JOURNAL OF Optical Communications and Networking

On the suitability, requisites, and challenges of machine learning [Invited]

RUI MANUEL MORAIS

Infinera, Rua da Garagem 1, 2790-078 Carnaxide, Portugal (RMorais@infinera.com)

Received 8 July 2020; revised 15 September 2020; accepted 15 September 2020; published 1 October 2020 (Doc. ID 401568)

The introduction of 5G, the increasing number of connected devices, and the exponential growth of services relying on connectivity are pressuring multilayer networks to improve their scaling, efficiency, and controlling capabilities. However, enhancing those features consistently results in a significant amount of complexity in operating the resources available across heterogeneous vendors and technology domains. Thus, multilayer networks should become more *intelligent* in order to be efficiently managed, maintained, and optimized. In this context, we are witnessing an increasing interest in the adoption of artificial intelligence (AI) and machine learning (ML) in the design and operation of multilayer optical transport networks. This paper provides a brief introduction to key concepts in AI/ML, highlighting the conditions under which the use of ML is justified, on the requisites to deploy a data-driven system, and on the challenges faced when moving toward a production environment. As far as possible, some key concepts are illustrated using two realistic use-cases applied to multilayer optical networks: cognitive service provisioning and quality of transmission estimation. © 2020 Optical Society of America

https://doi.org/10.1364/JOCN.401568

1. INTRODUCTION

It is beyond reasonable doubt that artificial intelligence (AI) and machine learning (ML) are taking the center stage in almost all industries of our modern society [1]. On a daily basis, headlines presenting new revolutionary applications based on AI and ML are being released [2]. The range of applications is so wide, from the creation of perfumes [3] up to the development of automatic weapons [4], and almost everything in between. Given the huge success of AI and ML across all those industries, it was completely natural that the telecommunications community started to investigate such techniques in order to apply them to multilayer optical network problems. Therefore, in this paper we intend to address the following question: how can AI/ML add any value to multilayer optical networks?

It is common to assume the interchangeable use of the concepts AI, ML, and deep learning (DL). However, not only do the three concepts differ, but they are subsets of each other (see Fig. 1) [5]. AI is any technique that enables machines to mimic some kind of human behavior. This set of techniques includes logic, if-then rules, planning, automation, and optimization, among many others. AI can then be divided into two categories: weak and strong [6]. Weak AI relates to systems designed to carry out just one particular task, whereas strong AI relates to systems that may simulate some kind of human intellectual activity below the level of human intelligence. A subset of AI techniques is then called ML. ML is a data-driven system relying on statistical techniques that enable machines to improve (at specific tasks) with experience, i.e., when exposed to more data [7]. It is worth noting that an AI system that is not data-driven is not using ML [8], for instance, an autonomous logic-based decision system [9]. Finally, DL is a subset of ML [10]. The main difference between ML and DL is that the latter usually refers to the use of a particular set of ML techniques called multilayer artificial neural networks (ANNs) [10].

As can be seen in Fig. 1, all those concepts are not so recent. It is controversial, but we can assume that the AI concept was introduced early, in the 1950s, when A. Turing developed theoretical computer science and the first programmable computers were deployed [12]. It is worth noting that around the same time the perceptron was also introduced by F. Rosenblatt, which served as the basis of the development of ANNs [13]. However, the computational power available at that time was not enough to democratize the usability of this new revolutionary technique. Consequently, in the 1970s the AI field experienced a period often called *"AI winter"* [14]. However, in the 1980s, the massification of personal computers, the increased computational power of such machines, and the introduction of the backpropagation algorithm [15] contributed to the second rise of AI. These developments enabled ML techniques to be used more often, even if at early stages. The trend continued to evolve, i.e., calculations per second per $1000 continued to increase, and the introduction of smartphones and social media, among other types of electronic devices, massified data generation [5,10,11]. The environment was set around the 2010s, and ANNs making use of more

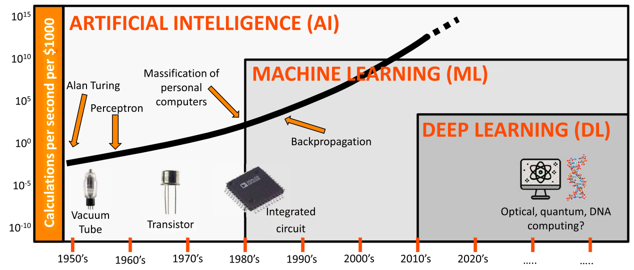


Fig. 1. Correlation between AI, ML, and DL and the exponential growth of computing over the past decades [5,11].

than one layer of neurons were successfully solved [10]. It was the birth of DL, which has been experiencing a boom since then. The expectation is that the continuous increase in computational power along with the exponential growth in data generation will pave the way toward the development of even more robust data-driven systems.

In recent years, AI/ML has been applied to communication systems and multilayer networking has been the subject of various surveys and reviews [2,8,14,16–19]. In [8] a review of the application of AI techniques for improving performance of optical communication systems and networks is presented. The range of reviewed approaches includes ML but also knowledge, reasoning, and planning methods; decision-making algorithms; game theory; or search methods and optimization. Common to [14,16–19] is the detailed presentation of the different learning types available in ML and the description of algorithms for each presented learning type. To leverage ML among the community, the surveys also present a list of use-cases where ML can have potential applicability. The use-cases presented in [14,16–18] target optical networks specifically. It is worth noting that [18] very briefly presents data sources in optical networks as well as frameworks for device monitoring and data storage. Unlike other papers that focus on the specific classes of ML algorithms, and attempt to illustrate them using use-cases applied to optical networks, this paper provides a brief introduction to key concepts and challenges on adopting ML in production environments where scale is an issue and a myriad of data sources are available. It is intended to highlight the conditions under which the use of ML is justified, the requisites to deploy a scalable and robust data-driven system, and the challenges faced when moving toward a production environment.

The rest of this paper is as follows: the type of problems that should rely on ML are presented in Section 2, as well as the data-flow infrastructure required. Section 3 highlights the major challenges faced in production environments, and Section 4 briefly presents the enablers for the deployment of AI/ML in multilayer optical networks. In Section 5 all introduced concepts are illustrated using two realistic use-cases: cognitive service provisioning and quality of transmission estimation. Finally, in Section 6 the main conclusions are drawn.

2. MACHINE LEARNING: WHEN AND HOW

The conventional engineering approach usually starts by acquiring the domain knowledge; i.e., the problem is scrutinized and a mathematical model often derived. Afterward, an algorithm is programmed in order to find solutions for the mathematical model, assuming that the model is an accurate representation of the reality [19]. In opposition, ML is the field of study that intends to give computers the ability to learn from data without being explicitly programmed [7]. A selected ML model is presented with input examples and their respective outputs. The model is then considered successfully trained when the difference between its predicted outputs and the outputs given in the training set is satisfactory. It is then expected that the trained ML model can generalize well, enabling the prediction of examples that were not explicitly used in the training phase. Thus, the ML approach replaces the phase of acquiring domain knowledge by the task of collecting a large dataset that represents the behavior of the problem, as well as replaces the phase of programming a set of rules by the task of training a preselected model.

ML can learn a function that maps inputs to outputs. Particularly, ANNs with one or more layers are proven universal function approximators; i.e., regardless of what function we are trying to learn, a large ANN will be able to approximate that function [10,20,21]. In theory, that means that ANNs are applicable to a plethora of problems. However, there is no guarantee that the training algorithm will be able to learn the target function, even if proven that is possible. Learning can fail for two reasons: (i) the optimization algorithm used for training is not able to tune the parameters that correspond to the desired function, or (ii) the training algorithm overfits the training dataset [10]. Thus, the opposite side of the coin shows the probabilistic and statistical nature of trained models. Even the best-trained ML model will hardly be 100% accurate in its predictions. Therefore, tolerance to errors is the first criterion that should hold for the adoption of ML. Moreover, although recent advances in ML systems are impressive, they are not equally suitable for all tasks. Thus, the use of a ML approach in lieu of a more conventional one should be justified based on its suitability and potential advantages [19,22].

In this context, this section intends to address the questions of when ML should replace the conventional engineering approach and how can it be done. It starts by enumerating the conditions that should hold for the suitability of ML, followed by the infrastructure requisites that should be in place prior to moving toward production environments. Moreover, the complex decisions taken by a data scientist to train a robust ML model are also briefly described.

A. Suitability and Cost-Effectiveness

As aforementioned, ML is not equally suitable for all types of tasks. Therefore, it is helpful to have some guidelines in order to distinguish between tasks that can benefit from the use of ML from tasks where ML is less likely to be successful. Based on [19,22], four major criteria were identified:

1. conventional approaches are not applicable or undesirable due to model-deficit or algorithm-deficit;

2. details of how the task is solved are not relevant, i.e., compatible with the "*black-box*" approach;
3. the phenomenon or function being learned is stationary for a sufficiently long time; and
4. a sufficiently large labeled training data set exists or can be created.

The first criterion that should be fulfilled is that the traditional approach has some deficit in terms of model or algorithm. Model-deficit is present in problems where no physics-based mathematical model exists due to insufficient domain knowledge [19]. In such cases a conventional model cannot be applied, and ML appears as the solution. In the algorithm-deficit case a mathematical model is available, but the algorithms to run it are too complex. Complexity can come in terms of lack of computational power to run the existing mathematical model or due to computational velocity, in a time-constrained environment. The second criterion that should be taken into consideration is the required level of detail in explaining the reasoning for such predictions. Most ML models are often seen as *"black-box"* approaches that do not make use of the same intermediate abstractions that humans do [22]. Therefore, ML targets problems that do not require the application of logic, common sense, or explicit reasoning based on background knowledge. The two last criteria to consider relates to data dynamicity and volume. Obviously, ML models generalize better when the distribution of future examples is similar to the distribution of training examples. If the gap between both distributions increases over time, retraining is often required. In this case the suitability of ML depends on the rate of change relative to the rate of acquisition of new data as well as retraining the algorithm. Finally, the last criterion is that huge volumes of data are available. It is expected that the more training examples are available the more accurate are the ML predictions. However, it is also important that the used data are properly labeled and balanced and represent as much as possible the environment to model. Unwanted biases present in the training data or missing irregularities in variables can easily hamper ML suitability.

Apart from the four presented criteria, there are also non-technological factors that will affect the suitability of ML. The performance of the ML model depends on the amount and quality of data used. However, it is not always economically viable to retrieve and, mostly, label the required volumes of data. The cost efficiency of ML when compared to conventional approaches is then directly related to the cost per labeled data example [2]. If a representative data set for the task can be generated with low cost, the motivation to use ML is high. For instance, social media is labeling data on a daily basis almost for free. However, if extended machinery and manpower are required to label the data, the motivation shrinks, for instance, as a result of deploying more/upgrading existing monitoring devices, creating dedicated setups to generate data, and human labeling of data, among other examples [23]. This scenario demands showing higher economic benefits from adopting ML.

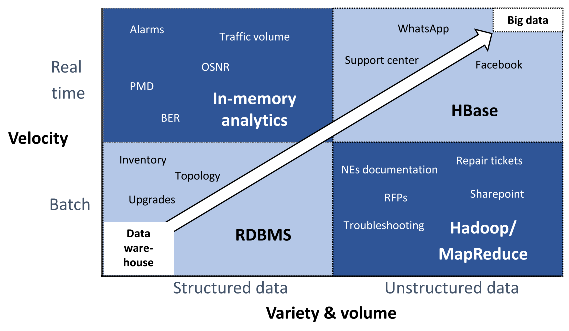


Fig. 2. Evolution of data characteristics from the traditional data warehouses toward Big Data.

B. Big Data Infrastructure

Exploring data for pattern recognition and knowledge discovery has been used by data warehouses for a long time [24]. However, some important data characteristics evolved over time giving rise to the appearance of Big Data (see Fig. 2) [25,26]. Unlike traditional data, Big Data is primarily defined by five characteristics: variety, availability, volume, velocity, and veracity [26]. Variety refers to the ability of handling not only structured data but unstructured data as well. As the name suggests, structured data adheres to a predefined data model (usually tabular format) and are therefore straightforward to analyze (e.g., Excel files or SQL databases). Because of the predefined data model, each field is discrete and can be accessed separately or jointly with data from other fields. However, most of the data generated nowadays is unstructured; i.e., the data either do not have a predefined data model or are simply not organized in a predefined manner (e.g., logs, text-heavy, audio, video, PDF, etc.). This results in irregularities and ambiguities that made it difficult to manipulate using traditional databases. Since a large part of data in organizations is unstructured, the ability to extract value from unstructured data is one of the main drivers behind the quick growth of Big Data [25]. Another three characteristics that evolved rapidly were availability, volume, and velocity. Much higher volumes of data are available nowadays, and they are generated at much higher rates. Moreover, the required response times from generating data to process it and translate the retrieved information into a decision are being dramatically shortened [27]. However, the evolution of the previously mentioned characteristics has given rise to an increase in inconsistent or erroneous data. As data are captured from several sources the veracity of the data is harder to maintain, which can obviously impose huge damage on the achieved conclusions.

The other important main driver for the explosion of Big Data was the continuous cost decrease in storing and processing hardware [11]. Figure 2 displays a schematic view of the evolution of data characteristics and the respective underlying storage infrastructure. As can be seen, the storage layer of the infrastructure depends on the data characteristics and type of data ingestion. Traditional database technologies such as relational database management systems (RDBMSs) have limited storage capacity and rigid management tools. Thus, companies could not store all their data for long periods or manage huge datasets [25]. Data were ingested, processed, and deleted in batches. However, for certain use-cases near real-time analysis is required (processing lasts from seconds up to a minute) and batch processing using data stored on physical disks is too slow. In-memory analytics technologies are used in these cases. With this technology, the data ingested in real time reside in the random access memory (RAM) of the server, resulting in data processing and querying that is more than 100 times faster [25]. However, for both RDBMS and in-memory analytics, centralized server managing and processing capabilities hampered the scalability of such infrastructures. As a result, new types of parallel computing frameworks and technologies were created to improve the storage, processing, and analytic capabilities of such data-flow systems [25]. In recent years, the Hadoop framework being selected has the standard option for Big Data infrastructure deployments [25–27]. Hadoop is an open-source distributed file system that allows the partitioning of computation processes across many host servers that are not necessarily high-performance computers [27,28]. It has two main components: the Hadoop distributed file system (HDFS) and the MapReduce programming model [25,27]. HDFS is the storage layer and provides the basis for parallel Big Data storage, whereas MapReduce is a programming model that enables the easy development of scalable parallel computation applications. HDFS mainly targeted batch data ingestion. Therefore, HBase was introduced for real-time data ingestion in parallel computing environments. HBase is an open source distributed nonrelational database that is built on top of HDFS and is designed for low-latency operations [25].

Enhanced storing capabilities is just one of the various functions that a Big Data infrastructure should efficiently perform. In addition, it should also be capable of collecting, cleaning, processing, and analyzing huge datasets coming from different heterogeneous sources, in a secure and private way [25,26]. Figure 3 displays a schematic representation of the different layers that are present in Big Data infrastructures.

The first layer, corresponding to the entering point of the infrastructure, is data ingestion [25,29]. This layer manages the collection, aggregation, and transport of large amounts of data from a number of different heterogeneous sources to a centralized data lake. It is also responsible for removing duplicated entries. For instance, sources of data in multilayer networks can be network elements (NEs), customer databases, internal documentation, troubleshooting tickets, etc. [18]. The data ingestion can be realized in batches, at specific time intervals, or in streaming, for real-time applications [25,29]. The ingested data are then saved in the storage layer. This layer is composed of a cluster of storage and processing computer resources and saves an exact copy of the sources [25,29]. The following layer in the data flow is the processing layer. This layer is responsible for ensuring the veracity and homogeneity of the data [25,29]. Other preprocessing tasks such as data enrichment or data chunk creation for parallel processing are also performed here. The last layer is then the exploitation layer. This is the layer where data mining; ML model training; and other statistical, analytical, or visualization tools can be deployed, on top of trustworthy data [25,29]. Transversal to all layers is the data governance layer. This layer manages the end-to-end process and ensures its security, privacy, and memory (metadata) [25,29]. It is also responsible for allocating system resources to the various applications running in the cluster and scheduling tasks to be executed on different cluster nodes.

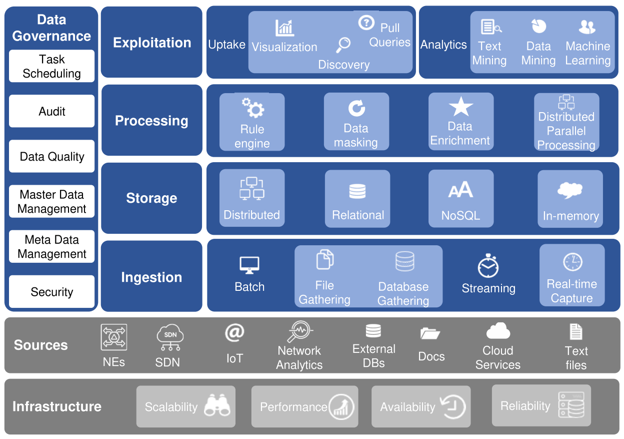


Fig. 3. Schematic representation of the different layers present in Big Data infrastructures.

One can deploy the full Big Data infrastructure in-house. However, the selection of the technological stack may be time-consuming and challenging as the landscape of possible solutions is overwhelming and ever evolving [30]. In fact, many parameters should be considered such as technological compatibility, deployment complexity, cost, efficiency, performance, reliability, support, and security risks. In order to facilitate Big Data infrastructure deployment, some integrated Hadoop distributions are available [25,29]. Most of the distributions are hardware agnostic, such as Cloudera [25,29,31]. However, public cloud service providers are making more capabilities of AI/ML (and the underlying Big Data infrastructure) available as a service, thus removing the need for having the full stack deployed from scratch. Some examples are Amazon AWS [32], Microsoft Azure [33], or Google Cloud [34]. It is worth noting that a very brief overview was given regarding Big Data technologies. A deep analysis of the specific technological stack is out of the scope of this paper; however, further details can be found in [25–29].

C. Exploiting Data

Prior to the deployment of Big Data infrastructures, data were usually siloed; i.e., the data of an organization, department, or device were isolated from and not accessible to other parts of the organization, department, or device [27]. The deployment of Big Data lakes (i.e., common data infrastructure accessible across the entire organization), as an alternative to the previous data silos, opened the door to correlate data that were not possible to correlate before. This alone made possible the extraction of important knowledge even without the need for advanced ML algorithms or DL [35]. Moreover, owing to the traditional database system limitations, typical tasks solved using data followed a top-down approach, i.e., focus on the specific problem that data might solve and only collect the data needed to solve it. With the advent of Big Data, a bottom-up approach can now be followed; i.e., an exploitation of the data currently available is done, and only then are the problems that data might help to solve determined.

Four major decisions need to be made by a data scientist during the ML training phase: type of learning, ML model, ML model parametrization, and optimizer algorithm. The first decision is the selection of the learning type that fits the task. Learning types are usually grouped based on their learning style and target objective. The most common learning types are supervised, unsupervised, reinforcement, active, and online [7,18,19]. Detailed explanation of the learning types is out of the scope of this paper; for more details see [7,10,18,19]. The decision about the learning type is often fixed by the task we intend to perform, and an experienced data scientist can quickly identify it. However, the remaining decisions require extensive comparison. There exists a plethora of different algorithms for each learning type [7,10]. For instance, logistic regression, support vector machines, k-nearest neighbor, and random forests are just some examples of supervised learning algorithms that can perform classification [7]. Moreover, even though some intuition can exist for the suitability of a specific model, it is impossible to know which one is the best *a priori*. The *"no free lunch"* theorem states that no single ML model works best for every problem [36]. Moreover, there is no universal procedure for examining a training set and choosing an algorithm that will generalize well. As a result, it is paramount to evaluate and compare different ML models to gain insight on the most promising one for the specific task. The ML model then needs to be parametrized so that its behavior can be tuned to fit the data. Most ML models rely on parameters or hyperparameters [37]. A parameter is a configuration variable that can be estimated from the given data. Thus, ML models relying on parameters are easier to tune. However, many of the most powerful ML models rely on hyperparameters; this is a configuration whose value cannot be estimated from the data and therefore needs to be found [37]. Finding the best combination of hyperparameters is itself an optimization problem. Many hyperparameter optimization/tuning methods can be found in the literature, the following being the simplest ones: grid search and random search [37]. In the grid search each hyperparameter is varied following a predetermined criterion, whereas in the random search hyperparameters are varied randomly. All those decisions target the goal of learning. Learning is the reduction of the difference between the predicted and the actual output, i.e., the minimization of a cost function. Thus, it involves mathematical optimization in highly nonlinear multivariate environments. After the cost function design, the optimizer algorithm needs to be selected. It is worth noting that each optimizer algorithm also has its own set of hyperparameters that need to be tuned. The complexity and variety of options that need to be evaluated in order to have a properly trained ML model further reveal the importance of having a reliable and high-performance data infrastructure and of carefully evaluating the suitability of ML for the task. A more detailed explanation of the complete training workflow can be found in [18]. From an implementation point of view, various open frameworks enabling the use of ML/DL models as black boxes are currently available. Some examples are scikit-learn [38], TensorFlow [39], or Caffe [40].

3. MOVING TOWARD PRODUCTION ENVIRONMENTS

ML-based applications are no longer a purely research discipline, and they currently assist important decision-making [22]. In this context, various challenges are faced when moving toward a production environment. This section enumerates and briefly describes some challenges faced by industrial ML applications, namely [2,16],

1. systems take priority over algorithms;
2. the objective function is usually multivariate and unknown by a single person;
3. model fairness and robustness;
4. data security and reliability; and
5. model interpretability and traceability.

The first barrier that a ML application will face when leaving the research world is that systems take priority over algorithms. In research ML accuracy takes full priority, most of the time at the expense of long run times and the use of complex and state-of-art ML algorithms. However, industry is a time-constrained environment where a trade-off between accuracy and processing time must be guaranteed. Faster is always better, and slower must be justified and show paramount benefits. In addition to the time constraint there exists the difficult design of the objective function. Research ML usually relies on clean single-objective functions. However, typical cost functions in industry usually have conflicting objectives; i.e., an improvement of one objective may negatively affect the other objectives. Multiobjective cost functions not only increase the complexity of the accuracy/speed trade-off optimization but also make its design difficult. The industry world is based on complex systems that are not deeply understandable by a single person. Thus, it is hard to have a clear view of the various objectives and their respective degree of priority.

After surpassing all these technical challenges comes the inherent issues of statistical and probabilistic methods. It is consensual that 99% accuracy in training alone is not enough for ML in operation. Further aspects such as fairness, robustness, explainability, and traceability must also be addressed in order to increase the trust of users/customers [16,41,42]. The main argument against, and major source of human anxiety, is the intrinsic uncertainty of the outputs. Major deviations can occur when an unknown mismatch between the training data distribution and the data distribution encountered in operation is present. As expected, this is a more worrisome problem as the environment in which the ML application is used becomes more dynamic. Thus, the deployment of monitoring tools to control biases of the data as well as changes in the behavior of the problem are mandatory. In [43], a framework to monitor ML algorithm predictions in optical networks is presented. However, robust methods to calculate confidence intervals [44] and to trigger the retraining phase are not addressed. Given the probabilistic nature of the algorithms, erroneous predictions can happen for the best-trained algorithm. Therefore, reliable criteria should be adopted for triggering the retraining phase. In that regard, [45] presents an open toolkit for algorithmic fairness. The package includes a comprehensive set of metrics and algorithms to track ML performance and mitigate possible biases. Another important aspect to verify is the robustness of the application. ML models can present unexpected vulnerabilities to adversarial examples; thus, a robustness verification to adversarial examples stating the level of vulnerability of the models is of paramount importance [46]. Owing to the numerous aspects that ML applications should follow, datasheets for datasets and declarations of conformity for ML services are being proposed [42,47]. The documents should contain purpose, performance, safety, security, and provenance information for examination by users/customers [42,47]. Although solid advances in solving the fairness and robustness issues of ML are being done, the explainability, interpretability, and traceability of such algorithms are still very much an open problem that is gaining popularity [48]. Interpretability methods for gaining a better understanding about the strategies followed by ML algorithms for problem-solving is of paramount importance as different users/customers have different needs of explanations [48]. In addition to interpretability, ML also lacks traceability, which focuses on tracing from output results back to input features [16].

4. COGNITIVE LOOP IN MULTILAYER NETWORKS

The goal of AI is to create systems capable of autonomously making decisions based on the perceived environment. Therefore, AI systems should be empowered with functions to observe the environment, decide based on what was observed, and act accordingly [18,49–52]. The combination of these three functions is usually called a cognitive loop [52]. Figure 4 displays a schematic representation of the cognitive loop deployed in multilayer optical network environments. As can be seen, ML is just one small (yet powerful) part of what is required to deploy an autonomous decision-making system for multilayer networks.

The observe function in multilayer networks should be deployed in all possible sources of data. Sources of data in networks can have their origin in three layers—resource, service, and customer—and across two vertical perspectives—infrastructure and product and operations [53]. In the resource

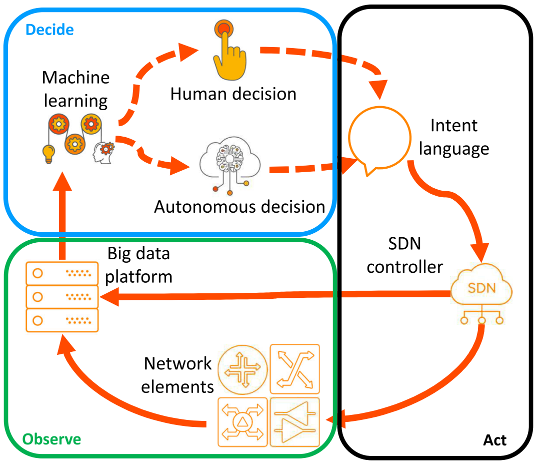


Fig. 4. Cognitive loop in multilayer optical networks.

layer, the performance of the NEs, including information related to alarms or key performance indicators (KPIs), is gathered. The service layer includes data related to the provisioning of services (voice, data, and video) such as service-level agreements (SLAs) or history logs. At the customer layer the main task is customer relationship management, which handles user inquires, orders, and trouble tickets. For some specific use-cases it would also be interesting to have external information such as weather forecasts [52]. All of this huge volume of generated data should flow toward the Big Data infrastructure (see Section 2.B). The AI system then needs to make decisions based on what was observed, i.e., based on the received data. The decide function is the heart of the cognitive loop. Here the ML algorithm, based on the current and historical data, performs its own analysis and learns from the network behavior. The insight obtained through the ML algorithm will then serve as the basis for a decision. Owing to all the uncertainties inherent in a statistical method (see Section 3), in the early stages of the system deployment one may expect that a human decision (on top of the ML insight) would be the standard process. However, as the system matures it should evolve into an autonomous decision-making system. The final function is the act function based on the decision that is made. At this point, the decision is translated into the system by means of some intent language and the status of the NEs adapted [52].

Analyzing the cognitive loop, we can clearly identify that some crucial enablers are already deployed on the ground: programmable hardware with monitoring capabilities and centralized software defined networking (SDN) control. Adaptability requires flexibility and remote control at both the electrical and optical layers. In modern multilayer optical transport networks, these flexible capabilities span from hybrid optical channel data units (ODUs)/packet switches (enabling the grooming of multiple types of client traffic over the same channel) [54,55] to colorless, directionless, contentionless (CDC) reconfigurable optical add/drop multiplexers (ROADMs) (enabling dynamic spectrum management) [56] up to highly spectrally efficient coherent wavelength division multiplexing (WDM) interfaces (which allow fine-grained trade-off between capacity and reach) [57]. This enhanced flexibility allows multilayer networks to be remotely configured and adapted to changing conditions. However, to do so in multidomain and multivendor environments, SDN controllers were introduced [58]. While traditional optical network management has been based on proprietary systems, the introduction of SDN has brought in extensive application programming interface (API) frameworks with the goal of normalizing the control and management functions across different vendor implementations [59].

Taking advantage of the aforementioned enablers, in recent years some prototypes and platform implementations of the cognitive loop in multilayer [49,51,60] or optical networks [50] have been presented. These platforms are enabling expanded automation on managing and operating the networks and will be mandatory to retrieve and label the data. In [51] a platform with monitoring/telemetry and data analytics capabilities that is suitable for disaggregated networks is presented. The ORCHESTRA project also proposes a platform to automate the cognitive loop [50]. However, the focus is more on optical networks and optical performance estimation. The Open Network Automation Platform (ONAP) [60] is another platform that provides a unified operating framework, including network control, management, orchestration, and analytics. To enhance the failure diagnosis capabilities present in ONAP, the CAT platform was proposed in [49]. These recent technological advances are decisive for the introduction of AI systems (i.e., automation) in multilayer networks and for the economic viability of retrieving and labeling the huge volumes of data that would feed autonomous data-driven systems. However, the mandatory deployment of Big Data infrastructures across all network layers and organization departments and the embracing of robust and reliable techniques for ML model training and monitoring are still an ongoing work.

5. PAVING THE WAY TO A SELF-DRIVING NETWORK

The end goal for the integration of all previous mentioned technologies is the deployment of a self-driving network [61]. A self-driving network is a network that autonomously can (re)assemble itself as requirements change, discover when something goes wrong, and fix it or explain why it cannot do so [62]. Thus, self-driving networks encompass several self-* properties, including but not limited to self-configuration, self-healing, and self-managing [2]. Although the vision to have *intelligent* networks has been around for a while, only recent advances in the required building blocks have enabled the materialization of first initiatives [43,49,51,60] and the identification of the first use-cases [16,18,43,49,51]. The end of the journey is still too far away; however, three multilayer networks use-cases where ML techniques have potential to make an impact are briefly highlighted [2]:

Failure prediction and preventive maintenance: this type of use-case detects anomalous network parameters that can potentially cause network failures, identifies the probable root cause, and prescribes preventive actions.

Cognitive service provisioning (CSP): this use-case predicts the traffic volume/growth dynamics and allocates optimized resources (spectrum, wavelengths, circuits, etc.) accordingly.

Quality of transmission (QoT) estimation: this use-case estimates end-to-end lightpath performance in optical networks by learning the characteristics from the optical devices and systems.

Failure prediction and preventive maintenance are probably the most interesting use-case, from the network operation point of view. However, this type of use-case is also the most demanding one as it relies on the correlation of data from several sources, implying the prior development of the Big Data infrastructure. Some examples of use-cases related to this field can be found in [18]. In the following, the remaining two use-cases will be presented with some more detail.

A. Cognitive Service Provisioning

The traffic management of multilayer transport networks must deal with dynamic changes under various network conditions. Nowadays, overprovisioning is the strategy of choice; i.e., the network capacity required to support the peak traffic is always active and is not adjusted dynamically following the traffic needs. Thus, wasted bandwidth is imposed and the power consumption is increased [54]. To make efficient use of the network resources, traffic management should be flexible. Flexibility in this context relates to the ability of reducing the total number of active electrical and optical resources without network outages. To do so, the traffic matrix must be quickly and accurately predicted at fixed time-steps. Historically, traditional statistical or analytical methods have been used. However, as a result of the massification of IoT devices and 5G, the known statistical properties of the traffic are no longer valid for the current traffic fluctuations [63]. The current nonlinear traffic oscillations are insufficiently modeled by the traditional statistical methods (model deficit) and advanced ML models appear as a valid alternative [63–65]. Moreover, the traffic volumes of the various layers are the type of data that operators have gathered for a long time, and the savings in power consumption obtained by such enhanced flexibility empowers the economic viability of the use-case.

Six elements and six data flows are required to implement the cognitive loop for CSP. A schematic representation of the elements and data flows is presented in Fig. 5. The monitored traffic volume is forwarded to the data lake (see green lines in Fig. 5). Subsequently, the data lake feeds the ML model with the required inputs (see black line in Fig. 5). The prediction performed by the ML model is then sent to both the ML monitoring tool as well as to the multilayer optimization model (see orange lines in Fig. 5). The ML monitoring tool will infer a confidence interval, i.e., how probable it is that the prediction is correct, based on the error of the actual observation and the previous prediction. It is worth noting that confidence intervals can be very hard to calculate for most of the advanced ML algorithms [44]. Therefore, robust mathematical frameworks should be adopted to implement this building block. In normal operation, the multilayer optimization model will compute the next optimum network state and forward it to the SDN controller (see blue line in Fig. 5). The final step is then acting; i.e., the SDN controller sends the proper commands to the NEs in order to update the network resources (see red lines in Fig. 5). To prevent traffic blocking, if some criteria defined in the ML monitoring tool fails, a ML model retraining trigger is launched. In that case, the network will start operating at full capacity upon the replacement of the old ML model by the retrained one. It is then expected that the new ML model (retrained with the expanded database) will adapt to the changed environment conditions (see dashed brown line in Fig. 5).

Regarding the ML model itself, two strategies can be employed to predict the traffic matrix based on historical data: a combination of various ML models (one per-node pair) or a single ML model to predict the full matrix [2,64]. Both strategies have their pros and cons. The per-node pair strategy relies on various low-dimensional ML models and only requires local information. Thus, the model training is fast and has low computational and memory requirements. In opposition, the full matrix ML model is a high-dimensional model requiring considerable computational power and training time. Moreover, it needs information from all NEs across the network. However,

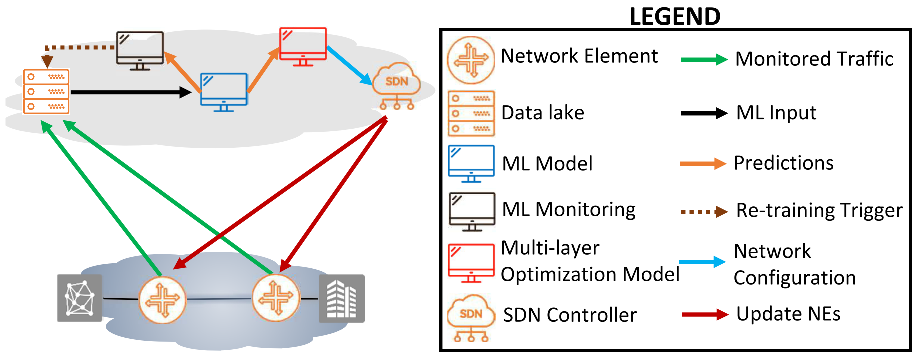


Fig. 5. Six elements and six data flows are required to implement the cognitive loop for CSP. The NEs forward monitored traffic volumes to the data lake. ML model predictions are then sent to both the ML monitoring tool and to the multilayer optimization model. The ML monitoring tool will evaluate the quality of the prediction, whereas the multilayer optimization tool will compute the next optimum network state. The SDN controller is responsible for updating the NEs.

by concentrating the knowledge in a single model, it is able to correlate traffic volumes from different nodes in order to infer the next traffic matrix; this tends to improve the estimation accuracy [2,64].

Traffic prediction falls into the time-series forecasting class of ML problems [10]. A time-series is a sequential set of data points indexed in time order typically taken at successive equally spaced points in time; i.e., it is a sequence of discrete-time data. Thus, the time-series adds an explicit chronological dependence between observations. Various ML models are being applied for time-series forecasting, for instance feedforward neural networks (FNNs) [2,64], long short-term memory (LSTM) networks [2], hybrid convolutional neural networks (CNNs)-LSTM [2], or diffusion convolutional recurrent neural networks (DCRNNs) [66,67]. The FNN is the simplest artificial neural network. This type of network does not form cycles; i.e., the data are only forwarded in one direction, from the input layer, through the hidden layer and to the output neuron(s). Owing to this simple structure, the FNN can only learn a fixed mapping from inputs to outputs. LSTM is a type of recurrent neural network and unlike the FNN has feedback connections [10]. LSTM neurons are often called cells which are composed of an input gate, an output gate, and a forget gate. This combination of gates allows LSTM to remember and to correlate observations over arbitrary time intervals which, in theory, makes them more suitable for time-series predictions. The CNN is a specialized type of neural network designed for working with 2D data, although they can also be used with 1D and 3D data [10]. Originally, they were designed to map image data to an output variable. However, generalizing, CNNs can capture spatial relationships in the data. Since in CSP it is intended to capture both spatial and temporal relationships, a hybrid model CNN-LSTM can be applied. The CNN can interpret spatial relationships and provides a time-series of those interpretations to the LSTM. The LSTM model can then discover the temporal relationships of the CNN spatial interpretation of the data. To capture both spatial and temporal dependencies in the traffic flow estimations, some more advanced algorithms such as DCRNNs are being adopted [66,67]. In simple terms, the DCRNN combines the RNN with convolutional layers in an encoder-decoder architecture.

Figure 6 presents a comparison between some of the aforementioned models and strategies when applied to a real traffic dataset. The used dataset was obtained online and comprises 43,908 measures of traffic with intervals of 5 m [68]. The original dataset corresponds to measured traffic in the Abilene network. To facilitate the analysis, the dataset was reduced to correspond to a 5-node network, thus 10 node pairs. The training set is composed of 80% of the data, whereas the testing set comprises the remaining 20%. The considered models were trained with the same data, number of epochs (100 epochs), activation function (rectified linear), and number of neurons per layer (200 neurons). The FNN has four hidden layers, the LSTM one, and the hybrid CNN-LSTM two. Three time-steps were compared: 50, 100, and 200. Figure 6(a) is related to the FNN, Fig. 6(b) to the LSTM, and Fig. 6(c) to the hybrid CNN-LSTM. Solid lines correspond to per-node pair ML models, whereas dashed lines regard the full matrix ML model. The metric used for comparison is the coefficient of determination (R^2). This is a goodness-of-fit measure for regression models that indicates the proportion of the variance in the dependent variable that is predictable from the independent variable, i.e., a statistical measure of how well the predictions approximate the real data points [7]. The R^2 ranges between 0 and 1 where an R^2 of 1 indicates the model predictions perfectly fit the data.

As can be observed in Fig. 6, there does not exist a single ML model that is best for every time-series. Depending on the specific node pair a given ML model obtains the best R^2. This highlights the need for comparing more than one ML model as well as the advantage of the per-node pair strategy of enabling the fine tuning of each model individually. However, the LSTM model is the one presenting less dependence on the strategy and number of time-steps [see Fig. 6(b)]. This is mainly explained by the LSTM's ability to interpret temporal relationships that in this case is of paramount importance. Another relevant observation is the degradation of the accuracy in the FNN and CNN-LSTM models with the increase of the number of time-steps, i.e., the number of prior observations that is given to the model in order to make a prediction (see

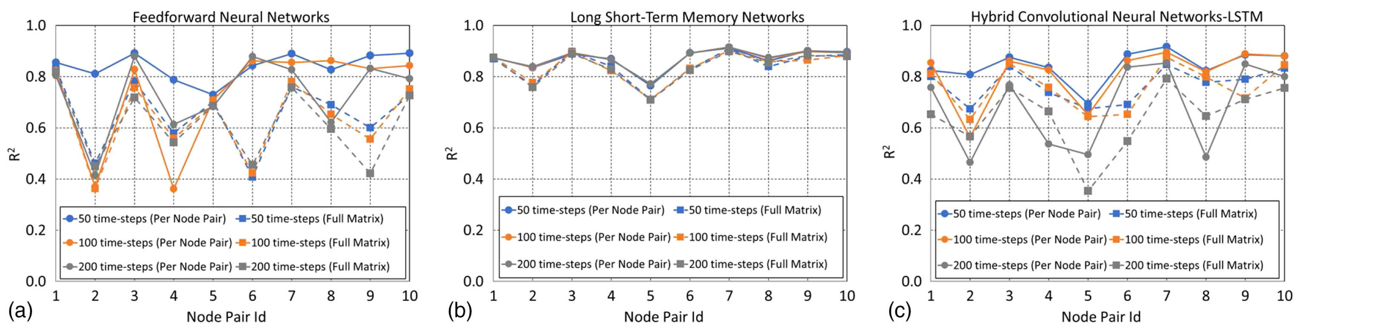


Fig. 6. R^2 for each node pair for (a) FNNs, (b) LSTM, and (c) hybrid CNN-LSTM. Solid lines correspond to the per-node pair model, whereas dashed lines correspond to the full matrix model.

gray lines in Fig. 6). This behavior reveals that these models were not able to find the correlation between very far past observations and the next one. Thus, data noise starts degrading the model's accuracy. Regarding the comparison between strategies, it can be observed that the per-node pair models tend to obtain higher R^2 than the full matrix one. Different observations are reported in [64], in which the full matrix model overperforms per-node pair ones. This highlights the strong correlation between the models and the dataset and the need for ML standards such as the ones reported in Section 3.

B. Quality of Transmission Estimation

To dynamically adjust the network resources, optical performance estimation is mandatory and a key element. However, when considering applications such as lightpath restoration both performance computation and path discovery times are very constrained [69]. Therefore, attaining both accurate and fast optical performance estimation is a key technical challenge. Currently, the optical performance models (OPMs) for online service provisioning are mostly based on simplified heuristic/analytical models, which can meet stringent computation times but at the expense of accuracy [70]. More accurate but time-consuming mathematical models are prohibitive for online applications. A recent additional constraint comes from the trend toward open and disaggregated optical networks [51,71]. OPM inputs are usually NE dependent; thus, models are commonly improved and adapted by the system vendor. In open and disaggregated environments (where NEs from multiple vendors are crossed) the proprietary models cannot be immediately applied to estimate the performance of lightpaths traversing these *"unknown"* NEs. This issue is being addressed by industry consortiums [such as the Telecom Infra Project (TIP)] promoting common models to be used in such scenarios [71]. While the OPM agreed to by the system vendor consortiums enables the applicability in open and disaggregated networks, this is achieved at the cost of extra assumptions and approximations. Therefore, attaining a method that can easily combine the estimation given by the common and eventually simpler OPM (agreed to in the scope of an industry consortium) with the estimation given by the proprietary and more detailed OPM (derived by system vendors based on a more extensive and detailed characterization) would enable the use of an improved estimation in the case of

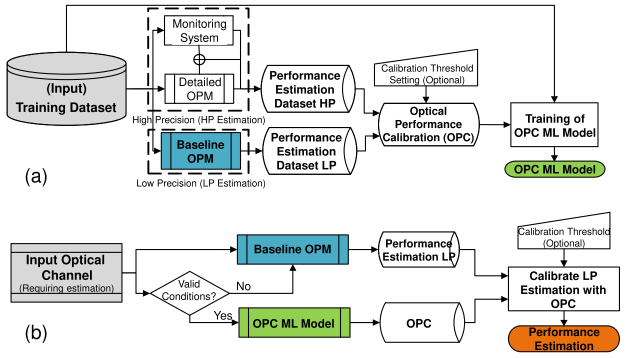
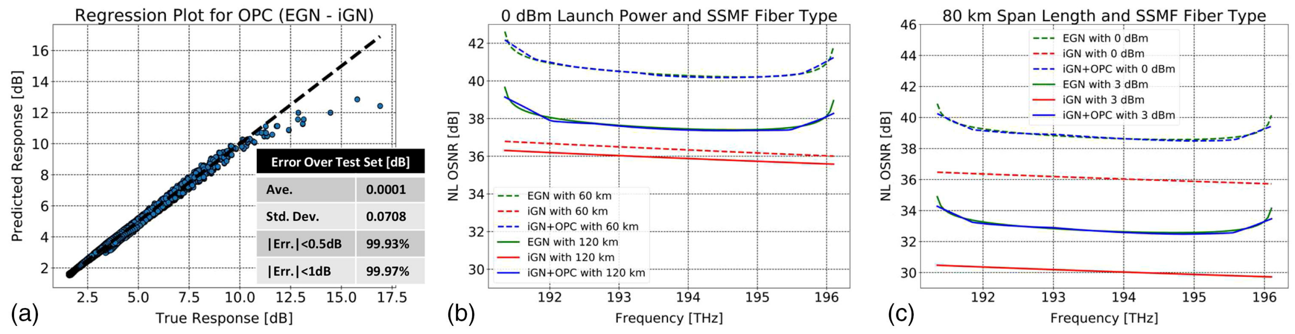

Monitoring System
(Input) Training Dataset
Detailed OPM
High Precision (HP Estimation)
Baseline OPM
Low Precision (LP Estimation)
Performance Estimation Dataset HP
Performance Estimation Dataset LP
Calibration Threshold Setting (Optional)
Optical Performance Calibration (OPC)
Training of OPC ML Model
OPC ML Model
(a)
Input Optical Channel (Requiring estimation)
Valid Conditions?
No
Yes
Baseline OPM
Performance Estimation LP
Calibration Threshold (Optional)
Calibrate LP Estimation with OPC
OPC ML Model
OPC
Performance Estimation
(b)

Fig. 7. Workflows for (a) training the OPC model by means of ML and (b) OPC model integration in the SDN routing engine. Based on two datasets with different optical performance precisions, a third OPC dataset is generated that will feed the training of the ML model. In operation, for the cases in which the conditions where OPC can be used hold, an enhanced optical performance is obtained by combining the OPM response with the ML-based calibration.

a proprietary deployment while not disabling the capabilities of operating in a fully open/disaggregated one. This is a typical algorithm-deficit challenge for which ML is suitable [2,69].

A ML-based methodology to address the aforementioned issue is proposed in [69]. The main idea is that using as a basis an OPM agreed to in the scope of an industrial consortium (and potentially less accurate) an enhanced estimation could be obtained in proprietary deployments by means of ML, without the need for replacing the OPM. A switch can be added to the routing engine that is turned on and off according to the conditions for which performance estimation is required; i.e., in the case of open and disaggregated scenarios the enhanced estimation using ML is turned off, whereas in the case of proprietary deployments the fine-tuning of the estimation is turned on. To achieve these capabilities, the methodology comprises two main stages: the training of an optical performance calibration (OPC) model by means of ML and the integration and use of the OPC ML model in a routing engine. Figures 7(a) and 7(b) display the workflow for both stages.

The first stage involves a method to generate the OPC ML model [see Fig. 7(a)]. In particular, an input training dataset is present, which can be assembled based on a collection of possible network segments or concatenations of network segments (e.g., different fiber types, different fiber lengths). This input



Error Over Test Set [dB]	
Ave.	0.0001
Std. Dev.	0.0708
|Err.|<0.5dB	99.93%
|Err.|<1dB	99.97%

Fig. 8. (a) Regression plot for optical performance calibration (EGN–iGN); (b) example for 0 dBm launch power, 60 and 120 km of span length, and SSM fiber type; (c) example for 80 km span length, 0 and 3 dBm launch power, and SSM fiber type.

training dataset can be obtained either for a specific network (in which case the resulting OPC is particularly suitable for that network) or based on a comprehensive list of possibilities that are representative of the conditions expected to be found in the generality of network deployments. The training dataset is simultaneously fed to the baseline OPM (i.e., a fast but less accurate model agreed to in the consortium), as well as to the second and more detailed OPM. After feeding both models with the input training dataset, two target datasets of performance estimations are obtained. A high-precision (HP) estimation target dataset, returned by the detailed OPM, and a low-precision (LP) estimation target dataset, computed by the baseline OPM. Both HP and LP estimations are used as input for the OPC, which will compute an outcome that is a function of both estimations, for instance the difference between HP and LP estimations. Importantly, the OPC module can also enforce additional actions, such as setting a threshold on the acceptable deviation from one of the models (e.g., by setting a maximum difference from the estimation given by the baseline OPM) to further control interpretability of the results. Both the input training dataset and the outcome of the OPC module are then used to train the ML-based model to predict the response of the OPC. The final goal is that by recombining the LP estimation with the OPC ML model estimation a HP* estimation whose accuracy is as close as possible to that of the HP estimation can be obtained.

In the second stage [see Fig. 7(b)] the SDN routing engine will perform the estimation by resorting to both the baseline OPM and the OPC ML model and computing a function of the outcome of both: e.g., adding. The SDN routing engine receives a request to estimate the performance of a given lightpath. The characteristics of the lightpath are fed in the form of features such as fiber type, fiber length, modulation format, etc. If the lightpath characteristics meet the conditions defined for the OPC ML model to be applicable they are submitted to both the baseline OPM and to the OPC ML. As a result, a LP estimation is computed as well as an OPC metric provided by the OPC ML model. Both are then combined via a suitable function, which may be set to further bound the impact of the OPC contribution according to, for example, externally defined conditions. The result of this combination is the final performance estimation of the input lightpath, which will have an accuracy close to that of the HP estimation utilized at the stage of generating the ML model for the OPC. It is worth noting that, if the lightpath characteristics do not meet the conditions defined for the OPC ML model to be applicable, optical estimation relies only on the baseline OPM. In this way it is easy to switch off the contribution of the OPC ML model in case the user wants to explicitly rely only on the baseline OPM (e.g., open or disaggregated deployment). Since running each model and then computing the result from the outcome of both models are expected to be very fast (the baseline OPM is selected to meet this expectation and the OPC ML model will be inherently fast), this routing engine can be both fast and accurate.

Figure 8 presents the results of this methodology applied to estimate the contribution to the OSNR from nonlinear (NL) impairments [69]. In particular, it considers the incoherent GN (iGN) model [70] as the baseline OPM and the enhanced GN (EGN) model [72] as the detailed OPM. The ML model used is the FNN. The goal is then to attain an accuracy comparable to that of the EGN at the expense of a computation time like that of iGN with the contribution of ML. Figure 8(a) presents the regression plot between OPC predicted and true values, whereas Figs. 8(b) and 8(c) exemplify the NL OSNR obtained by the baseline OPM (iGN), the detailed OPM (EGN), and the estimation attained with the contribution of ML (iGN + OPC) for specific configurations [2,69].

Overall, the average difference between the EGN estimation and the proposed method estimation is of 0.0001 dB with a standard deviation of 0.0708 dB. As can be observed in Fig. 8(a) almost all predictions are very close to the true response. Observing the red lines in Figs. 8(b) and 8(c), iGN tends to underestimate the NL OSNR and cannot fully capture the dependencies related to frequency and neighbor channels (i.e., see curved shaped of EGN curves versus straight lines of iGN). However, the combination of iGN with OPC can almost perfectly match EGN, showing high accuracy and the ability to capture the stated dependencies [see green and blue lines in Figs. 8(b) and 8(c)]. Importantly, this is attained at the expense of a much faster routing engine. The EGN required 330 ms to perform the required calculations, whereas the proposed method took only 8.3 ms. In this case ML enabled a 97.3% reduction in the computation time with minor impacts on accuracy [69].

6. CONCLUSIONS

Over the past decades we have been witnessing the constant development of technologies promising the appearance of an *intelligent* machine. Communication systems and multilayer networking are no exception, and the recent introduction of coherent transmission, flexible grid, and SDN, among other technologies, enhanced the network ability to efficiently adapt to changing conditions. Moreover, promising platforms for end-to-end network automation with advanced monitoring and telemetry capabilities have been recently proposed. To leverage these capabilities, much research is being done on the identification of use-cases for which ML can be applied. This paper discussed some of the challenges of ML applications when leaving research and moving toward production environments as well as the impact that this can have on the suitability of ML. It starts by enumerating four points that should hold for the suitability of ML for the tasks at hand, the highlights of which are that (i) the conventional engineering approach fails, and (ii) a large pool of data can be cost-effectively generated and manipulated. In this last regard, a detailed description of the several layers present in a robust and reliable Big Data infrastructure is given. It is worth noting that, only the deployment of a robust data lake allowing the ingestion, storing, processing, and exploitation of huge volumes of data in a private, secure, and efficient way would enable the adoption of bottom-up approaches in the use-case selection, as opposed to the actual top-down use-cases often present in the current literature. Moreover, despite proven universal function approximators, ML outputs will hardly be 100% accurate. Thus, the tolerance to errors enforced by ML demands that when moving to production environments monitoring tools should be deployed to meet fairness, robustness, and assurance. Recently, various research has been done showing the outstanding approximation capabilities of ML algorithms. However, the adoption of robust methods to calculate confidence intervals for the predictions as well as to reliably trigger the retraining phase still lay ahead. As an interdisciplinary subject, applying ML in multilayer networks encompasses numerous challenges, and the heterogeneity of skills and stacks required is one of them. Some of the challenges faced by ML applications in production environments are technology related and therefore industry independent. Thus, an effort to embrace available frameworks for ML application monitoring and algorithmic bias control should be done. In summary, while the increased operational complexity of multilayer optical transport networks is demanding the deployment of a more *intelligent* network, and ML compliant platforms are available, some challenges remain unaddressed when moving toward production environments.

Funding. Fundação para a Ciência e a Tecnologia (UIDB/50008/2020); European Regional Development Fund (LISBOA-01-0247-FEDER-045354).

Acknowledgment. The author would like to thank João Pedro, Bruno Pereira, Bernd Sommerkorn-Krombholz, and Harald Bock for valuable discussions and inputs.

REFERENCES

1. G. Gambhire, T. Gujar, and S. Pathak, "Business potential and impact of industry 4.0 in manufacturing organizations," in *International Conference on Computing Communication Control and Automation* (2018).
2. R. M. Morais, "Machine learning in multi-layer optical networks: why and how," in *Optical Fiber Communications Conference and Exhibition (OFC)* (2020), paper M1B.1.
3. R. Goodwin, J. Maria, P. Das, R. Horesh, R. Segal, J. Fu, and C. Harris, "AI for fragrance design," in *NIPS Workshop on Machine Learning for Creativity and Design* (2017).
4. P. Feldman, A. Dant, and A. Massey, "Integrating artificial intelligence into weapon systems," arXiv:1905.03899 (2019).
5. R. Ceron, "AI, machine learning and deep learning: what's the difference?" https://www.ibm.com/blogs/systems/ai-machine-learning-and-deep-learning-whats-the-difference/.
6. I. Sitnicki, "Why AI shall emerge in the one of possible worlds?" AI Soc. **34**, 365–371 (2019).
7. S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms* (Cambridge University, 2014).
8. 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. **28**, 43–57 (2018).
9. G. Phillips-Wren, "AI tools in decision making support systems: a review," Int. J. Artif. Intell. Tools **21**, 1240005 (2012).
10. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT, 2016).
11. M. Roser and H. Ritchie, "Technological progress," 2013, https://ourworldindata.org/technological-progress.
12. A. Turing, "Computing machinery and intelligence," Mind **59**, 433–460 (1950).
13. F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," Psychol. Rev. **65**, 386–408 (1958).
14. F. N. Khan, Q. Fan, J. Lu, G. Zhou, C. Lu, and P. T. Lau, "Applications of machine learning in optical communications and networks," in *Optical Fiber Communications Conference and Exhibition (OFC)* (2020), paper M1G.5.
15. D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," Nature **323**, 533–536 (1986).
16. R. Gu, Z. Yang, and Y. Ji, "Machine learning for intelligent optical networks: a comprehensive survey," J. Netw. Comput. Appl. **157**, 102576 (2020).
17. 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 **21**, 1383–1408 (2019).
18. D. Rafique and L. Velasco, "Machine learning for network automation: overview, architecture, and applications [Invited Tutorial]," J. Opt. Commun. Netw. **10**, D126–D143 (2018).
19. O. Simeone, "A very brief introduction to machine learning with applications to communication systems," IEEE Trans Cognit. Commun. Netw. **4**, 648–664 (2018).
20. K. Hornik, "Approximation capabilities of multilayer feedforward networks," Neural Networks **4**, 251–257 (1991).
21. G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems* (2014), pp. 2924–2932.
22. E. Brynjolfsson and T. Mitchell, "What can machine learning do? Workforce implications," Science **358**, 1530–1534 (2017).
23. Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: a big data–AI integration perspective," IEEE Trans. Knowl. Data Eng. (to be published).
24. W. H. Inmon, *Building the Data Warehouse* (Wiley, 1992).
25. A. Oussous, F. Benjelloun, A. Lahcen, and S. Belfkih, "Big data technologies: a survey," J. King Saud Univ. **30**, 431–448 (2018).
26. Y. Riahi and S. Riahi, "Big data and big data analytics: concepts, types and technologies," Int. J. Res. Eng. **5**, 524–528 (2018).
27. S. Sakr, "Big data processing stacks," IT Prof. **19**, 34–41 (2017).
28. T. White, *Hadoop: The Definitive Guide* (O'Reilly Media, Inc, 2012).

29. T. Coelho da Silva, R. Magalhães, I. Brilhante, J. Macêdo, D. Araújo, P. Rego, and A. Neto, "Big data analytics technologies and platforms: a brief review," in *LADaS–Latin America Data Science Workshop* (2018), p. 25–32.
30. M. Turk, "Great power, great responsibility: the 2018 big data & AI landscape," 2018, https://mattturck.com/bigdata2018/.
31. Cloudera, https://www.cloudera.com/.
32. AWS, https://aws.amazon.com/.
33. Microsoft Azure, https://azure.microsoft.com/en-us/.
34. Google Cloud, https://cloud.google.com/products/ai.
35. J. Li, Z. Wang, C. Wang, Q. Chen, P. Wang, R. Lu, S. Fu, and C. Xie, "Data analytics practice for reliability management of optical transceivers in hyperscale data centers," in *Optical Fiber Communications Conference and Exhibition (OFC)* (2020), paper T3K.6.
36. D. Wolpert, "The lack of a priori distinctions between learning algorithms," Neural Comput. **8**, 1341–1390 (1996).
37. M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning: Methods, Systems, Challenges* (Springer, 2019), pp. 3–33.
38. scikit-learn, https://scikit-learn.org/stable/.
39. TensorFlow, https://www.tensorflow.org/.
40. Caffe, https://caffe.berkeleyvision.org/.
41. P. Andras, L. Esterle, M. Guckert, T. Han, P. Lewis, K. Milanovic, T. Payne, C. Perret, J. Pitt, S. Powers, N. Urquhart, and S. Wells, "Trusting intelligent machines: deepening trust within socio-technical systems," IEEE Technol. Soc. Mag. **37**(4), 76–83 (2018).
42. M. Arnold, R. Bellamy, M. Hind, S. Houde, S. Mehta, A. Mojsilović, R. Nair, K. Ramamurthy, A. Olteanu, D. Piorkowski, D. Reimer, J. Richards, J. Tsay, and K. Varshney, "FactSheets: increasing trust in AI services through supplier's declarations of conformity," IBM J. Res. Dev. **63**, 6:1–6:13 (2019).
43. L. Velasco, B. Shariati, F. Boitier, P. Layec, and M. Ruiz, "Learning life cycle to speed up autonomic optical transmission and networking adoption," J. Opt. Commun. Netw. **11**, 226–237 (2019).
44. G. Shafer and V. Vovk, "A tutorial on conformal prediction," J. Mach. Learn. Res. **9**, 371–421 (2008).
45. R. Bellamy, K. Dey, M. Hind, S. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, S. Nagar, K. Ramamurthy, J. Richards, D. Saha, P. Sattigeri, M. Singh, K. Varshney, and Y. Zhang, "AI Fairness 360: an extensible toolkit for detecting and mitigating algorithmic bias," IBM J. Res. Dev. **63**, 4:1–4:15 (2019).
46. J. Mohapatra, T. Weng, P. Chen, S. Liu, and L. Daniel, "Towards verifying robustness of neural networks against a family of semantic perturbations," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR2020)* (2020), pp. 244–252.
47. T. Gebru, J. Morgenstern, B. Vecchione, J. Vaughan, H. Wallach, H. Daumé, III, and K. Crawford, "Datasheets for datasets," arXiv: 1803.09010 (2018).
48. W. Samek, G. Montavon, S. Lapuschkin, C. Anders, and K. Muller, "Toward interpretable machine learning: transparent deep neural networks and beyond," arXiv:2003.07631 (2020).
49. T. Tanaka, A. Hirano, S. Kobayashi, T. Oda, S. Kuwabara, A. Lord, P. Gunning, O. Gonzalez de Dios, V. Lopez, A. M. Lopez de Lerma, and A. Manzalini, "Autonomous network diagnosis from the carrier perspective [Invited]," J. Opt. Commun. Netw. **12**, A9–A17 (2020).
50. K. Christodoulopoulos, C. Delezoide, N. Sambo, A. Kretsis, I. Sartzetakis, A. Sgambelluri, N. Argyris, G. Kanakis, P. Giardina, G. Bernini, D. Roccato, A. Percelsi, R. Morro, H. Avramopoulos, P. Castoldi, P. Layec, and S. Bigo, "Toward efficient, reliable, and autonomous optical networks: the ORCHESTRA solution [Invited]," J. Opt. Commun. Netw. **11**, C10–C24 (2019).
51. L. Gifre, J. Izquierdo-Zaragoza, M. Ruiz, and L. Velasco, "Autonomic disaggregated multilayer networking," J. Opt. Commun. Netw. **10**, 482–492 (2018).
52. A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, G. Estrada, K. Máruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. Evans, F. Maino, J. Walrand, and A. Cabellos, "Knowledge-defined networking," ACM SIGCOMM Comput. Commun. Rev. **47**, 2–10 (2017).
53. C. Chen, "Use cases and challenges in telecom big data analytics," APSIPA Trans. Signal Inf. Process. **5**, e19 (2016).
54. R. M. Morais, J. Pedro, P. Monteiro, and A. N. Pinto, "Benefits of node architecture flexibility and hitless re-grooming in transport networks," J. Lightwave Technol. **33**, 4424–4436 (2015).
55. R. M. Morais, J. Pedro, and T. Rarick, "Evaluating modular node architectures with limited shelf interconnection [Invited]," J. Opt. Commun. Netw. **9**, A176–A188 (2017).
56. S. Gringeri, B. Basch, V. Shukla, R. Egorov, and T. J. Xia, "Flexible architectures for optical transport nodes and networks," IEEE Commun. Mag. **48**(7), 40–50 (2010).
57. H. Sun, M. Torbatian, M. Karimi, R. Maher, S. Thomson, M. Tehrani, Y. Gao, A. Kumpera, G. Soliman, A. Kakkar, and M. Osman, "800G DSP ASIC design using probabilistic shaping and digital sub-carrier multiplexing," J. Lightwave Technol. **38**, 4744–4756 (2020).
58. R. Alvizu, G. Maier, N. Kukreja, A. Pattavina, R. Morro, A. Capello, and C. Cavazzoni, "Comprehensive survey on T-SDN: software-defined networking for transport networks," IEEE Commun. Surv. Tutorials **19**, 2232–2283 (2017).
59. H. Bock, R. M. Morais, J. Pedro, B. S. Krombholz, A. Sadasivarao, S. Syed, L. Paraschis, and P. Kandappan, "Coming of age of AI-assisted network management & control," in *OSA Advanced Photonics Congress* (2020).
60. The Linux Foundation, "Open Network Automation Platform (ONAP)," https://www.onap.org/.
61. K. S. Atwal and M. Bassiouni, "DeepSDN: connecting the dots towards self-driving networks," in *IEEE 37th International Performance Computing and Communications Conference (IPCCC)* (2018).
62. D. Clark, C. Partridge, C. Ramming, and J. Wroclawski, "A knowledge plane for the Internet," in *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (2003), pp. 3–10.
63. L. Nie, D. Jiang, L. Guo, S. Yu, and H. Song, "Traffic matrix prediction and estimation based on deep learning for data center networks," in *IEEE Globecom Workshops* (2016).
64. J. Guo and Z. Zhu, "When deep learning meets inter-datacenter optical network management: advantages and vulnerabilities," J. Lightwave Technol. **36**, 4761–4773 (2018).
65. G. Liu, K. Zhang, X. Chen, H. Lu, J. Guo, J. Yin, R. Proietti, Z. Zhu, and S. J. B. Yoo, "Hierarchical learning for cognitive end-to-end service provisioning in multi-domain autonomous optical networks," J. Lightwave Technol. **37**, 218–225 (2019).
66. Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: data-driven traffic forecasting," in *International Conference on Learning Representations (ICLR)* (2018).
67. D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," in *IEEE Conference on Computer Communications Workshops (INFOCOM)*, Paris, France (2019), pp. 246–251.
68. M. Roughan, "Internet traffic matrices," http://www.maths.adelaide.edu.au/matthew.roughan/project/.
69. R. M. Morais, B. Pereira, and J. Pedro, "Fast and high-precision optical performance evaluation for cognitive optical networks," in *Optical Fiber Communications Conference and Exhibition (OFC2020)* (2020), paper Th3D.3.
70. P. Poggiolini, "The GN model of non-linear propagation in uncompensated coherent optical systems," J. Lightwave Technol. **30**, 3857–3879 (2012).
71. G. Grammel, V. Curri, and J. Auge, "Physical simulation environment of the telecommunications infrastructure project (TIP)," in *Optical Fiber Communications Conference and Exhibition (OFC)* (2018), paper M1D.3.
72. A. Carena, G. Bosco, V. Curri, Y. Jiang, P. Poggiolini, and F. Forghieri, "EGN model of non-linear fiber propagation," Opt. Express **22**, 16335–16362 (2014).