Demonstration of a Compositional Learning Framework for Open and Disaggregated Optical Network Control

Huy Quang Tran[1,4,*,§], Javier Errea[1,3,§], Huu Trung Thieu[2], Quan Pham Van[2], Nakjung Choi[2], Dominique Verchere[1], Adlen Ksentini[3] and Djamal Zeghlache[4]

[1] *Nokia Bell Labs, 12 rue Jean Bart, 91300 Massy, France;* [2] *Nokia Bell Labs, Murray Hill, US*
[3] *EURECOM Research Institute, Biot, France*
[4] *Telecom SudParis, Institut Polytechnique de Paris, France*

[*] *huy.h.tran@nokia-bell-labs.com*

Abstract: We introduce an automated Compositional Learning Framework, which can dynamically combine ML models to create a composite ML service. It leverages the MLOps principle to streamline drift-aware ML workflows. We showcase its applicability in the dynamic Routing Modulation and Spectrum Allocation scenario with an open disaggregated control platform.

1. Overview

As the need for increased capacity in B5G and 6G applications continues to rise, optical networks are undergoing a transformation towards greater dynamism and disaggregation. This evolution involves integrating intent-based autonomous operations facilitated by expanding artificial intelligence (AI) and machine learning (ML) within optical networks. The wide-ranging AI/ML applications of this integration are outlined in [1]. Despite diverse applications, the majority of present ML models in optical networking are designed for specific tasks, utilizing a single deployed model on the ML Operations (MLOps) platform [2]. MLOps is a set of practices that aim to streamline and automate the end-to-end ML lifecycle from data preparation; model development, training, and validation; to deployment and monitoring, known as ML pipeline. As these task-specific ML models grow in complexity over time, the demand for cloud resources for training and deployment also escalates, posing challenges in the deployment and scaling of these models.

On the other hand, two or more ML models/services can be integrated into a series of tasks, where each task corresponds to a specific ML model, contributing to the execution of a complex operation. Breaking down intricate ML tasks into simpler components and then composing them (sequentially or in parallel) into a complete operation is known as Compositional Learning (CL) [3]. This approach envisions a smooth assembly of existing AI/ML models or services, favoring re-usability, to create a novel, composite AI/ML service capable of tackling complex use cases effectively. However, several challenges associated with CL remain unaddressed, including (i) auto-decomposing and distributing ML models, (ii) dynamically composing/stitching multiple ML models, (iii) automatically handling ML model drift, and (iv) runtime ML pipeline programmability (i.e., the ability to change one or multiple models in the ML model chain). These challenges collectively form the requirement of intent-based CL operations. Consequently, there is no prototype dealing with this crucial aspect.

To address this, we present and demonstrate for the first time a Compositional Learning Framework capable of dynamically stitching multiple ML models, handling the ML model drift, and enabling the programmable ML runtime in the dynamic Routing Modulation and Spectrum Allocation (RMSA) scenario.

2. Innovation

To streamline the development and execution of CL-related applications within the control and management plane, a modular workflow-based Compositional Learning Framework (CLF) has been created based on the microservice architecture. This framework is designed to address the challenges associated with CL and serves as a foundation for intent-based CL operations. As shown in Fig.1a, the framework consists of a CL and ML Pipeline Manager (CPM), a CL Stitching Engine, a CL Workflow Execution Engine, an MLOps Adapter, and a CL Graphical User Interface (GUI). CPM is responsible for coordinating the entire lifecycle of ML model chains, exposing CL features to be consumed by external entities such as the SDN controller and its applications. The CL Stitching Engine guides and validates the stitching of multiple ML models at design time. The CL Workflow Execution Engine is the run-time engine responsible for executing the CL workflows generated during design time. The

[§] These authors have equally contributed to this work.

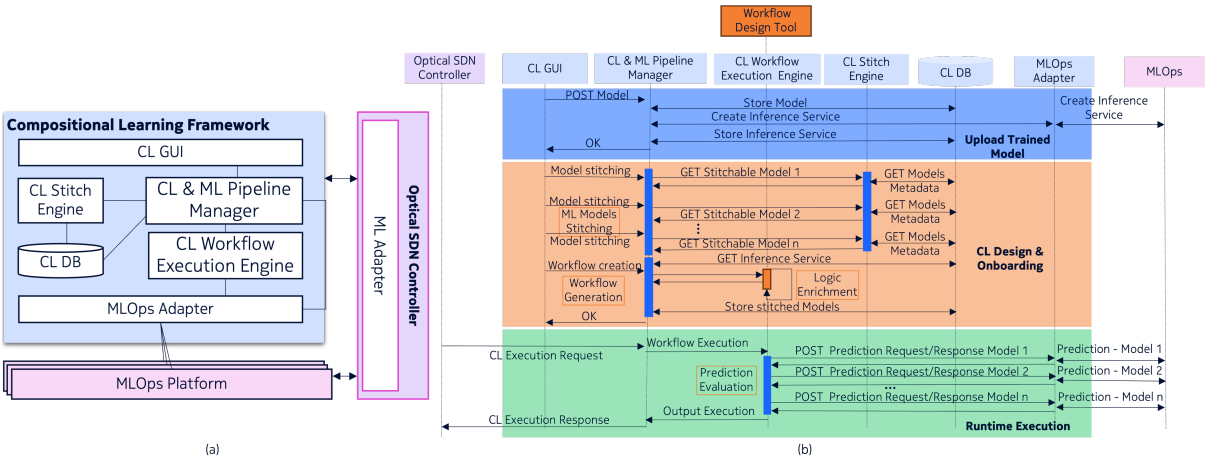

Fig. 1: (a) Compositional Learning Framework; (b) Compositional Learning Workflow

CPM manages the distribution of ML pipelines on various MLOps platforms. The MLOps adapter is a mediation component that translates common ML APIs into specific MLOps APIs. Finally, a CL GUI is designed to enhance the user experience and simplify CL stitching and management. The CL GUI provides a unified portal to enhance the user experience and simplify the CL stitching and management. The CLF is running on top of MLOps platforms to enhance the efficiency, reliability, and scalability of ML pipelines; while encompassing version control, continuous integration and deployment, model testing, and constant monitoring and maintenance. In addition, the CLF relies on MLOps to automatically monitor, evaluate, and detect drifts in the performance of each model in the chain through the execution of ML pipelines.

The operational workflow of the CLF is shown in Fig. 1b. First, we have the Upload Trained Model stage, which involves uploading an onboarding package to the CPM. This onboarding package consists of the model, a set of artifacts supporting the model, a manifest containing the model metadata such as versioning information, required libraries, candidate models to be combined with the onboarded model, etc., and a set of input and output parameter types. The onboarding preparation step does not require prior domain-specific knowledge, e.g., optical network systems or MLOps; thus, simplifying significantly the user experience. Second, the CL Design and Onboarding stage is executed during the design time to construct an ML model chain for a particular use case. Finally, we enter the Runtime Execution Stage to use the inference ML services to serve the use-case.

We demonstrate and validate the key features of the Compositional Learning Framework through its implementation for the drift-aware CL-based dynamic Resource Allocation use-case.

3. OFC Relevance

This demonstration will introduce a novel architectural design that implements Compositional Learning through seamless interaction among three key components: (i) the CL Framework, (ii) the Cloud-native Optical Transport Control Platform, and (iii) the MLOps platform. This innovative architecture enables ML model stitching, orchestration, and automated life-cycle management to control the optical network. Additionally, these tools will become an inevitable part of the development journey towards AI-native solutions for the realization of optical network automation, as they bring significant benefits for ML model designers, vendors, telco, and cloud operators.

4. Demo content and implementation

The demonstration is performed live on our remote physical workbench of Nokia 1830 equipment. We use the Cloud-native Optical Transport Control Platform in [4] to manage a network of four PSS-32 nodes (see Fig. 2a). Furthermore, the platform is extended with the CLF and two MLOps platforms. Each ML model has been trained, served, and monitored in its dedicated ML pipeline prior to the demo. The demo is described as follows:

Step 1: *Design and Onboarding phase* - The user navigates through the ML model catalog on the CL GUI. In this demonstration, the user selects the RMSA Deep Reinforcement Learning (DRL) model as the main actor to solve the dynamic Resource Allocation problem. The DRL agent performs online learning with the network state provided by the SDN Controller (SDNC) using RLOps principles [5]. Furthermore, to enhance the performance, the CPM recommends various combinations of the selected model with others and how to assemble them to form a composite ML service. The Link Utilization and Fragmentation (LUF) Forecaster and the Quality of Transmission (QoT) Classifier are recommended to combine with the RMSA DRL model (DeepLUF-RMSA) to form a composite ML service chaining. Then, the user chooses the components, stitches them, and sends the final configuration to the CPM for deployment. The CPM triggers the ML pipeline of each model on corresponding

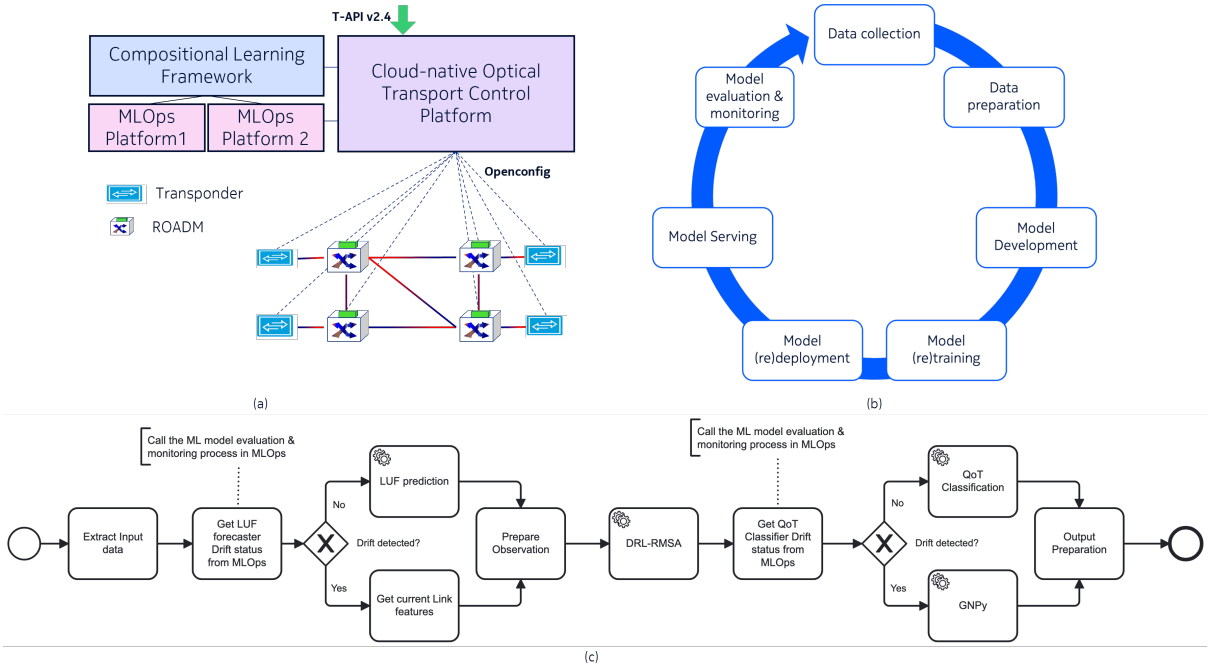

Fig. 2: (a) Optical Control Platform and Testbed; (b) MLOps pipeline ; (c) CL-based resource allocation workflow

MLOps platforms and deploys their inference services in the Cloud-native Optical Control Platform.

Step 2: *Inference phase* - The user creates a TAPI connectivity service request via the portal of the SDNC. The request is forwarded to the CPM along with other network state information. The CPM invokes the LUF Forecaster to predict the future link utilization and fragmentation of the network. The forecast LUF is processed with the set of candidate paths and current topology details to form the observation for the agent in DeepLUF-RMSA. The CPM receives the selection of DeepLUF-RMSA and combines it with monitoring data from the network to serve as the input to the QoT Classifier, which checks the feasibility of the selected lightpath. The CPM sends the result of the composite Resource Allocation task to the SDNC. Finally, the SDNC provisions the service on corresponding network elements and collects the performance monitoring (PM) data. The PMs are sent back to the CPM to assess the performance of the models based on their prediction accuracy. This complete workflow is shown in Fig. 2c, where a task or sub-task can correspond to an ML model chain inference service.

Step 3: *Monitoring phase* - We continue to serve more requests and monitor models' performance. All monitoring data of models are shown on a dedicated dashboard. Over time, we simulate a drift event in the QoT Classifier where its prediction accuracy falls below a given threshold. The CPM triggers a drift adaptation mechanism to execute a complete ML pipeline as in [6]. In the meantime, the CPM updates the composite ML service chain to let the SDNC use GNPy [7] as an alternative solution for QoT validation.

Step 4: *Recovery phase* - The ML pipeline to recover the QoT Classifier performance contains the following main steps: (i) processing on newly collected data; (ii) retraining of the model; (iii) evaluating model performance; (iv) serving the model (see Fig. 2b). Once the performance of the QoT Classifier surpasses the threshold, the CPM rolls back to the initial ML service chain automatically. The Optical Control Platform can continue serving the service requests seamlessly.

Acknowledgement. This work is partly supported by the European Commission via the Horizon Europe/JU SNS project Hexa-X-II (GA no. 101095759)

References

1. F. Musumeci et al. *IEEE Commun. Surv. Tutor.*, 21(2):1383–1408, 2019.
2. D. Kreuzberger et al. Machine learning operations (mlops): Overview, definition, and architecture. *IEEE Access*, 2023.
3. Debmalya Biswas. Compositional ai: Fusion of ai/ml services. Technical report, 03 2021.
4. Javier Errea et al. Open disaggregated optical network control with network management as code. In *2023 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, 2023.
5. Peizheng Li et al. Rlops: Development life-cycle of reinforcement learning aided open ran. *IEEE Access*, 2022.
6. Huy Quang Tran et al. Dynamic drift-adaptive ensemble-based quality of transmission classification framework in otn. In *2023 23rd International Conference on Transparent Optical Networks (ICTON)*, pages 1–4, 2023.
7. Alessio Ferrari et al. Gnpy: an open source planning tool for open optical networks. In *2020 International Conference on Optical Network Design and Modeling (ONDM)*, pages 1–6, 2020.