AI/ML-as-a-Service for optical network automation: use cases and challenges [Invited]

Citation for the original published paper (version of record):

Natalino Da Silva, C., Panahi, A., Mohammadiha, N. et al (2024). AI/ML-as-a-Service for optical network automation: use cases and challenges [Invited]. Journal of Optical Communications and Networking, 16(2): A169-A179. http://dx.doi.org/10.1364/JOCN.500706

AI/ML-as-a-Service for Optical Network Automation: Use Cases and Challenges [Invited]

CARLOS NATALINO[1,*], ASHKAN PANAHI[2], NASSER MOHAMMADIHA[2,3], AND PAOLO MONTI[1]

[1] Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden
[2] Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden
[3] Ericsson AB, Gothenburg, Sweden
[*] carlos.natalino@chalmers.se

Authors' version. Compiled December 19, 2023.

In recent years, artificial intelligence/machine learning (AI/ML) has played a significant role in automating optical networks. Despite this, the methods for creating, deploying, and monitoring AI/ML models still rely heavily on human intervention and trial-and-error. AI/ML-as-a-Service aims at automating the processes associated with AI/ML models, reducing the need for human intervention and thus facilitating the widespread adoption of AI/ML models. In this paper, we introduce the concept of AI/ML-as-a-Service in the context of optical network automation and propose an architecture for realizing this concept. We provide details of a reference implementation that focuses on the model creation stage. The reference implementation is tested using two use cases related to the quality-of-transmission (QoT) estimation of optical channels. We demonstrate that models created through AI/ML-as-a-Service are able to achieve similar performance as manually-tuned models while drastically reducing the need for human involvement. Finally, we discuss future challenges and opportunities for applying AI/ML-as-a-Service in optical network automation.

https://doi.org/10.1364/JOCN.500706

1. INTRODUCTION

Traditionally, optical networks have been built using single-vendor solutions, i.e., open line systems (OLSs) including devices and controlling software. These networks operate with a mostly-static traffic matrix that provides transport for aggregated traffic of digital services. With the introduction of 5G and the expected requirements for 6G, network operators have been moving away from this traditional approach. Modern optical networks are anticipated to handle per-service requirements, and disaggregated solutions. More importantly, optical networks are expected to provide individualized/customized transport for service-specific demands in a dynamic and agile fashion [1]. Therefore, conventional methods of controlling and managing optical networks must be reconsidered. This involves a more detailed and frequent monitoring system [2], as well as autonomous operations. Artificial intelligence/machine learning (AI/ML) is one of the key methods to realize autonomous operations of optical networks [3].

Automating the operation of optical networks can be achieved through a wide variety of use cases, usually taking advantage of AI/ML models. When provisioning an optical channel (OCh), two specific use cases come to light: quality-of-transmission (QoT) regression and QoT classification [4, 5]. In QoT regression, the optical software-defined networking (SDN) controller (O-SDNc) uses AI/ML to estimate, from the network and OCh configuration, a numerical representation of the signal quality, e.g., signal-to-noise ratio (SNR) and/or its variations [6, 7], error vector magnitude (EVM) [8]. In QoT classification, the O-SDNc uses AI/ML to determine whether or not a given OCh configuration would work given the current network state [9, 10]. AI/ML models have also been studied for a broad range of use cases for the scenarios where an OCh is already deployed. For instance, soft-failure detection, identification [11], and localization [12] methods can be used to prevent the interruption of optical services by an AI/ML-assisted analysis of optical performance monitoring (OPM) data. Visual representations of the OCh status, such as constellation diagrams, can be used to identify modulation format [13] and anomalies [14]. AI/ML can also be used for detecting and identifying physical layer breaches [15].

All the use cases mentioned above share the common requirement of using contextualized data, such as topology- or OCh-specific data, for both training and testing purposes. Operating a medium to large-scale optical network may require hundreds or even thousands of AI/ML models to achieve complete autonomous network operation. The use of contextualized models presents a challenge to the widespread integration of AI/ML models for automating optical networks. This

is because the creation and maintenance of such models currently rely on human involvement and empirical processes. For example, when developing models, it is typical to make empirical decisions on which model to utilize, such as artificial neural networks (ANNs) or support vector machines (SVMs). This is made evident by the performance comparison among AI/ML models usually adopted in the literature, which assesses several models to find the most suitable one [4, 9, 15]. Moreover, hyperparameters play a crucial role in the performance of the models, and their tuning is done manually and based on empirical knowledge. As a consequence, building AI/ML models becomes time-consuming, onerous, and dependent on empirical knowledge and trial-and-error. The tasks of training complex models, selecting the best model and hyperparameters tuning, require extensive computational resources. The need for specialized practitioners and resources for AI/ML models impede their widespread adoption, thereby hindering the full potential of autonomous optical network operation. A solution to this challenge is to automate the AI/ML lifecycle itself.

AI/ML-as-a-Service is a way to outsource the tasks related to building and maintaining AI/ML models [16]. AI/ML-as-a-Service is based on the cloud computing service model known as "*Something*-as-a-Service", where resources are provided in a timely and scalable manner over the Internet [17]. AI/ML-as-a-Service focuses on automating the stages of creating, adapting, and operating AI/ML models by utilizing a set of standard interfaces that allow communication between a client system and the AI/ML-as-a-Service framework. Internally, the AI/ML-as-a-Service framework includes a series of automated procedures that enable it to construct AI/ML models based on the specifications provided by the client system. AI/ML-as-a-Service has already been utilized in various fields such as electricity demand prediction, health, and software engineering. A subset of AI/ML-as-a-Service functionalities (mainly encompassing AI/ML model operations) has also been applied to wireless networks [18, 19]. It has shown encouraging outcomes in terms of reduction in human involvement, enhanced reproducibility, and faster model creation [16].

This paper examines the concepts and tasks related to applying the AI/ML-as-a-Service concept in automating optical network operations. More specifically, it focuses on how to incorporate the model creation and deployment capabilities of AI/ML-as-a-Service into a traditional O-SDNc and evaluates the advantages of AI/ML-as-a-Service in two typical use cases, i.e., QoT regression and classification. We are not aware of any research on the use of AI/ML-as-a-Service for automating optical networks, apart from the initial investigation mentioned in [20]. In particular, we extend the work in [20] by: *(i)* formally introducing the concept of AI/ML-as-a-Service and its associated requirements, specifying the stages that compose the AI/ML model lifecycle; *(ii)* describing the main interfaces of the AI/ML-as-a-Service framework and and their interaction with the O-SDNc; and *(iii)* demonstrating the suitability of the model creation stage and assessing the performance of the proposed implementation versus traditional manually-tuned ANNs. The performance analysis of the model operation and adaptation phases are not in the scope of this paper.

The performance assessment shows that the models generated by a AI/ML-as-a-Service pipeline achieve results comparable with the ones achieved by manually-tuned ANNs. More importantly, the process of creating AI/ML models is streamlined by reducing the need for trial-and-error experiments. Upon full integration with an O-SDNc, AI/ML models can be generated with minimal human involvement. Nonetheless, it is important to highlight that this work does not advocate to completely remove human involvement from the process of creating AI/ML models, but rather to reduce the tedious, repetitive, time-consuming, and onerous part.

2. "*-AS-A-SERVICE*" SERVICE MODEL

Before we formally introduce the proposed AI/ML-as-a-Service framework, it is important to understand what the service model (i.e., *Something*-as-a-Service) entails. The suffix "*-as-a-Service*" was defined in the late 2,000s in the context of cloud computing service offerings [21]. In this service model, providers offer a product *-as-a-Service* to their customers. In cloud computing, the three traditional products are Infrastructure-as-a-Service, Platform-as-a-Service, and Software-as-a-Service.

There is no consensus as to which is the definitive list of requirements for a product to be considered *-as-a-Service*, but five requirements are always present:

- **Controllable through the Web with an application programming interface (API)**: customers can request/adjust/release the service through an API. This enables the automated, programmatic, and unmanned use of the service. For instance, a customer can request a new virtual machine (VM) by simply sending a request to a Web endpoint.
- **Available through the Web**: the services offered *-as-a-Service* are delivered through the Web. Upon request, the customer receives an address/endpoint through which the requested service can be accessed. For instance, if you request a VM, it will be accessible through a public Internet protocol (IP) address.
- **On-demand**: the service will be available at any time shortly after the request is received. With the short time between the request and the service being available, customers can now only order when (or shortly before) the need arrives. For instance, a VM only takes the boot time (i.e., a few seconds or minutes) from the request to the time it is ready.
- **Pay-as-you-go**: the customer only pays for the service during the time in which the service is accessible. There are zero costs associated when the customer has not requested any service. For instance, a VM only generates monetary charges while it is being executed and there are processing resources associated with it.
- **No-Code**: the customer is not required to write any code (i.e., in a programming language) to have access or manipulate the service. For instance, a VM can be requested through a Web user interface or by sending a request to an endpoint.

Over the years, several other products have been proposed by academia and industry, but the five aforementioned characteristics have remained fundamental for the products to be considered *-as-a-Service*.

3. AI/ML-AS-A-SERVICE PIPELINE

Nowadays, the design and development of AI/ML models remain largely dependent on empirical knowledge and trial-and-error experiments performed by AI/ML experts. The process of

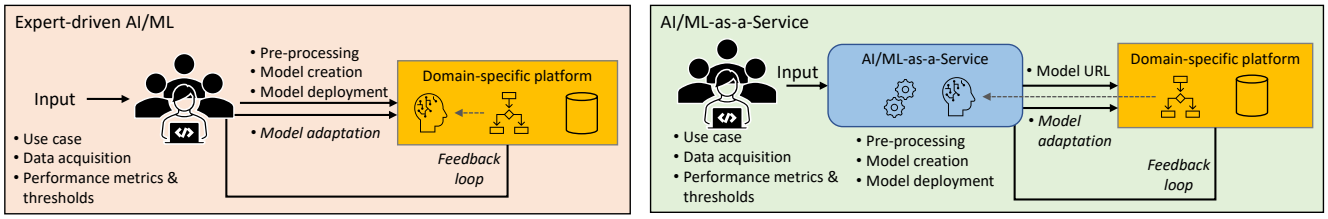


Fig. 1. AI/ML pipeline built by experts (left) and built using AI/ML-as-a-Service (right).

developing AI/ML models can be expensive due to the time and computational resources required for each trial-and-error round carried out by experts in the field. As a result, many companies face challenges in building their own AI/ML models, either due to a lack of expertise or resources [22]. Given the increasing success of AI/ML in several areas, automating this lifecycle is a necessary step toward enabling the widespread use of AI/ML.

AI/ML-as-a-Service is a relatively new area that focuses on streamlining the entire lifecycle of AI/MLmodels,and has recently gained increased interest [16]. The process of automating the AI/ML lifecycle entails developing a series of algorithms that can replicate the steps involved in creating, adapting, and operating AI/ML models. However, achieving full realization of AI/ML-as-a-Service involves two challenging tasks: *(i)* the development of a large and complex system, and *(ii)* modifying the platforms that use AI/ML models to take advantage of the AI/ML-as-a-Service. In the following, we first present a high-level overview of how AI/ML-as-a-Service reduces the complexity of the required human interaction, followed by a description of related concepts and previous works.

Fig. 1 illustrates the differences between using a traditional, expert-driven, AI/ML pipeline and a pipeline leveraging the AI/ML-as-a-Serviceconcept. Generally, an AI/ML pipeline follows three main stages. To begin, the process requires an input describing the use case. This description requires the acquisition of the data that will be utilized to train the AI/MLmodel, the applicable performance metrics, and the expected performance thresholds (e.g., minimum accuracy in the case of a classifier). Second, the model creation stage generates a model that is deployed. Finally, the deployed model should be monitored for performance degradation (based on the defined performance thresholds) and model adaptation.

Three differences can be observed between the two cases illustrated in Fig. 1. The first one is observed when it comes to processing the input. The way information is presented in an expert-driven pipeline can vary, but for AI/ML-as-a-Service, the input needs to follow a specific structure to allow for automatic processing. Sec. 5 will present an example of a formal input.

The second difference lies in the responsibilities of the experts involved. In the traditional approach, experts are responsible for the complete pipeline, which includes data acquisition, preprocessing, model creation, model deployment, and eventually model adaptation. However, in the AI/ML-as-a-Service approach, human involvement is mainly required for preparing the formal description of the pipeline and making sure that needed data is available (i.e., data acquisition).

The third difference is related to where the model is deployed. In the traditional approach, the AI/ML model is deployed within the domain-specific platform that uses it. Inference can be performed by invoking the model locally. On the other hand, with a AI/ML-as-a-Service framework, the model is deployed within the AI/ML-as-a-Service platform. The AI/ML-as-a-Service provides an uniform resource locator (URL) that can be used by the domain-specific platform to invoke model inference through the network. By generating the model independent of the domain-specific platform, the architecture becomes more flexible.

There are various techniques for the automation of the model creation and adaptation processes. For example, there are recent attempts to combine different steps into a unified pipeline, leading to the emergence of the automated ML (AutoML) research field [23]. AutoML focuses on automating the AI/ML model creation process by integrating many of the steps shown in the upper part of Fig. 2 and described (except for domain adaptation) in Sec. 4.A. Following our definition of AI/ML-as-a-Service, AutoML is an integrating part of AI/ML-as-a-Service. AutoML does not however cover the critical aspects such as the definition of APIs, *no-code* model creation, operational aspects of running ML models, etc.

Another important discipline related to AI/ML-as-a-Service is machine learning operations (MLOps), also referred to as ML DevOps (ML Development and Operations). MLOps primarily concentrates on the comprehensive lifecycle management of AI/ML models, encompassing their development, deployment, serving, monitoring, and maintenance [24]. This discipline emphasizes the automation of essential processes involved in building and operating AI/ML models. MLOps tools and practices are integrated as part of AI/ML-as-a-Service to deliver the required level of automation. As the natural deployment of AI/ML-as-a-Service is on cloud infrastructure, AI/ML-as-a-Service provides functionalities beyond the ones offered by MLOps by providing accessible and scalable cloud-based platforms that simplify the integration and deployment of AI/ML models. MLOps services offer pre-built AI/ML models, API, and a managed infrastructure, reducing the complexity for users who may not possess extensive AI/ML expertise.

There are a few works in the literature targeting a subset of AI/ML-as-a-Service functionalities. In [18], the authors consider a pre-trained model, and analyze the challenges in on-boarding, deploying, and invoking AI/ML models. In [19], the authors consider a workflow where the model is already defined (i.e., decision tree), and trained based on contextual data. However, a full realization of AI/ML-as-a-Service requires additional stages and tasks.

4. AI/ML-AS-A-SERVICE: STAGES AND TASKS

Fig. 2 presents a breakdown of the stages and steps involved in the proposed AI/ML-as-a-Service pipeline. The pipeline is divided into two main stages: *model creation/adaptation* and *model operation*. During the model creation/adaptation stage, a suitable AI/ML model is generated based on the provided dataset and a formal specification of the problem that the model is meant to address. After creation/adaptation, the AI/ML model becomes

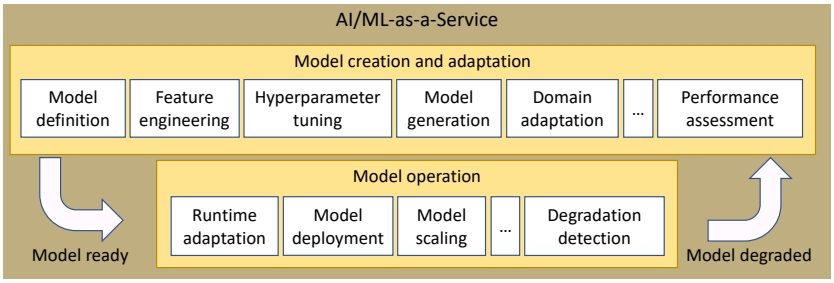


Fig. 2. Main stages and tasks involved in the AI/ML-as-a-Service pipeline.

available for use during the operation stage. The model is accessible and monitored according to the thresholds outlined in its specification.

A. Model Creation and Adaptation

The *model creation and adaptation* stage is responsible for receiving a model specification and outputting a trained AI/ML model. The model specification comprises a dataset with the definition of inputs and target feature, the AI/ML task (e.g., classification, regression), the performance metric(s) to be considered (e.g., mean squared error (MSE) for regression or accuracy for classification), and corresponding thresholds. First, the model to be used needs to be defined/selected among several options, e.g., ANNs, SVMs, and/or decision trees. Some properties of the dataset can be used to narrow down the number of potential models, but in the end, the decision is mostly empirical.

Feature engineering is the task responsible for pre-processing the dataset, selecting which features are relevant and which ones can be disregarded. The main objective of this task is to remove features that do not contribute, and sometimes even degrade, the performance of the resulting model. As a consequence of this task, data acquisition and the model itself can be simplified by keeping only the relevant features as input to the model. Usually, feature engineering strategies can be generalized for several models [25].

Hyperparameter tuning focuses on refining the model or its training parameters selected during model definition. The process involves finding the right hyperparameters with minimal computing resources. However, it can be quite expensive as the model needs to be trained multiple times with different hyperparameters. In the literature, there are several algorithms targeting hyperparameter tuning [26].

Once the the dataset has been pre-processed by the feature engineering task, and the model has been configured by hyperparameter tuning task, *model training* takes place. This task is closely associated with the particular model in use. For instance, ANNs are usually trained using backpropagation and gradient descent. On the other hand, decision trees (DTs) can be trained using a variety of methods, e.g., entropy gain, variance reduction.

Sometimes, it is important to modify a model that has deteriorated and is no longer delivering satisfactory performance, i.e., no longer meets the specified performance thresholds. Alternatively, if the model has only been trained on simulation data, it may need additional training with real-world data from the field [27]. Rather than creating a new model, another option is to adapt the existing model to the new data, a process known as *domain adaptation* [28]. Domain adaptation is performed by further training the model with the new/updated data. Alternatively, in optical networks, the learning from one model may need to be transferred to another model due to changes in, e.g., network topology [29]. In this case, the domain adaptation task can take advantage of transfer learning techniques [30].

Finally, *performance evaluation* is responsible for assessing the suitability of the model for the AI/ML task at hand. Depending on the task at hand, one can use a variety of metrics either individually or in combination when evaluating the goodness of an AI/ML model. When it comes to binary classification (i.e., between two classes), binary cross-entropy is a common choice to drive model updates, while accuracy and f1-score are used for human interpretation purposes. In the case of multi-class classification (i.e., among three or more classes), categorical cross-entropy is commonly used to update a model, while accuracy and confusion matrices are used as more interpretable metrics. When it comes to regression tasks, several error metrics can be used, such as mean absolute error (MAE), MSE, and root mean squared error (RMSE) [31]. The metric chosen for an AI/ML task depends on its purpose and how the model will be used. While default settings are available, it is best for an expert to consciously select a specific option.

B. Model Operation

Once the model is ready, the *model operation* stage is responsible for two main aspects of the model lifecycle: *(i)* making the model available for the domain-specific platform, and *(ii)* monitoring the model performance. The *runtime adaptation* is an optional task responsible for adapting the model to the intended platform where the model is running (e.g., instruction set, operating system). Depending on the case, *runtime adaptation* can be also a task of the model creation stage, or even characterize a constraint to which models are suitable for the particular use case. The task is only required if the platform for deploying the model is different from the one where it was created. It involves taking the model from the previous stage as input and generating a version that is suitable for deployment on the intended platform. Sometimes, a model is designed/trained with the x86 instruction set but needs to be used on an ARM platform. Similarly, a model may be created using graphical processing units (GPUs) but needs to run on a standard central processing units (CPUs). Another reason to adjust a model is to optimize it for the intended platform, such as compiling it into a native binary for easier and more efficient deployment.

Model deployment is the task responsible for making the model available for the customer. Ideally, the AI/ML-as-a-Service infrastructure contains specialized hardware that can be used to accelerate the execution of AI/ML models. Another way to improve the efficiency of the model is by compiling it to a native binary that can take advantage of specialized instructions of modern processors. For instance, when deploying ANNs, a common and efficient solution is to use TensorFlow-Serving [32],

which provides a URL invoking model inference through the network. For example, models are commonly made available through representational state transfer (REST) or gRPC remote procedure call (gRPC) interfaces.

Model scaling involves adjusting the resources accessible to the model to ensure that its inference time stays within a set limit. One method is to increase resources by scaling vertically, meaning more resources are allocated to the environment where the model runs. Alternatively, this can be done by horizontally scaling the resources, i.e., creating multiple replicas of the model and using a load balancer among the replicas [7, 33].

Finally, *degradation detection* is responsible for monitoring the model performance against the specified thresholds as the model is used over new data. This is done by considering the thresholds specified during the model specification, and triggering an alarm when any of the performance metrics fall outside the thresholds. There are various factors that can affect the performance of the model, including data drift or concept drift [34]. In optical networks, these drifts may be caused by equipment aging, changes in network load, or the replacement of faulty equipment. When degradation is detected, the AI/ML-as-a-Service framework should initiate the stage of creating and adapting the model to accommodate the new data.

5. AI/ML-AS-A-SERVICE FOR OPTICAL NETWORK AUTOMATION

This section introduces the proposed AI/ML-as-a-Service framework and its integration with traditional optical network automation platforms. Fig. 3 illustrates a traditional architecture and workflow for the adoption of AI/ML models in the control plane of optical networks. Additionally, the figure depicts how the control plane can be modified to utilize the suggested AI/ML-as-a-Service framework. Note that the O-SDNc in Fig. 3 represents the *domain-specific platform* mentioned in Fig. 1.

In the figure, the architecture is divided into three main planes. In the data plane, we consider an optical network composed of elements such as fiber links, reconfigurable optical add-drop multiplexers (ROADMs), Erbium-doped fiber amplifiers (EDFAs), and transceivers. In the control plane, the optical network automation platform is responsible for establishing/maintaining/terminating OChs by controlling the elements in the data plane, following a traditional SDN approach. The *inventory and monitoring* component consists of a database that stores tabular and time-series data on the services operating in the network and the monitoring of both services and infrastructure. The AI/ML plane is accountable for creating, operating, and monitoring AI/ML models.

The control plane illustrated in Fig. 3 shows a simplified view of the internal components typically found in an optical SDN platform. The management of optical services from start to finish falls under the *end-to-end optical service management* responsibility. This includes the provisioning and operation of OChs throughout their lifetime. To accomplish this goal, the control plane performs a number of tasks related to the OCh lifetime, leveraging AI/MLmodels. Among the most common tasks, we can mention QoT estimation, traffic prediction, soft-failure detection, and network sensing. Each of these tasks (and sometimes each OCh) has its own dedicated AI/MLmodels.

A traditional architecture and workflow are illustrated in Fig. 3a. In this process, human experts handle all tasks related to the AI/ML pipeline. The process begins with an external event triggering the need for a new AI/ML model to be built manually (*step 1*). Acquisition of relevant data from O-SDNc's inventory and monitoring database is performed in *step 2*. The data is then pre-processed, and a new AI/ML model is trained and validated (*step 3*). The newly generated model is deployed within the O-SDNc (*step 4*) and can be invoked by the end-to-end optical service management component (*step 5*). The interface with the model is hard-coded in the O-SDNc implementation and must be compatible with the programming language used. After *step 5*, the model monitoring starts. The outputs given by the AI/ML models is monitored (*step 6*), and later compared to the experienced values (*step 7*). If a degradation is detected, the model adaptation is requested (*step 8*), triggering a new loop (*step 2*).

The diagram in Fig. 3b demonstrates how the O-SDNc can utilize an AI/ML-as-a-Service framework with minimal adjustments. One change involves requesting a new model (*step 1*). With a standardized model descriptor, the O-SDNc can automatically request a new AI/ML model when needed. Listing 1 provides an example of a model descriptor that can be automatically populated by the O-SDNc. In *step 2*, the AI/ML-as-a-Service framework acquires data related to the model by directly querying the O-SDNc. After creation, the model is deployed within the AI/ML-as-a-Service framework (*step 3*). The O-SDNc receives a model description (*step 4*), enabling it to invoke model inference through the network (*step 5*). It is essential to note that the with this approach, the model no longer has to be deployed within the O-SDNc. It can be deployed in a separate logical execution platform (e.g., a virtual machine or container), or even in a separate datacenter. This provides extra flexibility in terms of model deployment options (i,e., potentially allowing for the optimization of other important parameters such as energy consumption) and eliminates the requirement of the O-SDNc to support an AI/ML execution platform. The *degradation detection* module monitors the inference results (*step 6*), and obtains the experienced values (*step 7*). If any performance threshold is violated, the module triggers model adaptation (*step 8*).

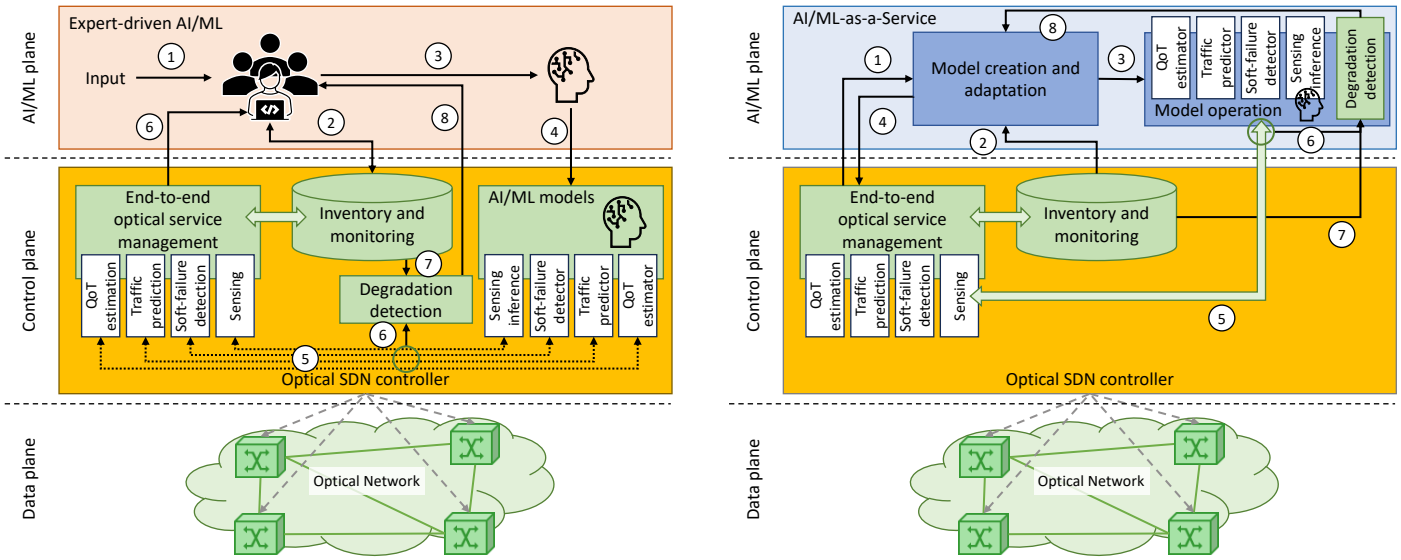
In this section, we focused on providing an overview of how an AI/ML-as-a-Service platform can be integrated with an O-SDNc. In the following section, we focus on the model creation and operation stages, which encompasses steps 1-5 of Fig. 3b. Steps 6-8 are outside of the scope of the presented implementation.

Listing 1. Example of YAML content of a model request.

```yaml
task: classification
dataset:
  provider: http
  format: pickle
  identifier: http://localhost:32000/dataset01.h5
configuration:
  time_budget_for_this_task: 480
performance:
  metrics:
    accuracy:
      threshold: 0.98
```

6. PERFORMANCE ASSESSMENT

This section evaluates the performance of a reference implementation of the model creation and operation stages of the AI/ML-as-a-Service framework introduced in the previous section. For



(a) Conventional AI/ML workflow

(b) Workflow based leveraging on AI/ML-as-a-Service

Fig. 3. A traditional architecture and workflow of AI/ML-based optical network automation compared with the one adopting the proposed AI/ML-as-a-Service framework.

this purpose, we first introduce the details of the reference implementation. Then, we describe the setup and datasets used in this evaluation. We present the performance results for two use cases, i.e., QoT regression and estimation, followed by an analysis of the inference performance of the reference implementation.

A. Reference Implementation

The AI/ML-as-a-Service framework illustrated in Fig. 3b can be implemented with any technology of choice. This section describes the languages and tools that we selected for our reference implementation of the proposed AI/ML-as-a-Service framework which focuses on the model creation and operation stages. A custom Python component (herein denoted as *coordinator*) is responsible for coordinating the main steps of the framework. The coordinator provides a REST API that the O-SDNc can use to send model requests. These model requests are expected to be in YAML format, as shown in Listing 1.

When a model request is received, the coordinator downloads the referred dataset. Currently, only Python Pickle format is supported, although other formats such as Xarray and Scikit-Learn's Bunch can be considered. The expected format should be a tuple with six elements, where three of them indicate the inputs and the other three indicate the outputs for the training, validation, and testing sets.

After the dataset is downloaded, the coordinator invokes the model creation stage. For this purpose, we have utilized the Auto-Sklearn framework [35, 36] as our engine. Auto-Sklearn utilizes the machine learning models implemented in the well-known Scikit-Learn library to construct an ensemble.

For feature pre-processing, Auto-Sklearn uses a set of methods, e.g., principal component analysis (PCA). For the regression tasks, the framework evaluates various models, such as AdaBoost, decision trees, Gaussian process, *k*-nearest neighbors, multi-layer perceptron (MLP) ANNs, random forest, and logistic regression with stochastic gradient descent (SGD) training. For the classification tasks, in addition to the previously mentioned models, the framework also evaluates linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA). Models are initialized randomly and go through the process of successive halving. Successive halving allocates a given amount of time to all models. After the time is up models with low performance are pruned, and models with high performance are given more resources (i.e., more training time). We refer the reader to [35, 36] for more details on the Auto-Sklearn framework.

The model creation stage adheres with the time budget specified in the request (if any), as illustrated in Listing 1. If no suitable model can be found within the allocated time budget, the model request fails. Since we are focusing on model creation, the definition of a threshold is not necessary. The generated ensemble model can be utilized during the inference phase to calculate a weighted average of the output from all the models that make up the ensemble. The resulting model is saved into a file system shared with the model operation stage.

The model operation stage is implemented as a custom Python component. The component utilizes Flask to offer a REST API that can process JavaScript Object Notation (JSON) format data. This allows the O-SDNc to invoke the model's inference through Web requests by specifying the model, the version, and the samples to execute the inference on. By adopting JSON, controllers that are written in any modern programming language can interact seamlessly with the model inference API. The model operation component can detect new models within the file system and load them accordingly. Once an inference request is received, the appropriate model is selected, and the inference is performed, with the result returned.

B. Setup and Datasets

In this section, we test the performance of the proposed AI/ML-as-a-Service framework using two use cases detailed in the following subsections: QoT regression and QoT classification. The goal is to determine if models generated by the AI/ML-as-a-Service framework perform similarly to manually-tuned models. QoT regression is relevant when the O-SDNc needs an AI/ML model to estimate the Optical Signal-to-Noise Ratio

(OSNR) of a new OCh based on past statistics. The OSNR estimation can be used to define which modulation format to be used or whether or not the margins are within the desired bounds [37]. QoT classification is relevant when the O-SDNc needs to know in advance (i.e., prior to OCh deployment) whether or not a given OCh configuration will work. This can be used by the O-SDNc to decide the best OCh configuration [9, 10]. In addition, we assess the runtime performance of the model generated by the AI/ML-as-a-Service framework versus an ANN under realistic usage conditions.

For both the classification and regression tasks, we adopt the four datasets reported in [10] and made available upon request in [38]. We refer to them as datasets 01–04. These datasets contain samples that represent OCh configurations as features. The datasets are obtained using the GNM analytical model from [39]. While analytical models do not capture completely the physical phenomena typical of real-world deployments, their performance shows relatively good accuracy. Moreover, adopting an open dataset enables other researchers to reproduce and improve upon the results presented in this paper.

For the QoT regression task, we used the OSNR value of the OChs as the target feature. For the QoT classification task, we used the information about the OCh status, i.e., working and not working, as the target feature. From each dataset, we selected 100,000 balanced samples, i.e., with the same number of working and not working OCh configurations. Out of the 35 features described in [10] (Table 4), we ignored only the information about the connection identifier. All the other features were adopted. Each dataset was split using 70%, 20%, and 10% of the data for training, validation, and testing purposes, respectively. For the ANN, the features are pre-processed by subtracting the mean and scaling to unit variance, the same approach adopted in [10]. For the AI/ML-as-a-Service , dataset pre-processing is not necessary since the AI/ML-as-a-Service pipeline already contains a built-in pre-processing stage. The results presented below were obtained through the use of an AMD Ryzen Threadripper 3960X 24-Core Processor, which was running Ubuntu 22.04.

We used Keras and TensorFlow to implement the ANNs. Since the inputs to the ANN are the same for the regression and classification problems, we adopted the same architecture and hyperparameters. The ANN has 34 inputs, 256 neurons using a tanh activation function in the hidden layer, and a single output neuron. For the regression use case, the output neuron uses a linear activation function. For the classification use case, the output neuron uses a sigmoid activation function, similar to [10]. Both ANNs are trained using RMSprop with a learning rate of 0.01.

C. Use Case 1: QoT Regression

In this use case, the O-SDNc needs an AI/ML model to estimate the OSNR of a new OCh based on past statistics. We adopt mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE) to characterize the performance of the models. Both ANN and AI/ML-as-a-Service use the RMSE as the loss function, i.e., the training objective is to minimize the RMSE. Listing 2 shows the YAML descriptor used to trigger the AI/ML-as-a-Service pipeline for this use case. Both models were given a 1,440 seconds budget for the training. This number was chosen based on an empirical experiment that observed the convergence time over dataset 01 with a budget of two hours.

Table 1 summarizes the performance of the ANN and AI/ML-as-a-Service model. In terms of MAE, the models have relatively close performance. The AI/ML-as-a-Service model achieves slightly better performance in datasets 02, 03, and 04, while the ANN performs better in dataset 01. When it comes to MSE, the trend is the opposite, with the ANN having better performance except for dataset 01. RMSE shows the highest differences between the two models, with the AI/ML-as-a-Service model performing the best across all datasets. In summary, the performance of the two models is relatively close, and the difference in accuracy should result in minimal differences in performance when the models are deployed and used during network operation.

Figure 4 shows the accuracy of the trained model over the testing set of dataset 01. We can observe that with an OSNR range of 15 to 25 dB, the two models present similar accuracy performance, with the AI/ML-as-a-Service doing slightly better. However, in the 25 to 35 dB OSNR range, the ANN is not able to estimate the OSNR value precisely. Analyzing Fig. 4 and Table 1, we can observe that only the RMSE metric captured the degraded performance of the ANN in the higher OSNR range. This justifies the use of several metrics to characterize the performance of AI/ML models in regression tasks.

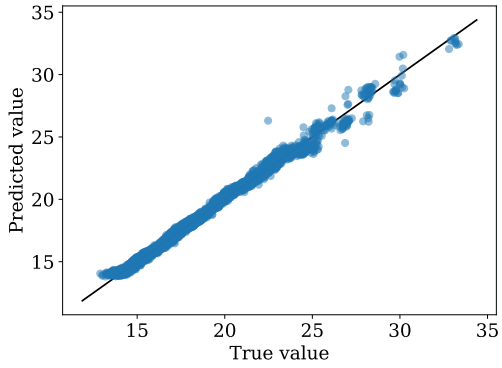
Table 2 shows the Scikit-Learn models that compose the final ensemble selected by our AI/ML-as-a-Service implementation for the regression use case. The cost is measured in terms of the loss function, i.e., RMSE in this case. The weight represents how much a specific model impacts the final output of the ensemble. The ensembles are composed of one or two models. In most cases, gradient boosting was the best model found. However,

Listing 2. YAML file used to trigger the regression pipeline for dataset 01.

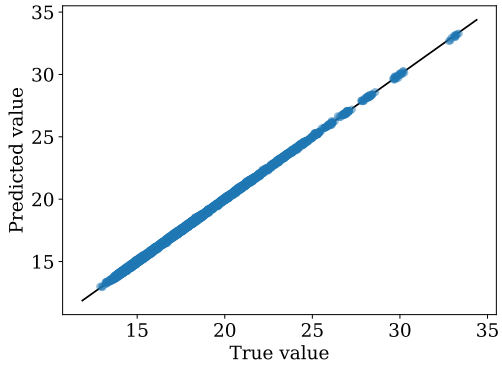
```
task: regression
dataset:
  provider: http
  format: pickle
  identifier: http://localhost:32000/ds01_reg.h5
configuration:
  time_budget_for_this_task: 1440
  loss: mean_squared_error
```

Table 1. Regression performance using the testing dataset.

Dataset	Metric	ANN	AI/MLaaS
01	MAE	0.061	0.061
	MSE	0.006	0.006
	RMSE	0.100	0.003
02	MAE	0.061	0.060
	MSE	0.007	0.077
	RMSE	0.087	0.006
03	MAE	0.104	0.041
	MSE	0.017	0.051
	RMSE	0.131	0.002
04	MAE	0.089	0.058
	MSE	0.012	0.073
	RMSE	0.111	0.005



(a) Artificial neural network (ANN)



(b) AI/ML-as-a-Service

Fig. 4. Predicted OSNR [dB] using the testing samples in dataset 01.

note that even if the ensemble has two gradient boosting models, they have different hyperparameters, and may contribute differently to the final performance. Finally, for dataset 04, an Ada Boost model was used in addition to gradient boosting.

In summary, we can observe that model resulting from the proposed AI/ML-as-a-Service framework has performance close to the ones presented by manually-tuned ANNs. However, AI/ML-as-a-Service does not require any human intervention in the model selection, training, and validation phases.

D. Use Case 2: QoT Classification

In this use case, the AI/ML model estimates whether or not a particular OCh configuration will work. This task can be mapped to a binary classification problem. We adopt accuracy as the only metric to characterize the performance of the models under exam. During training and testing, we also measured precision, recall, and f1 score, which showed similar trends as accuracy. Listing 3 shows the YAML descriptor used to trigger the AI/ML-as-a-Service framework for this use case. Both models were given 480 seconds budget for the training. Similarly to the regression use case, this number was chosen based on empirical experiments that observed the convergence time over dataset 01 with a budget of two hours.

Table 3 summarizes the accuracy of the models. We also include the results reported by [10] (Table 6) for comparison purposes. We highlight that the results from [10] cannot be directly compared to the ones obtained in our experiments due

Table 2. Models comprised in the regression ensemble built by the AI/ML-as-a-Service framework.

Dataset	Rank	Model	Cost	Weight
01	1	HGBCT*	0.0037	0.74
	2	HGBCT*	0.0040	0.26
02	1	HGBCT*	0.0058	1.0
03	1	HGBCT*	0.0026	0.84
	2	HGBCT*	0.0029	0.16
04	1	HGBCT*	0.0057	0.7
	2	Ada Boost	0.0070	0.3

* HGBCT: Histogram-based Gradient Boosting Classification Tree

Listing 3. YAML file used to trigger the classification pipeline for dataset 01.

```
task: regression
dataset:
  provider: http
  format: pickle
  identifier: http://localhost:32000/ds01_class.h5
configuration:
  time_budget_for_this_task: 480
```

to potential differences in the samples used for the experiments. Nevertheless, the results obtained by our ANN follow the same trend as the ones reported in [10], i.e., the relative accuracy between datasets is the same. We observe that in most cases, both models achieve classification accuracy above 99%. However, we observe that for dataset 01, the model resulting from the AI/ML-as-a-Service framework achieves 99% accuracy, while the ANN is unable to achieve the same accuracy level.

Table 4 shows a representative set of the Scikit-Learn models that compose the final ensemble selected by our AI/ML-as-a-Service implementation. The cost is measured as the loss function, i.e., binary cross-entropy in this case. The weight represents how much each model impacts the final output of the ensemble. Unlike the regression case, in the classification use case, the ensemble is composed of at least 6 models. They include gradient boosting, extra trees, and stochastic gradient descent classifiers. For the sake of space, Table 4 includes only the three models with the highest weight. For dataset 01, the final ensemble is composed of ten models. For datasets 02 and 03, eight models composed the final ensemble. For dataset 04, only six models were used in the ensemble.

In summary, the results show that the models resulting from the AI/ML-as-a-Service implementation are equivalent to the ones resulting from manually-tuned models. Moreover, the implementation takes advantage of several AI/ML models to achieve suitable performance.

E. Inference Performance

Having good accuracy is only one of the attributes that make a AI/ML model suitable for real-world applications. Another important attribute is its inference performance, i.e., how much time it takes to perform an inference. To assess the performance of the generated models in this respect, we designed a test to

Table 3. Summary of the classification performance for the testing dataset.

Dataset	ANN [10]	ANN	AI/MLaaS
01	0.984	0.976	0.990
02	0.994	0.994	0.997
03	0.990	0.991	0.993
04	0.991	0.992	0.996

Table 4. Top three models comprised in the classification ensemble built by the AI/ML-as-a-Service framework.

Dataset	Rank	Model	Cost	Weight
01	1	HGBCT*	0.011	0.18
	2	HGBCT*	0.151	0.16
	3	SGD**	0.377	0.12
02	1	Extra Trees	0.003	0.28
	2	SGD**	0.267	0.10
	3	HGBCT*	0.003	0.06
03	1	SGD**	0.135	0.64
	2	HGBCT*	0.007	0.06
	3	Extra Trees	0.007	0.04
04	1	HGBCT*	0.004	0.22
	2	Extra Trees	0.004	0.20
	3	HGBCT*	0.031	0.12

* HGBCT: Histogram-based Gradient Boosting Classification Tree

** SGD: Stochastic Gradient Descent

measure their inference time. We used the QoT classification model with 10,000 samples from the testing set of dataset 01. In real deployments, it is unlikely that multiple OCh requests will arrive at the same time. As a result, few inferences are needed over time. To replicate this scenario, we introduced a random pause between inferences whose value is drawn uniformly in the range of $(0,\ 1)$ seconds to represent a realistic provisioning scenario. We then measured the time taken by each inference.

On average, the ANN inference takes 40 ms given the available hardware. This time could be further improved if the inferences were made in batches using an accelerator such as a GPU, but for our use case, inferences will be sparse over time. The model returned by the AI/ML-as-a-Service framework takes on average 553 ms per inference. This represents a $14\times$ increase in inference time. When the inference is executed over the network through the REST API running in the same computer, we observe an overhead of 10ms for both models. Note that none of the models were fine-tuned for performance, hence there exist optimizations that can be done to increase their inference performance.

Part of this higher time taken by the model generated by our implementation is explained by the ensemble of models. The ensemble executes several internal inferences before a final output is computed. Another reason is that the model generated by our implementation runs using Python, i.e., it has not been compiled to native code. On the other hand, the ANN is executed using a native implementation below the Python front end. One way to optimize the output of the AI/ML-as-a-Service implementation is to implement a similar ensemble in a language that compiles to a native binary. Then, the ensemble could be imported to the native implementation by adopting the same hyperparameter values. Nevertheless, these AI/ML-as-a-Service models are expected to be used over the network, e.g., as illustrated in Fig. 3b. This means that the O-SDNc and the AI/ML-as-a-Service server might be running far away from each other. Therefore, inference time might experience a high network latency, reducing the relative contribution of the model inference time.

7. OPEN CHALLENGES

To the best of our knowledge, this is the first paper to introduce the concept of AI/ML-as-a-Service for optical network automation. The results obtained illustrate that AI/ML-as-a-Service has has potential. On the other hand, several challenges still remain open.

The first challenge is related to the availability of data, which impacts not only the study of AI/ML-as-a-Service, but the study of general AI/MLapplied to optical networks. While the datasets in [10] are helpful for QoT regression and classification, we need more datasets for essential tasks such as anomaly and soft-failure detection, prediction and localization. Moreover, it is crucial to have datasets collected from real-world deployments.

The second challenge is related to the explainability and trustworthiness required for the widespread adoption of AI/ML. Nowadays, the explainability of AI/ML is already an key aspect to be explored. AI/ML-as-a-Service will add another layer between the AI/ML model and the network operator, increasing the need for explainability. This means that not only AI/ML-as-a-Service needs to generate explainable models, but it also needs explainability for its internal processes.

Moreover, the inference performance results show that there is a gap in performance for the models generated with the proposed AI/ML-as-a-Service framework. Further improvements can be made via runtime adaptation and/or taking advantage of AI/ML accelerators.

Another open challenge is the study of degradation detection and model adaptation. These operations are challenging due to the scarcity of suitable datasets. Moreover, different use cases may require different thresholds and strategies to detect model degradation. Finally, different AI/ML models will require different strategies to reduce the overhead of model adaptation.

Finally, the reference implementation presented in this paper has been tested over datasets with input composed of a few tens of numerical features. Adding to the implementation the ability to process graphical representations of the optical signal (e.g., eye diagrams) and/or time series data remain a challenge and are left for future work.

8. FINAL REMARKS

This paper proposed an AI/ML-as-a-Service framework aiming at streamlining the process of creating and operating artificial intelligence/machine learning (AI/ML) models used by optical network automation platforms. We detailed the main concepts related to AI/ML-as-a-Service, and the specific tasks involved in the AI/ML model pipeline. The integration and

interfaces necessary to integrate the proposed framework with existing optical network automation platforms have been described. Focusing on the model creation and operation stages, the performance of the proposed framework was benchmarked using two representative use cases in optical networks: quality-of-transmission (QoT) regression and QoT classification. The accuracy performance results show that the models generated by the AI/ML-as-a-Service framework have performance equivalent to the ones achieved by manually-tuned artificial neural networks (ANNs). The inference performance results show that optimizations made in the ANN inference make it hard for unoptimized models (such as the ones from our AI/ML-as-a-Service implementation) to reach such a performance. On the other hand, a degradation of a few ms does not represent a critical aspect of the model. It is important to highlight, however, that human intervention is still necessary for a few steps in the process, such as data acquisition.

FUNDING

This work was supported by Chalmers' ICT-AoA seed project "Auto5G" and by Sweden's innovation agency VINNOVA, within the framework of the EUREKA cluster CELTIC-NEXT project AI-NET-PROTECT (2020-03506).

DISCLOSURES

The authors declare no conflict of interest.

REFERENCES

1. R. Casellas, R. Martinez, R. Vilalta, R. Munoz, A. Gonzalez-Muniz, O. G. de Dios, and J.-P. Fernandez-Palacios, "Advances in SDN control and telemetry for beyond 100g disaggregated optical networks [invited]," J. Opt. Commun. Netw. **14**, C23–C37 (2022). DOI: 10.1364/JOCN.451516.
2. F. Paolucci, A. Sgambelluri, F. Cugini, and P. Castoldi, "Network telemetry streaming services in SDN-based disaggregated optical networks," J. Light. Technol. **36**, 3142–3149 (2018). DOI: 10.1109/JLT.2018.2795345.
3. D. Rafique and L. Velasco, "Machine learning for network automation: overview, architecture, and applications [invited tutorial]," J. Opt. Commun. Netw. **10**, D126–D143 (2018). DOI: 10.1364/JOCN.10.00D126.
4. S. Aladin, A. V. S. Tran, S. Allogba, and C. Tremblay, "Quality of transmission estimation and short-term performance forecast of lightpaths," J. Light. Technol. **38**, 2807–2814 (2020). DOI: 10.1109/JLT.2020.2975179.
5. L. Zhang, X. Li, Y. Tang, J. Xin, and S. Huang, "A survey on QoT prediction using machine learning in optical networks," Opt. Fiber Technol. **68**, 102804 (2022). DOI: 10.1016/j.yofte.2021.102804.
6. M. Ibrahimi, H. Abdollahi, C. Rottondi, A. Giusti, A. Ferrari, V. Curri, and M. Tornatore, "Machine learning regression for QoT estimation of unestablished lightpaths," J. Opt. Commun. Netw. **13**, B92–B101 (2021). DOI: 10.1364/JOCN.410694.
7. C. Manso, R. Vilalta, R. Munoz, N. Yoshikane, R. Casellas, R. Martinez, C. Wang, F. Balasis, T. Tsuritani, and I. Morita, "Scalability analysis of machine learning QoT estimators for a cloud-native SDN controller on a WDM over SDM network," J. Opt. Commun. Netw. **14**, 257–266 (2022). DOI: 10.1364/JOCN.449009.
8. Y. Fan, X. Pang, A. Udalcovs, C. Natalino, L. Zhang, V. Bobrovs, R. Schatz, X. Yu, M. Furdek, S. Popov, and O. Ozolins, "Feedforward neural network-based EVM estimation: Impairment tolerance in coherent optical systems," IEEE J. Sel. Top. Quantum Electron. **28**, 1–10 (2022). DOI: 10.1109/JSTQE.2022.3177004.
9. C. Rottondi, L. Barletta, A. Giusti, and M. Tornatore, "Machine-learning method for quality of transmission prediction of unestablished lightpaths," J. Opt. Commun. Netw. **10**, A286–A297 (2018). DOI: 10.1364/JOCN.10.00A286.
10. G. Bergk, B. Shariati, P. Safari, and J. K. Fischer, "ML-assisted QoT estimation: a dataset collection and data visualization for dataset quality evaluation," J. Opt. Commun. Netw. **14**, 43–55 (2022). DOI: 10.1364/JOCN.442733.
11. S. Shahkarami, F. Musumeci, F. Cugini, and M. Tornatore, "Machine-learning-based soft-failure detection and identification in optical networks," in *Optical Fiber Communications Conference and Exposition (OFC),* (2018), p. M3A.5.
12. T. Panayiotou, S. P. Chatzis, and G. Ellinas, "Leveraging statistical machine learning to address failure localization in optical networks," J. Opt. Commun. Netw. **10**, 162–173 (2018). DOI: 10.1364/JOCN.10.000162.
13. F. N. Khan, K. Zhong, W. H. Al-Arashi, C. Yu, C. Lu, and A. P. T. Lau, "Modulation format identification in coherent receivers using deep machine learning," IEEE Photonics Technol. Lett. **28**, 1886–1889 (2016). DOI: 10.1109/LPT.2016.2574800.
14. C. Natalino, A. Udalcovs, L. Wosinska, O. Ozolins, and M. Furdek, "Spectrum anomaly detection for optical network monitoring using deep unsupervised learning," IEEE Commun. Lett. **25**, 1583–1586 (2021). DOI: 10.1109/LCOMM.2021.3055064.
15. C. Natalino, M. Schiano, A. Di Giglio, L. Wosinska, and M. Furdek, "Experimental study of machine-learning-based detection and identification of physical-layer attacks in optical networks," J. Light. Technol. **37**, 4173–4182 (2019). DOI: 10.1109/JLT.2019.2923558.
16. R. Philipp, A. Mladenow, C. Strauss, and A. Völz, "Machine learning as a service: Challenges in research and applications," in *Proc. 22nd International Conference on Information Integration and Web-Based Applications & Services,* (New York, NY, USA, 2021), iiWAS '20, pp. 396–406. DOI: 10.1145/3428757.3429152.
17. U. Moghe, P. Lakkadwala, and D. K. Mishra, "Cloud computing: Survey of different utilization techniques," in *CSI Sixth International Conference on Software Engineering (CONSEG),* (2012), pp. 1–4. DOI: 10.1109/CONSEG.2012.6349524.
18. H. Lee, J. Cha, D. Kwon, M. Jeong, and I. Park, "Hosting AI/ML workflows on O-RAN RIC platform," in *IEEE Globecom Workshops (GC Wkshps),* (2020), pp. 1–6. DOI: 10.1109/GCWkshps50303.2020.9367572.
19. J. Baranda, J. Mangues-Bafalluy, E. Zeydan, C. Casetti, C. F. Chiasserini, M. Malinverno, C. Puligheddu, M. Groshev, C. Guimarães, K. Tomakh, D. Kucherenko, and O. Kolodiazhnyi, "Demo: AIML-as-a-Service for SLA management of a digital twin virtual network service," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),* (2021), pp. 1–2. DOI: 10.1109/INFOCOMWKSHPS51825.2021.9484610.
20. C. Natalino, N. Mohammadiha, and A. Panahi, "Machine-learning-as-a-service for optical network automation," in *2023 Optical Fiber Communications Conference and Exhibition (OFC),* (2023), p. W4G.3. DOI: 10.1364/OFC.2023.W4G.3.
21. B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Fifth International Joint Conference on INC, IMS and IDC,* (2009), pp. 44–51. DOI: 10.1109/NCM.2009.218.
22. M. Ribeiro, K. Grolinger, and M. A. Capretz, "MLaaS: Machine learning as a service," in *IEEE 14th International Conference on Machine Learning and Applications (ICMLA),* (2015), pp. 896–902. DOI: 10.1109/ICMLA.2015.152.
23. X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," Knowledge-Based Syst. **212**, 106622 (2021). DOI: 10.1016/j.knosys.2020.106622.
24. D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (MLOps): Overview, definition, and architecture," IEEE Access **11**, 31866–31879 (2023). DOI: 10.1109/ACCESS.2023.3262138.
25. J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," ACM Comput. Surv. **50** (2017). DOI: 10.1145/3136625.
26. L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," Neurocomputing. **415**, 295–316 (2020). DOI: 10.1016/j.neucom.2020.07.061.
27. 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). DOI:

10.1364/JOCN.11.000226.
28. A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia, "A brief review of domain adaptation," in *Advances in Data Science and Information Engineering,* R. Stahlbock, G. M. Weiss, M. Abou-Nasr, C.-Y. Yang, H. R. Arabnia, and L. Deligiannidis, eds. (Springer International Publishing, Cham, 2021), pp. 877–894.
29. J. Yu, W. Mo, Y.-K. Huang, E. Ip, and D. C. Kilper, "Model transfer of QoT prediction in optical networks based on artificial neural networks," J. Opt. Commun. Netw. **11**, C48–C57 (2019). DOI: 10.1364/JOCN.11.000C48.
30. F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," Proc. IEEE **109**, 43–76 (2021). DOI: 10.1109/JPROC.2020.3004555.
31. T. Panayiotou, M. Michalopoulou, and G. Ellinas, "Survey on machine learning for traffic-driven service provisioning in optical networks," IEEE Commun. Surv. & Tutorials **25**, 1412–1443 (2023). DOI: 10.1109/COMST.2023.3247842.
32. C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh, and J. Soyke, "Tensorflow-serving: Flexible, high-performance ML serving," arXiv preprint arXiv:1712.06139 (2017).
33. C. Natalino, L. Gifre, F.-J. Moreno-Muro, S. Gonzalez-Diaz, R. Vilalta, R. Munoz, P. Monti, and M. Furdek, "Flexible and scalable ML-based diagnosis module for optical networks: a security use case," J. Opt. Commun. Netw. **15**, C155–C165 (2023). DOI: 10.1364/JOCN.482932.
34. D. M. Manias, A. Chouman, and A. Shami, "Model drift in dynamic networks," IEEE Commun. Mag. pp. 1–7 (2023). DOI: 10.1109/MCOM.003.2200306.
35. M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Proc. of NeurIPS,* (2015).
36. M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-Sklearn 2.0: Hands-free AutoML via meta-learning," J. Mach. Learn. Res. **23**, 1–61 (2022).
37. M. Lonardi, J. Pesic, T. Zami, E. Seve, and N. Rossi, "Machine learning for quality of transmission: a picture of the benefits fairness when planning WDM networks," J. Opt. Commun. Netw. **13**, 331–346 (2021). DOI: 10.1364/JOCN.433412.
38. G. Bergk, B. Shariati, P. Safari, and J. K. Fischer, "QoT dataset collection," https://www.hhi.fraunhofer.de/networkdata (2022).
39. P. Poggiolini, "The GN model of non-linear propagation in uncompensated coherent optical systems," J. Light. Technol. **30**, 3857–3879 (2012). DOI: 10.1109/JLT.2012.2217729.