can apply to communication networks.

The first element is the data itself. Data is the fundamental source of information on which the entire ML stack depends. Next are the different algorithms that can be used

Manuscript received March 13, 2018; revised July 3, 2018; accepted July 4, 2018; published August 6, 2018 (Doc. ID 325933).

D. Côté (e-mail: dcote@ciena.com) is with Ciena Corporation, Ottawa, Ontario K2K 0L1, Canada.

https://doi.org/10.1364/JOCN.10.00D100

to extract (or learn) the relevant information from raw data (provided all the required infrastructure is in place). Last are the applications that leverage this information to solve concrete problems and provide added value.

A. Data

Many data sources can be exploited to get information about every network component, from the physical (or virtual) devices to the communication channels, usage patterns, environment, and business context.

Network devices generate performance monitoring (PM), alarms, and/or logging data. These include items like power levels; error counters; received, transmitted, or dropped packets; CPU utilization; geographic coordinates; threshold crossings; and many others. Communication channels (or "services") also generate PM data for all layers of the open systems interconnection model [14]. For example, the performance of the optical layer is characterized by optical signal-to-noise ratio, chromatic dispersion, polarization-mode dispersion, transmit and received power, and bit error rate. The performance of the IP layer is characterized by bandwidth, throughput, latency, jitter, and error rate. End-users and environmental or business data typically come from third-party proprietary databases.

Every time any of the just-described data is collected, it is useful to record a timestamp associated with it. Time is especially important because it can be used to correlate independent data sources. Data collected during a common time interval may be associated to define a "snapshot". Furthermore, sorting data in chronological order is frequently used to measure time-series trends.

Most communication networks connect to many different device types, and different types of devices from different equipment vendors tend to produce different data in different formats. Therefore, communication networks tend to generate a wide variety of data. In addition, the frequency at which the data is collected (a.k.a. velocity) can vary for each source. Likewise, the amount of time during which the data is kept in storage can also vary. When networks contain many devices and services, with high-frequency data collection and/or long storage periods, the result is large data volumes. The combination of variety, velocity, and volume is often referred as "Big Data".

When the proper infrastructure is set up, a common approach is to collect and store all available data, and enable *ad hoc* analysis after the fact. When this is not possible, tradeoffs must be made to filter a fraction of the data and only keep what is most valuable for specific applications. In general, wider variety, larger velocity, and larger volumes of data will broaden the coverage and increase the accuracy of ML-driven applications. For example, it has been shown that state of polarization "transients" can be understood more accurately by augmenting traditional optical network parameters with weather data [15].

So far, we have discussed the collection of raw data, which can be easily automated to produce "Big" datasets. However, the situation is different for collecting labels. As discussed in sub-section B, labels are required to train supervised ML algorithms. They are also very useful to test the performance of trained ML models. Labels qualify the true nature of the system under study. They differ from raw telemetry data because they require subject matter expertise to provide a higher-level interpretation. For instance, a label could indicate the state of a network device ("normal state", "abnormal state", etc.) at the time its PM data was collected. The collection of labels from live systems tends to be hard to automate. This is not surprising since the label information is what we typically want machines to learn in the end. As a result, labels tend to suffer from lack of statistics, which can be a bottleneck for ML algorithms upstream. Sources of label data include simulated, control, and special samples prepared by subject matter experts [16].

B. Algorithms

In this sub-section, we will first review the key concept of model training. Then we will focus on selected ML tasks that are particularly useful for networking applications (for a comprehensive review of all ML tasks, see Refs. [1–4]). We will start with tasks that can be performed as "read-only" operations on the network, namely regression, classification, and anomaly detection. We will discuss how each task can be implemented with different algorithms, and how machines can learn how to act on a network in the context of closed-loop software-defined networking (SDN) automation.

The development of increasingly better ML algorithms is a very active research topic, both in academia and in industry. However, it is worth pointing out that many problems can be solved with well-known data science techniques today. Indeed, most of the algorithms discussed in this paper are readily available in popular ML frameworks like SciPy, SciKitLearn, Keras, TensorFlow, Torch, R, ROOT, Spark MLlib, and others [17–24].

1) Model Training: In ML, the process of learning from data is sometimes called "training". ML algorithms can be divided into two broad categories, supervised and unsupervised learning, depending on how their training is performed.

Unsupervised ML involves three components: a dataset $\mathbf{X}$, a model $M(\boldsymbol{x},\theta)$, and a cost function $C(\boldsymbol{x},M(\boldsymbol{x},\theta))$. The vector $\boldsymbol{x}$ represents a "snapshot" of the system under study. For example, $\boldsymbol{x}$ can contain PM data from a network device at a given time. Then, the dataset $\mathbf{X}$ would be a list of "snapshots" collected at multiple times (in mathematical terms, $\mathbf{X}$ is a vector of vectors also known as a *tensor*). The model aims to represent the true probability distribution $P(\boldsymbol{x})$; it depends on parameters $\boldsymbol{\theta}$ whose values are unknown *a priori* but can be learned from data. The learning itself consists of finding the values $\boldsymbol{\theta}^*$ that minimize the cost function for the entire dataset $\mathbf{X}$:

$$M(\boldsymbol{x},\boldsymbol{\theta}) \xrightarrow{\text{training}} M(\boldsymbol{x},\boldsymbol{\theta}^*), \tag{1}$$

$$\boldsymbol{\theta}^* = \text{argmin}_{x \in X} C(\boldsymbol{x}, M(\boldsymbol{x}, \boldsymbol{\theta})). \quad (2)$$

A common way to implement Eq. (2) is the gradient descent method with the squared loss $(M(\boldsymbol{x}, \boldsymbol{\theta}) - y)^2$ cost function, though several other functions can also be used (see Refs. [1–4] for a comprehensive discussion). After that, we say that the ML model has been trained. In principle, the trained model $M(\boldsymbol{x}, \theta^*)$ provides the best estimate of the true $P(\boldsymbol{x})$ given the amount of information in $\mathbf{X}$. To improve further, we can add training data (i.e., extend $\mathbf{X}$) such that:

$$\lim_{|X| \to \infty} M(\boldsymbol{x}, \boldsymbol{\theta}^*) \approx P(\boldsymbol{x}). \quad (3)$$

Note that Eq. (3) only applies if the model M is appropriate for the dataset $\mathbf{X}$. If this is not the case, the accuracy of M will saturate and one should consider changing to a different model $M'(\boldsymbol{x}, \boldsymbol{\theta}')$.

For supervised ML, an additional form of data—the label—indicates the true nature of the system under study. This turns a raw dataset $\mathbf{X}$ into a labeled dataset $\mathbf{X}_y$ where y represents the label(s) associated with each $\boldsymbol{x}$. The additional label information can be leveraged in the cost function $C'(y, \boldsymbol{x}, M(\boldsymbol{x}, \boldsymbol{\theta}))$. The minimization of C' will favor parameters that generate the correct answer for y. Thus, in supervised ML, the machine can learn to predict labels y from $\boldsymbol{x}$, such that:

$$\lim_{|X| \to \infty} M(\boldsymbol{x}, \boldsymbol{\theta}^*) \approx P(y|\boldsymbol{x}). \quad (4)$$

For example, labels can tell the true state of a network device (“normal state”, “abnormal state”, etc.) at the time its PM data was collected. and supervised ML can learn to identify devices in an abnormal state from their raw PM data.

A very useful property of supervised ML is its ability to measure accuracy in a reliable way. This is done by splitting the labeled dataset in (at least) two independent parts: $\mathbf{X}_y^{\text{train}}$ and $\mathbf{X}_y^{\text{test}}$. The model is trained using $\mathbf{X}_y^{\text{train}}$ only, and the properties of the trained model can be benchmarked on $\mathbf{X}_y^{\text{test}}$. By doing so, each prediction of $M(\boldsymbol{x}, \boldsymbol{\theta}^*)$ can be compared to the “truth” provided by the labels in $\mathbf{X}_y^{\text{test}}$. For a binary classifier trained to separate signal (S) from background (B), for example, the confusion matrix can be derived from the measured rates of true positive (S event correctly identified as S), false positive (B event mistakenly identified as S), true negative (B event correctly identified as B), and false negative (S event mistakenly identified as B) [25]. This guarantees that the test results will be unbiased because $\mathbf{X}_y^{\text{test}}$ is statistically independent from $\mathbf{X}_y^{\text{train}}$; it also guarantees that $\mathbf{X}_y^{\text{test}}$ is a representative control sample because it derives from the original sample $\mathbf{X}_y$.

A concrete example of this procedure—implemented with the Ciena Network Health Predictor application [26]—is shown in Figs. 1 and 2. Figure 1 shows the output of a supervised Random Forest Regression algorithm [27] on $\mathbf{X}_y^{\text{test}}$ for two labels, “normal” and “abnormal”. The output distribution is continuous between 0 and 1, with “normal” outputs towards 0 and “abnormal” outputs towards 1. This can be turned into a binary classifier with a cutoff threshold illustrated by the dashed red line. In turn, the performance of the classifier can be characterized by a receiver operating characteristic (ROC) curve [25] illustrated by Fig. 2. An ROC curve plots True Positive versus False Positive rates at various cutoff threshold settings. Greater area under the curve indicates better performance of the classifier. In the ROC graph of Fig. 2, True Positive corresponds to correctly identifying a network element as being in the “Abnormal” state. The purple dot in Fig. 2 corresponds to a threshold of 0.3 illustrated in Fig. 1. Through this process, it is possible to adjust the threshold on the supervised ML output—to tune the algorithm—in order to achieve a target performance point on the ROC curve. This can be useful because different use cases will require different tradeoffs between true positive efficiency and false positive noise.

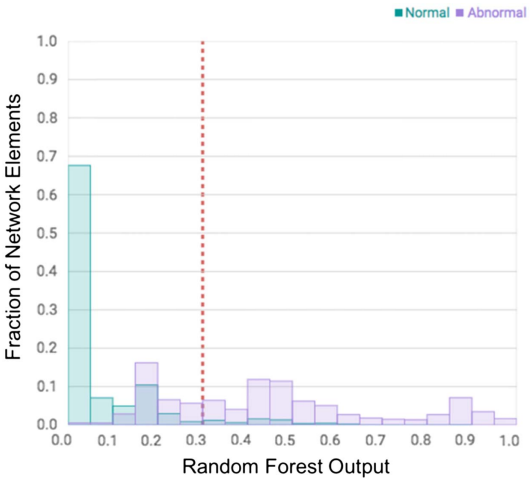

Fig. 1. Example of test distribution from a supervised ML algorithm (random forest regression).

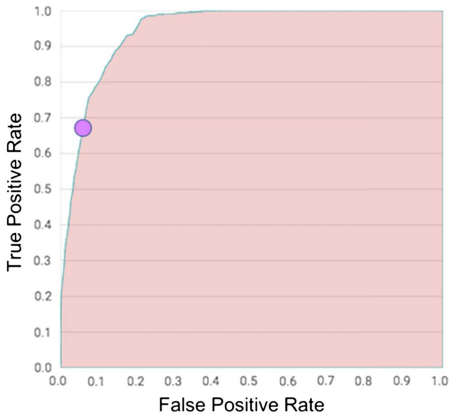

Fig. 2. Example of ROC curve derived from the test results shown in Fig. 1. The purple dot corresponds to the dashed red line in Fig. 1.

One important drawback of supervised ML, however, is that labeled datasets are often difficult to obtain in practice (see sub-section A). In particular, raw telemetry data from

communication networks is usually not labeled. Therefore, it is often necessary to use unsupervised algorithms in concrete networking applications. Hybrid approaches such as semi-supervised learning [28], multi-instance learning [11], or one-shot learning [29] can be attempted to overcome this difficulty.

In the remainder of this section, we discuss how these ML tasks (regression, classification, and anomaly detection) can be implemented with different algorithms.

2) Regression—Data-Driven Network Optimizations: During a regression, the task is to predict a numerical value given some input [4]. A straightforward application is to solve optimization problems such as network planning. Defining a model $M(\boldsymbol{x}, \theta)$ in which θ represents classic network parameters (topology, bandwidth, etc.) and $\boldsymbol{x}$ represents historical data (traffic patterns, business data, etc.), one can find the best network configuration that will satisfy constraints from all inputs by solving Eq. (2). This is a simple but potentially powerful extension of traditional network optimization based solely on network parameters ($M(\theta)$, ignoring $\boldsymbol{x}$).

3) Regression—Prediction of Future Events from Trends: Another use case for regression is to measure time-series trends and predict future events. This type of algorithm is commonly used for algorithmic trading, for example, but can also be used in networking. For instance, it can anticipate network capacity exhaust, bandwidth congestion, or device failures. Here again, this boils down to solving Eq. (2) in the special case where time is one of the inputs in $\boldsymbol{x}$. A variety of models can be used for time-series regression, ranging from simple polynomials to deep recurrent neural networks and many others [1–4].

4) Classification of Network Events: Solving classification tasks for the sake of image recognition is among the hallmarks of ML [11,12]. By using network "snapshots" instead of photographs, this type of algorithm can be used to characterize network properties or recognize patterns associated with different types of network issues.

ML classifiers are usually implemented as supervised learning, using the "predict labels y from $\boldsymbol{x}$" formalism of Eq. (4). Different approaches are commonly used for this task [1–4]. The simplest approach is linear discriminant [30]. A middle-ground approach is decision trees [30], boosted decision trees, or random forest [27,30]. The most advanced approach is artificial neural networks (ANN) [4,30]. Compared to linear discriminants, random forest and ANN typically provide better results since they account for nonlinear effects. Compared to ANN, random forest is easier to use, easier to interpret, and faster to train. However, for complex problems, ANN will usually provide the best results.

It is much less common and rather difficult to implement classifiers as unsupervised ML. Nevertheless, researchers from Stanford and Google have reported that this can be achieved with deep auto-encoders [31].

To classify among multiple types of labels, one can either implement multiple binary classifiers that will each return the probability of one label, given $\boldsymbol{x}$, or one can use a single classifier that will compute the probabilities of all labels at once. The latter is commonly implemented using a "SoftMax" function [4].

5) Detection of Network Anomalies: In anomaly detection, the task is to identify abnormal or atypical elements $\boldsymbol{X}_i$ among a dataset $\boldsymbol{X}$. This can be useful in flagging elements that require further attention from a stream of events generated by a live network.

If labeled datasets are available, anomaly detection can be seen as a simple type of classification with supervised ML, and can be implemented as a binary classifier with only one label ("normal" or "not normal").

However, anomaly detection is unique in the ways it can be implemented, with unsupervised ML consuming raw input data. This makes it relatively easy to deploy in practice.

For example, as will be discussed further in Section III, one can collect PM data from a production network and build a Likelihood model that characterizes its properties during normal conditions. One can then use this model to estimate the probability $P(\boldsymbol{x})$ that a network producing new incoming data is actually in normal conditions [see Eq. (3)]. Low probabilities can be flagged as "anomalies".

See Ref. [32] for a comprehensive review of all anomaly-detection methods.

6) Learning to Take Actions on the Network: So far, we have discussed supervised and unsupervised ML algorithms that can produce insights from "read-only" datasets. As will be seen in sub-section D, these insights can be leveraged by prescriptive applications that can act on the network automatically if rules mapping insights to actions are provided by the end users. However, the rules themselves cannot be learned with this approach. This can be addressed by a different type of algorithm, Reinforcement Learning (RL), that can learn how to interact with an environment through a feedback loop [33,34].

In RL, the learning happens by minimizing a cost function $C(\boldsymbol{a}, \boldsymbol{x}, S(\boldsymbol{x}))$, where $\boldsymbol{a}$ is a list of actions that can influence an environment modeled by $S(\boldsymbol{x})$. This is similar to Eq. (2), with notable differences that now: 1) the algorithm learns the best actions given a state (instead of the best parameters θ given a dataset $\mathbf{X}$), and 2) minimization of the cost function happens dynamically after each action (instead of being done once for a given $\mathbf{X}$).

RL is famous for achieving superhuman performance at Go [13], but also at Atari video games [35]. The analogy between communication networks and video games is imperfect but interesting. Both can be seen as a closed system with a finite list of possible actions, to a first approximation. In Atari games, actions include moving the joystick and pressing buttons; the state is derived from pixel images and the cost function is the score. In networks, actions can include operations such as adjusting bandwidth profiles or re-routing traffic; the state is derived from network "snapshots" ($\boldsymbol{x}$), and the cost function can be derived from PMs such as throughput, latency, or error rate. Hence, it is

conceivable that RL could achieve superhuman performance at driving networks in the future.

A number of challenges remain to be solved before RL can be routinely applied to production networks. These include demonstrating that RL algorithms can scale to handle large and complex production networks as well as limiting the impact of inefficient actions during the early exploration phase of RL training.

Nevertheless, a prototype RL application has been demonstrated on a real network by Ciena at the conference *Supercomputing 2017* [36]. In this demonstration, several Layer 2 (ethernet) services with different traffic patterns and priorities were managed by a RL algorithm. The services were competing for bandwidth on an over-booked link. The RL application sampled the network *state* every 5 s and took *actions* on the network every 20 s. After each action, the RL application received a *reward*. The state included the received and transmitted traffic of the live network, as well as the committed information rate of each service. The possible actions were to increase or decrease the bandwidth allocated to each service, and the reward was a function of the traffic throughput and dropped packets, weighted by the priority of each service. As a result, the application was able to learn how to adjust bandwidth profiles of competing services in an optimal way, in spite of continuously changing traffic conditions, by maximizing the reward over time.

C. Infrastructure

In the simplest case, ML applications are hosted on a single computer with regular data storage and central processing units (CPUs), assuming software is available to collect raw data and transform it into a format consumable by ML algorithms. This basic setup is sufficient to process small data sets in non-production environments.

To use deep-learning algorithms, it is generally required to accelerate computations with specialized hardware such as graphics or tensor processing units.

To exploit synergies of ML with Big Data (see Section I), more infrastructure is required to handle the large variety, volume, and/or velocity of the "Big" data. Wide variety requires an abstraction layer between the raw inputs from many sources and the ML algorithms (this is sometimes called a "Resource Adapter"). Large data volumes require distributed storage and parallel computing on a computer cluster (this is sometimes called a "data lake" or a "cloud"). Furthermore, it requires an efficient mechanism to read back and process batches of data. This is commonly achieved with the software tools Apache Hadoop [37] and Apache Spark [24]. Finally, fast velocity requires data-streaming capabilities. This can be achieved by adding tools such as Apache Kafka [38] to the Hadoop/Spark cluster.

Figure 3 illustrates the full infrastructure to support ML-driven applications for communication networks. The solid lines are required for read-only applications.

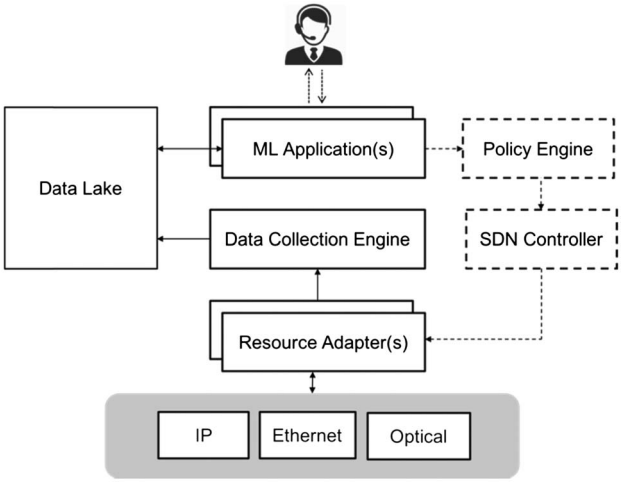

Fig. 3. Infrastructure required to fully support ML applications in communication networks. It includes: a data collection engine, a data lake, a series of ML applications, and an SDN-aware network. The policy engine is an optional interface to abstract the insights reported by the applications from the resulting actions on the network.

The additional dashed lines are required for closed-loop automation.

D. Applications

After the aforementioned data, algorithms, and infrastructure are all in place, many potential ML applications become possible for the telecommunications industry. These can be categorized as descriptive, predictive, and prescriptive.

Descriptive applications are the simplest. They include such items as analytics dashboards and interactive data-mining tools, which are generally included with this infrastructure. Despite their simplicity, these applications enable an unprecedented view of the "big picture" for large and complex networks. Furthermore, they open the door to agile data exploration of diverse data sources that could not be looked at simultaneously and combined before.

Predictive applications only require "read only" access to network data, but leverage arbitrarily sophisticated ML to extract impactful insights. These range from network security and fraud detection to network level and service level assurance, pre-emptive maintenance, troubleshooting assistance, root cause analysis, or network design optimization and planning [39–41]. ML applications have the potential to reduce the cost of network operations amid an unprecedented time of increased complexity. They can also improve end-user experience and create new revenue opportunities for network service providers. The potential for innovation is particularly interesting when feeding ML applications with inputs that were historically separate from each other but can now be accessed from the same data lake. For example, ML can quantify the risk of customer churn by combining network health and service-level data with end-user and business data [40].

Prescriptive applications require a closed feedback loop and SDN automation (see Fig. 3). They enable what can be described as an "adaptive network" [42] or a "self-healing and self-learning network fueled by artificial intelligence" [41]. Their use cases are similar to the predictive applications described, except that ML insights can now be applied to the network in near-real time and provide maximum operational efficiency. However, this requires confidence that the ML insights are indeed reliable. Therefore, predictive applications must gain market acceptance first, before prescriptive applications can be commonly deployed in production. During the transition period from predictive to prescriptive, ML applications can run in a hybrid mode in which their recommendations are first approved by a human operator before they are automatically applied in the network.

III. Case Study: Detection of Abnormal Elements in a Multi-Layer Network Using Unsupervised ML

In this section, we present results from a concrete application developed and tested on a real network at Ciena. We show that the application can detect anomalies at each layer of a multi-layer network, including the optical layer, using unsupervised machine learning. Furthermore, it can anticipate anomalies before they become problems and can be trained to identify the root cause of each anomaly.

This demonstrates that using ML for communication network applications is not just a theoretical possibility, but in fact is achieved in practice today.

A. Setup

To obtain these results, we have used the multi-layer network illustrated in Fig. 4. Two Ciena 6500 nodes are connected with an optical link, each containing Layer 0 [amplifiers, optical supervisory channel (OSC)], Layer 1 [100-gigabit optical transport module (OTM4)], and Layer 2 [FLEX gigabit ethernet (GE)] cards. There is a light attenuator in between the two nodes. The traffic is generated by a test set sending ethernet packets. We have purposely used a simple network configuration to make the interpretation of our results straightforward.

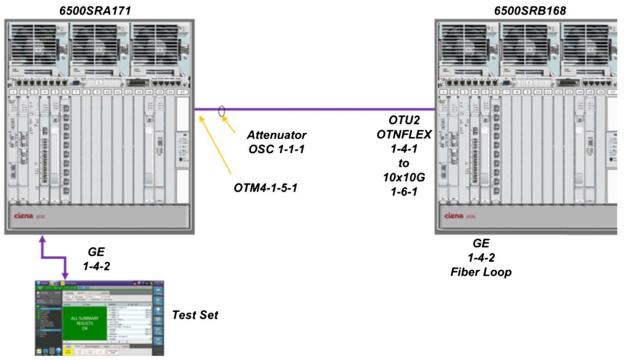

Fig. 4. Multilayer network used to test detection of abnormal network elements.

The ML application—known as network health predictor (NHP)—is executed with the Blue Planet Analytics (BPA) software platform [43]. The BPA platform is itself connected to a Hadoop cluster hosted in a Ciena private cloud, similar to the architecture shown in Fig. 3.

B. Methodology

In this demonstration, we reproduce what could happen in a production network over several days or weeks, but within an "accelerated" time span. The BPA software pulls PM data from each 6500 card every 10 s, using unbinned TL1 counters (instead of using 15 min binned data, which is more common). This data is transformed on the fly from its raw format to the NHP schema, using Spark-streaming [24] pipelines, before being written to the Hadoop distributed file system (HDFS) [37]. The location of the data on the HDFS is tracked by an entry in the dataset catalog of the BPA platform.

First, we collect data for a few minutes while the network operations are normal. Then we feed this "normal conditions" dataset to NHP to build an unsupervised ML model of this data. NHP does it by 1) building the one-dimensional probability density function (PDF) of each PM of each network card type, and 2) combining all the relevant PDFs into a global likelihood. This characterizes the network properties under normal conditions.

From this point on, we begin a so-called "recurring NHP analysis" that examines new incoming data every 5 min, with a 5 min sliding window. Here again, this is an "accelerated time" version of NHP. In production, we would typically re-analyze new incoming data every few hours using a sliding window of several days. Each Ciena 6500 port is analyzed independently. The data used for this analysis are listed in Table I.

TABLE I
PM Data Used in This Demonstration

- **Layer 0**
  - Optical Channel (OCH)
    - OPR-OCH: Optical Power Received
    - SPANLOSS-OCH: power loss through a span
- **Layer 1**
  - Optical Transponder Unit (OTU)
    - QAVG-OTU: average value of bit error rate (log)
    - QSTDEV-OTU: standard deviation of bit error rate (log)
- **Layer 2**
  - Physical Coding Sub-layer (PCS)
    - CV-PCS: number of code violations
    - ES-PCS: number of errored seconds
    - UAS-PCS: number of unavailable seconds
  - Ethernet (E)
    - INFRAMESERR-E: number of input frame errors
    - ES-E: number of errored seconds
    - UAS-E: number of unavailable seconds

For a given card and timestamp, the NHP analysis uses a frequentist inference methodology inspired by particle physics (see, e.g., Refs. [7,8,16]). First it compares a vector of incoming PM values from the live network with their expected values from the Likelihood model. Then it derives a probability that such values could be obtained under normal conditions (a.k.a. "p value").

This process is repeated for every timestamp and the results are sorted in chronological order so as to build a graph of "probability of being normal" (y axis) versus time (x axis). A regression algorithm is executed on the graph to measure the trend versus time for this port.

Finally, a risk factor ranging from 0 (no problem) to 10 (maximum probability of having a problem) is derived for each port, using the combined information of p values and trend. This process is repeated for every port of every network card each time an NHP analysis is executed (every 5 min in this case).

To recap, all the end user must do is train a ML model from a dataset and start a recurring NHP analysis for new incoming data. These operations can be performed through the application's graphical user interface (GUI). The only subject matter expertise required was to 1) ensure that the training dataset is truly representative of normal network conditions, and 2) select appropriate PMs (Table I) that are sensitive to the effects we seek to detect during the analyses. Everything else is done by the ML completely unsupervised.

We then introduce various types of network problems artificially in the lab, and observe how the ML application (NHP) reacts. These results are reported in the next two sub-sections.

C. Anticipation of Network Problems

Taking a photonic network component as an example, we progressively attenuate the light signal by up to 12 dB to mimic the effect of fiber aging in "accelerated time".

As shown in Fig. 5, the effect of fiber aging is detected on the optical line-side port OSC-1-1-1 (Layer 0), on February 16th, 2018 between 16:35 and 17:09, because of PMs span loss and optical power received (OPR). As a result, NHP reports a risk factor of 9.1 for this element.

Note that the port is not yet in a problematic state. As shown in Fig. 6, the value of span loss is not that uncommon. Yet the port is identified as being risky because it is trending towards an abnormal state. In this case, the trend is fitted using a piecewise combination of degree-1 polynomials. The trend is first measured on historical data; then it is extrapolated towards the future. The pink envelope displayed in Fig. 5 represents the 95% confidence interval of the trend.

Interestingly, the Layer 1 port OTM4-1-5-1 (100 GE) is also flagged with a risk factor of 9.3. As shown in Fig. 7, this port is identified because of a different set of PMs, namely QAVG, QSTDEV, and OPR (note that Q is a PM proportional to the logarithm of pre-FEC BER, the bit error rate prior to forward error correction). Here again, the problem is reported based on a trend, even if the NE is not yet in a problematic state.

This demonstrates that unsupervised ML can enable pre-emptive maintenance for the optical Layer 0 and Layer 1. We have obtained similar results for Layer 2 ethernet cards; the corresponding graphs are not shown here.

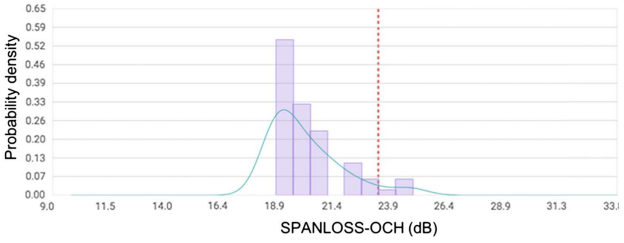

Fig. 6. Likelihood model for span loss. The purple bars correspond to the raw data used to compute the model. The solid green line corresponds to the model itself. The dashed red line corresponds to the value in Fig. 5 at the time of the last probability computation.

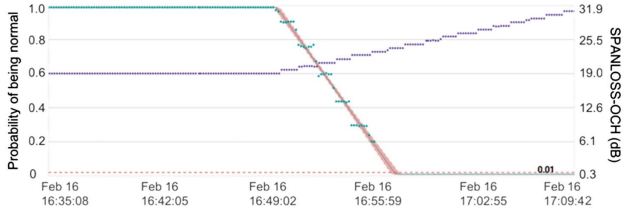

Fig. 5. Analysis details of Layer 0 port OSC-1-1-1. The x axis is UTC time. The green (pale gray) dots and left y axis show the probability of being normal versus time. The trend is fitted using a piecewise combination of degree-1 polynomials. The pink envelope displayed represents the 95% confidence interval of the trend. The purple (dark gray) dots and right y axis show the raw span loss.

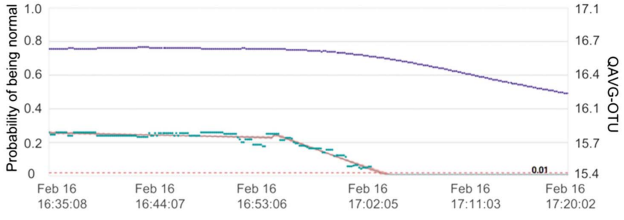

Fig. 7. Analysis details of Layer 1 port OTM4-1-5-1 (100 GE). The x axis is UTC time. The green (pale gray) dots and left y axis show the probability of being normal versus time. The trend is fitted using a piecewise combination of degree-1 polynomials. The pink envelope displayed represents the 95% confidence interval of the trend. The purple (dark gray) dots and right y axis show the average value of Q.

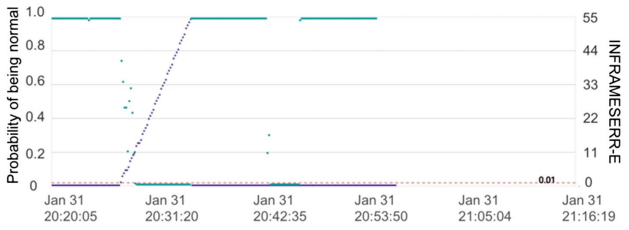

Fig. 8. Analysis details of Layer 2 port FLEX-1-4-2. The x axis is UTC time. The green (pale gray) dots and left y axis show the probability of being normal versus time. The purple (dark gray) dots and right y axis show the number of input frame errors.

D. Classification of Network Problems

Next, we consider an example using packet network components. We introduce three types of ethernet problems with the test set (see Fig. 4): frame check sequence bursts, code errors (at various rates), and laser glitches.

Figure 8 shows port FLEX-1-4-2 during two of these problems on January 31, 2018 between 20:20 and 20:55. Ethernet code errors are injected from 20:26 to 20:35. Then a laser glitch occurs from approximately 20:42 to 20:48. Both problems are flagged by a risk factor 10 and a very low probability of being normal ($<10^{-3}$). The problems are identified based on the combined information of seven PMs: OPR, ES-PCS, input frame errors, ES-E, UAS-E, UAS-PCS, and CV-PCS (see Table I).

Interestingly, different problems affect different raw PMs. As shown in Fig. 8, the ethernet code errors produced input frame errors, but the laser glitch did not. Conversely, the laser glitch produced UAS-PCS errors (not shown in the figure), but not the code errors.

In general, we observed that all problem types tested in the lab were flagged by NHP risk factors but each resulted in different raw PM patterns. These results are summarized in Table II.

This demonstrates that: 1) unsupervised ML can be used to detect a variety of network problems robustly and 2) it will be possible to use special datasets to train supervised ML to identify different root causes for problems based on different raw PM patterns.

These results were obtained with a pure Ciena 6500 network, but we believe that our application works equally well with other packet-optical devices.

E. Added Value of the Application

The NHP application enables pre-emptive maintenance by identifying risky NEs from their trends before they become problematic for network operations. This can be very valuable because network operators no longer need to react to catastrophic events but can work on risky network elements during scheduled maintenance windows.

In combination with Big Data infrastructure, the application can continuously monitor arbitrarily large and complex networks automatically. When abnormal elements are identified, the application helps operators to troubleshoot issues and identify root causes more quickly. In the near future, it is expected that the application will be able to learn to do this automatically.

The insights reported by the application are reported on a GUI. However, they can also be used to trigger follow-up actions automatically. For example, in the simplest case, this can mean opening tickets in a troubleshooting system or sending messages to on-call experts. In more advanced solutions, services can be automatically re-routed to their protection paths.

IV. Conclusions

We are in a golden age of artificial intelligence. It is now established that ML not only works technically, but is also valuable economically. In communication networks, current ML is only scratching the surface, but there is no doubt

TABLE II
Raw PM Patterns Observed for Network Problems with Different Root Causes

	Line-side light attenuation	Frame Check Sequence Bursts	Code Errors	
Raw PM	Fiber aging	Malfunctioning card	Loosely connected port, dirty fibers	Laser glitch
Span Loss	↑	–	–	–
Optical Power Received	↓	–	–	↓
Bit Error Rate–average	↑	–	–	–
Bit Error Rate–deviations	↓	–	–	–
Ethernet–Frame Errors	–	↑	↑	–
Ethernet–Errored Seconds	–	↑	↑	–
Code Violations Errors	–	–	↑	–
Physical Coding Sub-layer Errored Seconds	–	–	↑	–
Physical Coding Sub-layer Unavailable Seconds	–	–	–	↑

that this technology will be applied successfully in the future.

In this paper, we have reviewed how the main ML concepts can be applied to networking applications from a theoretical viewpoint. In addition, we have presented a concrete ML application that is being deployed in production today.

In the near future, we expect that a series of impactful ML applications will be developed in the networking space. Descriptive and predictive applications simply need "read only" access to network data. Hence, they are safe and relatively simple to deploy even at an experimental stage. Nevertheless, they have the potential to leverage arbitrarily sophisticated ML algorithms to solve many problems.

With ML, Big Data, and SDN infrastructures all coming to maturity, the telecommunications and networking industry is getting ready to exploit closed-loop systems driven by ML applications. This will pave the way for "adaptive networks" that will transform today's static network into a dynamic, programmable environment driven by analytics and intelligence.

Acknowledgment

Thanks to Sarah Burke, Merlin Davies, John Dell, Alex Dodin, Emil Janulewicz, Mina Paik, Arslan Shahid, Olivier Simard, Abel Tong, Long Tran, Thomas Triplet, Mark Weddle, and the entire Blue Planet Analytics team at Ciena. Thanks to Choudhury Al Sayeed, Tom Bonta, Dave Doucet, Jiten Goswami, Trevor Ibach, Anurag Prakash, Maurice O'Sullivan, Andrew Shiner, Craig Suitor, and Gary Swinkels for subject matter expertise on optical networks and Ciena 6500 devices. Thanks to Jay Fonte and Mayank Goyal for the demonstration in Part III. Thanks to Prof. Yan Liu from Concordia University (Canada) and Prof. Amir Farbin from the University of Texas (USA). Thanks to Sylvie Brunet for the support.

References

[1] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[3] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT, 2012.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT, 2016.

[5] G. Shilov, *Linear Algebra*, Dover Books on Mathematics Series, Dover, 1977.

[6] E. T. Jaynes, *Probability Theory: The Logic of Science*, Cambridge University, 2003.

[7] BABAR Collaboration, "Measurement of the CP-violating asymmetry amplitude sin 2β," *Phys. Rev. Lett.*, vol. 89, 201802, 2002.

[8] ATLAS Collaboration, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC," *Phys. Lett. B*, vol. 716, pp. 1–29, 2012.

[9] CMS Collaboration, "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC," *Phys. Lett. B*, vol. 716, pp. 30–61, 2012.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp 436–444, 2015.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *25th Int. Conf. on Neural Information Processing Systems*, 2012.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on ImageNet classification," arXiv:1502.01852, 2015.

[13] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, pp 354–359, 2017.

[14] "Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model," ISO/IEC standard 7498-1, 1994.

[15] D. Charlton, S. Clarke, D. Doucet, M. O'Sullivan, D. L. Peterson, D. Wilson, G. Wellbrock, and M. Bélanger, "Field measurements of SOP transients in OPGW, with time and location correlation to lightning strikes," *Opt. Express*, vol. 25, no. 9, pp. 9689–9696, May 2017.

[16] M. Baak, G. J. Besjes, D. Cote, A. Koutsman, J. Lorenz, and D. Short, "HistFitter software framework for statistical data analysis," *Eur. Phys. J.*, vol. C75, 153, 2015.

[17] https://www.scipy.org.

[18] http://scikit-learn.org/.

[19] https://keras.io.

[20] https://www.tensorflow.org.

[21] http://torch.ch.

[22] https://www.r-project.org.

[23] https://root.cern.ch.

[24] http://spark.apache.org.

[25] T. Fawcett, "An introduction to ROC analysis," *Pattern Recogn. Lett.*, vol. 27, no. 8, pp. 861–874, June 2006.

[26] http://www.blueplanet.com/resources/Blue-Planet-Network-Health-Predictor.html.

[27] T. K. Ho, "Random decision forests," in *3rd Int. Conf. on Document Analysis and Recognition (ICDAR)*, 1995.

[28] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*, MIT, 2009.

[29] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp 594–611, 2006.

[30] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, 2009.

[31] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng, "Building high-level features using large scale unsupervised learning," arXiv: 1112.6209, 2011.

[32] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," *ACM Comput. Surveys*, vol. 41, p. 15, 2009.

[33] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT, 1998.

[34] D. P. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou,

H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[36] https://sc17.supercomputing.org.

[37] http://hadoop.apache.org.

[38] https://kafka.apache.org.

[39] C. Reece, "Mobile World Congress," 2018 [Online]. Available: https://spectrum.ieee.org/tech-talk/telecom/wireless/teaching-an-artificial-intelligence-to-operate-a-telecommunications-network-isnt-possibleyet.

[40] "Five machine learning applications in telecoms," June 2017 [Online]. Available: http://sysmech.co.uk/machine-learning-applications/.

[41] K. Sennaar, "Telecom machine learning applications—Comparing AT&T, Verizon, Comcast and more," Feb. 2018 [Online]. Available: https://www.techemergence.com/telecom-machine-learning-applications/.

[42] http://www.ciena.com/insights/what-is/What-Is-the-Adaptive-Network.html.

[43] http://www.blueplanet.com/products/analytics.html.

David Côté is co-founder and lead engineer of the Blue Planet Analytics (BPA) program at Ciena. He holds many patents in the field. His group is developing a Big Data and Machine Learning platform that supports software products and professional services for Ciena's customers. He also participates in various research projects with Ciena's Photonics group and CTO office, as well as university partners. BPA was recognized by two industry innovation awards (Telecom-Asia and Metro Ethernet Forum) in 2017. He received a Ph.D. in Physics from Université de Montréal in 2007. Prior to joining Ciena, he worked as a particle physicist for 14 years at the Stanford Linear Accelerator Center (USA) and CERN (Switzerland), where he got substantial hands-on experience with Big Data engineering, data science, and world-class research.