Thèse présentée pour l'obtention du grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Spécialité

Ingénierie / Systèmes Informatiques

École doctorale

Informatique, Télécommunication et Électronique Paris (ED130)

# Automated Control of Future Open and Disaggregated Optical Networks aided by Machine Learning Methods

---

Javier Errea Moreno

Soutenue publiquement le : *18 décembre 2025*

Devant un jury composé de :

| | |
|---|---|
| **Kandaraj PIAMRAT**, Maître de conférences, Nantes University | *Rapporteur* |
| **Yassine HADJADJ-AOUL**, Professeur, University of Rennes | *Rapporteur* |
| **Djamal ZEGHLACHE**, Professeur, Télécom SudParis | *Président du Jury* |
| **Esther LE ROUZIC**, Docteur, Orange | *Examinateur* |
| **Adlen KSENTINI**, Professeur, EURECOM Research Institute | *Directeur de thèse* |
| **Dominique VERCHERE**, Docteur, NOKIA Bell-Labs | *Co-directeur de thèse* |

*À mes parents et à Uxue,*
*pour leur soutien constant et leur patience.*

# Acknowledgements

This thesis marks the end of a long and challenging journey, one that I would not have been able to complete without the support of many people to whom I am deeply grateful.

First, I wish to sincerely thank my supervisors for their guidance, patience, and encouragement throughout these years. Their advice has been instrumental in shaping not only this manuscript, but also my growth as a researcher.

I am also indebted to my colleagues at Nokia, whose collaboration and friendship have made this path both productive and enjoyable. In particular, I want to thank Huy Tran, with whom I shared many intense discussions, late problem-solving sessions, and the satisfaction of seeing our ideas take shape.

Beyond the academic environment, I am forever grateful to my parents. Their unwavering support and belief in me have carried me through every step of this journey. They know better than anyone the effort behind this thesis, and they stood by me during both the moments of excitement and the inevitable times of doubt.

Finally, my heartfelt thanks go to my partner, Uxue. She has lived every part of this process with me, sharing the highs and the lows, celebrating the small victories, and giving me strength when the challenges felt overwhelming. Without her patience, love, and encouragement, this work would simply not have been possible.

To all of you: thank you.

# Abstract

Optical transport networks are evolving from closed and vertically integrated systems into open and disaggregated architectures capable of accommodating diverse services while adapting to rapidly increasing and variable traffic demands. This evolution is enabled by programmable hardware, standardized data models, and cloud-native control platforms. The thesis proposes a breakthrough microservice architecture that allows combining open and disaggregated optical systems, integrating GitOps and Network as Code practices to achieve automation, continuous integration, and lifecycle management of optical network control functions.

Building on this foundation, this research explores how Machine Learning (ML) and Reinforcement Learning (RL) can be applied to address key control and management challenges in optical networks. This work focuses on routing and spectrum assignment, data traffic and optical network state and resource prediction, quality of transmission estimation, and adaptive transponder configuration optimization. To enhance the application of ML and RL in optical networks, a Compositional Machine Learning (CML) framework is introduced, which decomposes complex tasks into modular ML models that can be combined and stitched as optical control function pipelines. This approach improves scalability, interpretability, and reuse of ML-driven control functions. In addition, Machine Learning Operations (MLOps) methods are incorporated, which involve automated retraining, drift detection, and continuous monitoring, into the CML framework to ensure robust, resilient, and adaptive model operation under dynamic network conditions.

By bridging cloud-native architectures with ML-driven control, this thesis advances the integration of automation and intelligence in optical networks. It also outlines future perspectives, including cross-layer IP/optical coordination, self-healing closed loops, and AI-native optical control, paving the way towards autonomous, resilient, and sustainable optical networks.

**Keywords:** Open and Disaggregated Optical Networks; Network Management as Code; Autonomous Optical Networks; Machine Learning; Reinforcement Learning; ML-driven Optical Network Control; Compositional ML; AI-Native Optical Networks; Self-Driven Optical Networks.

# Résumé

Les réseaux de transport optique évoluent de systèmes fermés et verticalement intégrés vers des architectures ouvertes et désagrégées capables de prendre en charge des services diversifiés tout en s'adaptant à une demande de trafic en forte croissance et hautement variable. Cette évolution est rendue possible grâce au matériel programmable, aux modèles de données normalisés et aux plateformes de contrôle *cloud-native*. Cette thèse propose une architecture à base de microservices combinant des systèmes optiques ouverts et désagrégés avec les pratiques GitOps et *Network as Code*, afin de réaliser l'automatisation, l'intégration continue et la gestion du cycle de vie des fonctions de contrôle des réseaux optiques.

Sur cette base, la recherche examine comment l'Apprentissage Automatique (*Machine Learning*, ML) et l'Apprentissage par Renforcement (*Reinforcement Learning*, RL) peuvent relever les principaux défis de contrôle et de gestion. Les travaux portent sur l'allocation du routage et du spectre, la prédiction du trafic et de l'état du réseau, l'estimation de la qualité de transmission ainsi que la configuration adaptative des transpondeurs. Pour améliorer ces tâches, un cadre de *Compositional Machine Learning* (CML) est introduit, décomposant des opérations complexes en modèles modulaires pouvant être enchaînés en pipelines de contrôle optique. Cette approche améliore l'évolutivité, l'interprétabilité et la réutilisation des fonctions pilotées par ML. Des méthodes de *MLOps* — incluant le réentraînement automatisé, la détection de dérive et la supervision continue — sont intégrées au cadre afin de garantir un fonctionnement robuste et adaptatif des modèles dans des conditions réseau dynamiques. Les simulations et expérimentations démontrent des gains significatifs en termes d'utilisation spectrale, de probabilité de blocage et d'automatisation opérationnelle par rapport aux approches existantes.

En réunissant architectures *cloud-native* et contrôle piloté par ML, cette thèse fait progresser l'intégration de l'automatisation et de l'intelligence dans les réseaux optiques. Elle esquisse également des perspectives telles que la coordination inter-couche IP/Optique, les boucles fermées auto-cicatrisantes et le contrôle de ressources *AI-native*, ouvrant la voie vers des infrastructures optiques autonomes et durables.

**Mots-clés:** Open and Disaggregated Optical Networks; Network Management as Code; Autonomous Optical Networks; Machine Learning; Reinforcement Learning;

ML-driven Optical Network Control; Compositional ML; AI-Native Optical Networks; Self-Driven Optical Networks.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| A2C | Advantage Actor-Critic |
| A3C | Asynchronous A2C |
| AE | Automation Engine |
| AI | Artificial Intelligence |
| AMF | Aggregated Mondrian Forest |
| ANN | Artificial Neural Networks |
| API | Application Programming Interface |
| ARF | Adaptive Random Forest |
| ARIMA | Auto-Regressive Integrated Moving Average |
| AUC | Area Under the Curve |
| AutoML | Automated ML |
| AWS | Amazon Web Services |
| BER | Bit Error Rate |
| BO | Bayesian Optimization |
| BP | Blocking Probability |
| BPSK | Binary Phase Shift Keying |
| BSS | Business Support System |
| C-NMF | Collective non-Negative Matrix Factorization |
| CAPEX | Capital Expenditure |
| CBR | Case-Based Reasoning |
| CD | Continuous Delivery |
| CDC | Colorless, Directionless and Contentionless |
| CDTE | Centralized Training with Decentralized Execution |
| CI/CD | Continuous Integration / Continuous Deployment |
| CLI | Command Line Interface |
| CML | Compositional Machine Learning |
| CMLF | Compositional Machine Learning Framework |

| | |
|---|---|
| CMLFM | CMLF Manager |
| CNN | Convolutional Neural Networks |
| CPM | CML and ML Pipeline Manager |
| DAE | Drift-Adaptive Ensemble |
| DBSCAN | Density-Based Spatial Clustering of Applications with Noise |
| DCI | Data Center Interconnect |
| DDM | Drift Detection Method |
| DNN | Deep Neural Network |
| DQN | Deep Q-Networks |
| DRL | Deep RL |
| DSP | Digital Signal Processing |
| DWDM | Dense WDM |
| EDDM | Early DDM |
| EDFA | Erbium-Doped Fiber Amplifiers |
| EM | Expectation-Maximization |
| eMBB | enhanced Mobile BroadBand |
| EON | Elastic Optical Network |
| FNN | Feedforward Neural Network |
| FS | Frequency Slot |
| GBM | Gradient Boosting Machines |
| GCN | Graph Convolutional Network |
| GCP | Google Cloud Platform |
| GenAI | Generative AI |
| GMM | Gaussian Mixture Models |
| GMPLS | Generalized Multi Protocol Label Switching |
| GN | Gaussian Noise |
| gNMI | gRPC Network Management Interface |
| GNPy | Gaussian Noise Python |
| GOSNR | Generalized OSNR |
| GPR | Gaussian Process Regression |
| gRPC | google Remote Procedure Call |

| | |
|---|---|
| GRU | Gated Recurrent Unit |
| GUI | Graphical User Interface |
| H-SDNC | Hierarchical SDNC |
| HCL | HashiCorp Configuration Language |
| IaC | Infrastructure as Code |
| ILP | Integer Linear Programming |
| IoT | Internet of Things |
| JS | Jensen-Shannon |
| JSON | JavaScript Object Notation |
| KDE | Kernel Density Estimation |
| KL | Kullback-Leibler |
| KNN | K-Nearest Neighbors |
| KS | Kolmogorov-Smirnov |
| LB | Leverage Bagging |
| LCM | Life Cycle Management |
| LF | Linux Foundation |
| LIME | Local Interpretable Model-agnostic Explanations |
| LLM | Large Language Model |
| LP | Lightpath |
| LSTM | Long Short-Term Memory |
| LUF | Link Utilization and Fragmentation |
| MADDPG | (Multi-Agent Deep Deterministic Policy Gradient |
| MAE | Mean Absolute Error |
| MAGC-RSA | Multi-Agent Graph Convolutional Reinforcement Learning for RSA |
| MARL | Multi Agent RL |
| MDP | Markov Decision Process |
| ML | Machine Learning |
| MLOA | MLOps Adapter |
| MLOps | Machine Learning Operations |
| MLSC | Machine Learning Service Chain |
| mMTC | massive Machine Type Communications |

| | |
|---|---|
| MR | Merge Request |
| MSE | Mean Square Error |
| MTTR | Mean Time To Repair |
| NaC | Network as Code |
| NBI | NorthBound Interface |
| NETCONF | Network Configuration Protocol |
| NMaC | Network Management as Code |
| NMC | Network Media Channel |
| NMS | Network Management System |
| OI | Optical Impairment |
| OLS | Open Line System |
| OMS | Optical Multiplex Section |
| ONMI | Open Network Modeling and Interfaces |
| OO-SDN | Open Optical SDN |
| OOPT | Open Optical & Packet Transport |
| OPEX | Operating Expenditure |
| OPM | Optical PM |
| OSNR | Optical Signal-to-Noise Ratio |
| OSS | Operational Suport System |
| OTN | Optical Transport Network |
| OTS | Optical Transmission Section |
| PCA | Principal Component Analysis |
| PCE | Path Computation Element |
| PM | Performance Monitoring |
| PPO | Proximal Policy Optimization |
| PR | Pull Request |
| PSI | Population Stability Index |
| PSS | Photonic Service Switch |
| QoS | Quality of Service |
| QoT | Quality of Transmission |
| RBF | Radial Basis Function |
| RCA | Root-Cause Analysis |

| | |
|---|---|
| RESTCONF | Representational State Transfer Configuration Protocol |
| RL | Reinforcement Learning |
| RLOps | Reinforcement Learning Operations |
| RMSA | Routing, Modulation, and Spectrum Assignment |
| RMSE | Root MSE |
| RNN | Recurrent Neural Network |
| ROADM | Reconfigurable Optical Add-Drop Multiplexer |
| S-BVT | Sliceable Bandwidth-Variable Transceivers |
| SBI | SouthBound Interface |
| SDG | Stochastic Gradient Descent |
| SDN | Software-Defined Networking |
| SDNC | SDN Controller |
| SE | Stitching Engine |
| SF | Spectrum Fragmentation |
| SHAP | (SHapley Additive exPlanations |
| SLA | Service-Level Agreement |
| SNMP | Simple Network Management Protocol |
| SNR | Optical Signal-to-Noise Ratio |
| SOM | Self Organizing Map |
| SP-FF | Shortest Path – First Fit heuristic |
| SRP | Stream Random Patches |
| SVM | Support Vector Machine |
| SVR | Support Vector Regression |
| T-API | Transport API |
| T-PCE | TransportPCE |
| T-SDNC | Transport SDNC |
| T-SNE | t-distributed Stochastic Neighbor Embedding |
| TE | Raffic Engineering |
| TIP | Telecom Infra Project |
| TM Forum | TeleManagement Forum |
| UMAP | Uniform Manifold Approximation and Projection |

| | |
|---|---|
| uRLLC | ultra-Reliable Low-Latency Communication |
| VAE | Variational AutoEncoders |
| VNF | Virtual Network Function |
| VNT | Virtual Network Toopologies |
| WDM | Wavelength Division Multiplexing |
| WFEE | Workflow Execution Engine |
| WSS | Wavelength Selective Switch |
| XAI | Explainable Artificial Intelligence |
| YAML | Yet Another Markup Language |
| YANG | Yet Another Next Generation |

# 1 Introduction

## 1.1 Overview

The future of optical networks is rapidly being defined by the convergence of four major technological shifts: architectural disaggregation, standardization, cloud-native software practices, and the incorporation of Machine Learning (ML) methods into optical network control and management workflows.

Traditionally, conventional optical networks have been designed as vertically integrated systems, with vendor-specific hardware, closed control interfaces, and monolithic management software, which allowed for tight functional coupling and predictable behavior, but which limited the ability to scale, interoperate between vendors, or adapt to rapidly changing service requirements. Recent standardization efforts and open-source initiatives have accelerated a transition toward disaggregated optical network architectures where devices expose abstracted capabilities through standardized data models and open Application Programming Interfaces (APIs), enabling independent evolution of hardware and software and promoting flexible and programmable control.

Simultaneously, operational paradigms are shifting to cloud-native principles, where network control stacks are being designed as distributed microservice architectures, deployed on container orchestration platforms, and managed through version-controlled GitOps workflows. This provides faster iteration, declarative state convergence, and more resilient system behavior through stateless services and automated reconciliation loops, bringing significant benefits in operational agility and reproducibility, and preparing the ground for intent-based and closed-loop network automation.

However, despite these advancements in programmability and modular design, the core challenge that remains is the complexity of decision-making introduced by such systems. The control plane is now required to manage a growing number of operational parameters, interdependent resources, and real-time telemetry sources. Manual intervention, static heuristics, or rule-based logic are no longer sufficient to scale with the dimensionality and dynamics of today's optical networks. This lack of

scalable and adaptive intelligence in the control loop comprises the central problem addressed in this thesis.

This complexity is further increased by the heterogeneity of devices, the variability in service demands, and the need to meet increasingly stringent performance and energy-efficiency requirements. Without mechanisms to interpret telemetry, anticipate changes, or make context-aware decisions, the operational benefits of open and disaggregated networks cannot be fully realized. As networks become more programmable, the need for automation becomes essential.

In this context, Machine-Learning (ML) offers a promising alternative to conventional control methods. By learning from data rather than relying on pre-defined rules, ML models can capture hidden patterns, adapt to non-stationary behavior, and optimize for multiple (and possibly) conflicting objectives. Forecasting future traffic demands, estimating signal quality under varying impairments, and selecting appropriate modulation formats are examples of tasks that benefit from data-driven inference and continuous learning.

This thesis proposes a Compositional Machine-Learning (CML) framework for open and disaggregated optical network control. The solution integrates modular learning components including time-series forecasting models, graph-based representations, and Reinforcement Learning (RL) agents—into a cloud-native control architecture driven by GitOps workflows. In addition, it addresses the life-cycle management of such components through Machine-Learning Operations (MLOps) practices, ensuring reproducibility, monitoring, and safe integration into production environments.

Unlike previous approaches, which often focus on offline model development or isolated inference tasks, the framework developed in this work is designed for end-to-end deployment in realistic network scenarios. Its compositional structure enables reusability across network functions, topologies, and service objectives. Its cloud-native orchestration ensures compatibility with existing automation pipelines. Additionally, its data-driven decision mechanisms offer better adaptability and performance than static control solutions.

By bridging the gap between disaggregated infrastructure, cloud-native software engineering, and Machine-Learning-based intelligence, this work aims to contribute to the development of intelligent, autonomous, and sustainable optical network control systems.

## 1.2 Motivation

The motivation for this research comes from the perception that current approaches to optical network control and automation are insufficient to meet the operational demands of future communication systems. As transport infrastructures evolve to support services with stringent requirements on latency, bandwidth elasticity, and availability, the ability to make prompt and informed control decisions becomes increasingly critical. However, the current optical network management practices remain largely reactive, configuration-driven, and dependent on human operators.

One of the fundamental barriers to progress is the lack of scalable intelligence embedded in the control plane. Even in scenarios where programmability and telemetry are available, decision logic is often static, heuristic-based, or confined to a narrow operational context. This results in inefficient resource utilization, slow response to failures or demand fluctuations, and limited adaptability to unforeseen conditions. The inability to exploit historical optical network behavior/states/data or to reason under uncertainty restricts the network's capacity to operate autonomously or to optimize for multiple service objectives simultaneously.

At the same time, there is a growing awareness in the research community that Machine Learning could offer a more flexible and data-driven approach to address such challenges. Yet, the practical application of Machine Learning in real network environments remains limited. There is a breach between academic research—often focused on offline model performance—and the operational facts of deploying Machine Learning models in production/operational network control loops. Issues such as integration with orchestrators, model generalization across domains, update safety, and observability of Machine Learning decisions are often overlooked.

This thesis is motivated by the need to bridge that gap. It is driven by the conviction that Machine Learning must not be treated as an external add-on but as an integral component of the network control architecture—subject to the same principles of modularity, automation, and reproducibility that govern the rest of the system. Moreover, there is a need to move beyond isolated use cases and toward reusable, composable, and lifecycle-aware learning components that can be orchestrated alongside traditional control logic.

Ultimately, this research is motivated by the ambition to enable a new generation of optical networks—ones that are not only open and programmable, but also capable of learning from their own behavior, adapting to changing conditions, and making correct decisions with minimal human intervention. It seeks to contribute to the

architectural, methodological, and operational foundations required to realize that vision.

## 1.3 Research Objectives and Questions

The main objective of this thesis has been to develop a modular, intelligent, and cloud-native framework for the control and automation of open and disaggregated optical networks. This is achieved through the systematic integration of Machine Learning and Reinforcement Learning techniques into software-defined optical network control planes, applicable to any optical network systems. In particular, this requires ensuring compatibility across multi-vendor environments, which introduces challenges related to standardization, abstraction of device capabilities, and the implementation of interoperable interfaces. The research spans architectural design, control methodology, data-driven learning mechanisms, and the orchestration of Machine-Learning models in production-like environments.

To achieve this goal, the thesis addresses several key challenges across multiple dimensions of optical network evolution and intelligence. These include: enabling openness and disaggregation through standardized control interfaces and declarative workflows; building Machine-Learning-native components that learn from real or emulated telemetry; supporting the lifecycle of Machine-Learning models within cloud-native microservice architectures; and composing multiple learning agents into cohesive, end-to-end pipelines for multi-objective network optimization.

The following research questions were guided by these requirements:

**RQ1:** *What are the architectural principles required to support modular, programmable, and flexible control in open and disaggregated optical networks, ensuring interoperability across any optical systems?* This question explores how openness and disaggregation change optical network control, and what are the design choices needed to support interoperability, scalability, and automation readiness.

**RQ2:** *How can network automation workflows be designed to enable declarative, intent-based control and integrate seamlessly with continuous deployment and telemetry feedback mechanisms?* This question investigates how cloud-native software practices such as GitOps, microservice orchestration, and declarative APIs can be adapted to the context of optical network operations.

**RQ3:** *What are the types of Machine Learning and Reinforcement Learning methods suitable for enhancing decision-making in optical network operations, and how can they address challenges such as prediction, resource optimization, and failure response?* This question examines the applicability of learning-based techniques to fundamental control problems in optical networks and investigates the requirements for their effective training, inference, and generalization.

**RQ4:** *How can learning tasks be structured and composed in a modular way, to enable scalable integration of multiple Machine Learning models and facilitate their reuse across network functions?* This question addresses the design of composable learning pipelines that decompose large optical network control functions into atomic, manageable and reusable software components.

**RQ5:** *What operational mechanisms are needed to manage the lifecycle of Machine Learning models in optical network environments, including deployment, monitoring, retraining, and rollback?* This question explores the integration of Machine Learning Operations (MLOps) practices to ensure reliability, traceability, and maintainability of Machine Learning-based optical network controls.

## 1.4 Scientific and Technical Contributions

### Publications in Peer-Reviewed Conferences

The following scientific publications were produced during the course of this research and presented at top-tier international conferences in optical networking and automation:

- Huy Quang Tran, Omar Houidi, **Javier Errea**, Dominique Verchère, and Djamal Zeghlache. "MAGC-RSA: Multi-agent graph convolutional reinforcement learning for distributed routing and spectrum assignment in elastic optical networks." In *Proceedings of the European Conference on Optical Communication (ECOC)*, 2022. This work proposes a distributed multi-agent graph convolutional reinforcement learning (MAGC-RSA) approach for solving the Routing and Spectrum Assignment (RSA) problem in elastic optical networks. It introduces an architecture based on localized decision agents and shared graph embeddings to improve scalability and spectrum utilization. The technical details and system design are discussed in **Part II**, in the chapters related to ML applications for resource allocation.

- **Javier Errea**, Huy Quang Tran, Dominique Verchère, Huu Trung Thieu, Andrea Mazzini, Lahcen Abnaou, Jelena Pesic, Marina Curtol, Abdelali El Imadi, Adlen Ksentini, et al. "Open disaggregated optical network control with network management as code." In *Proceedings of the Optical Fiber Communication Conference (OFC)*, 2023. This paper presents a complete control plane architecture for open and disaggregated optical networks based on the principles of Network Management as Code (NaC). It demonstrates the integration of GitOps workflows with T-API-compliant controllers and outlines the automation of lifecycle operations. The architecture and implementation are detailed in **Part I**.

- **Javier Errea**, Deborah Djon, Huy Quang Tran, Dominique Verchère, and Adlen Ksentini. "Deep reinforcement learning-aided fragmentation-aware RMSA path computation engine for open disaggregated transport networks." In *Proceedings of the International Conference on Optical Network Design and Modeling (ONDM)*, 2023. This contribution focuses on a fragmentation-aware path computation engine for RMSA using deep reinforcement learning in disaggregated transport networks. It introduces a novel learning reward function that reduces fragmentation over time, improving long-term spectrum availability. The methods and evaluation are presented in **Part II**, and further integrated into the compositional framework discussed in **Part III**.

- Huy Quang Tran, **Javier Errea**, Van-Quan Pham, Dominique Verchère, Adlen Ksentini, and Djamal Zeghlache. "Dynamic drift-adaptive ensemble-based quality of transmission classification framework in OTN." In *Proceedings of the International Conference on Transparent Optical Networks (ICTON)*, 2023. This work proposes a dynamic, ensemble-based Quality of Transmission (QoT) classification framework that adapts to model drift using online performance monitoring. It addresses the instability of QoT estimation models under changing conditions, and its contribution is discussed in **Part II**.

- **Javier Errea**, Huy Quang Tran, Huu Trung Thieu, Van Quan Pham, Nakjung Choi, Dominique Verchère, Adlen Ksentini, and Djamal Zeghlache. "Demonstration of a compositional learning framework for open and disaggregated optical network control." In *Proceedings of the Optical Fiber Communication Conference and Exhibition (OFC)*, 2024. This demonstration paper showcases the practical deployment of a compositional ML control framework, combining forecasting and reinforcement learning in a production-like orchestration setup. It validates the feasibility of modular ML pipelines and their orchestration via

GitOps workflows. The demonstration and related architectural principles are presented in **Part III**.

### Open-Source and Standardization Contributions

- Contributor to the **TransportPCE (T-PCE)** project within the OpenDaylight (ODL) platform. Public code contributions: `https://git.opendaylight.org/gerrit/q/project:transportpce+owner:errea%2540eurecom.fr`
- Active participant in the evolution of **Transport-API (T-API) standard** under Linux Foundation (LF) Open Network Modeling and Interfaces (ONMI) project

### Talks and Technical Community Engagement

- **LF Developer and Testing Forum**, January 2022:
  - "ODL: Planned extension of T-API module and integration of OpenConfig device models in T-PCE" - `https://lf-networking.atlassian.net/wiki/spaces/LN/pages/15679938`
  - "ODL: T-API contribution to T-PCE" - `https://lf-networking.atlassian.net/wiki/spaces/LN/pages/15679936`

### Intellectual Property

- Co-inventor of the patent application titled: *"Energy Saving in Dynamic Networks by Extension of TE-Bundles"*, filed under the European procedure and currently pending approval.

## 1.5 Organization

The thesis is structured to progressively explain the architectural, methodological, and operational impacts by integrating cloud-native practices and Machine Learning (ML) to automate open and disaggregated optical networks. The plan follows a bottom-up approach—starting with the evolution of network architectures and

ending with the design, orchestration, and experimental evaluation of intelligent ML-driven control frameworks.

- **Part I – Towards Programmable and Automated Optical Networks: From Legacy Systems to Openness, Disaggregation and Optical Network Management as Code.** This part introduces the historical and technological transformation from vertically integrated optical systems to open and disaggregated infrastructures. It examines the role of Software Defined Networking (SDN), standardization efforts, and multi-vendor interoperability in enabling programmable and modular transport networks. It also presents the application of cloud-native engineering principles to the orchestration of disaggregated optical networks. It outlines the key enabling technologies—such as containerized microservices, declarative interfaces, and GitOps workflows—and motivates their relevance to optical network control. Subsequently, it presents the implementation of a Network Management as Code (NaC) framework, showcasing practical results and integrations with open and programmable optical platforms. The part concludes with a summary of the contributions and findings.

- **Part II – Machine Learning Foundations and Applications in Optical Networks.** This part provides a general overview of Machine Learning (ML) and Reinforcement Learning (RL) techniques. It surveys existing research on ML-based network control, categorizing methods by task and learning model, and highlighting the operational and deployment challenges that persist. It also focuses on the proof-of-concepts of Machine Learning applications into resource control and service management functions of disaggregated optical environments. It introduces the architectural context and design considerations for Machine Learning integration. Then, it present the development, training, and evaluation of models for different operational functions of optical networks. An analysis of the results, their limitations, and the lessons learned concludes this part.

- **Part III – Compositional Machine Learning Framework for Open and Disaggregated Optical Network Control.** This part presents a modular Compositional Machine Learning framework that focuses on the composition and chaining of ML-based control components to automate optical network operations and thereby reduce human intervention. It first introduces the design principles and system architectures that enable interoperability and reuse of different models. Subsequent chapters detail the integration of time-series forecasting, reinforcement learning agents, and orchestration mechanisms

into composite workflows capable of predictive and adaptive decision-making. Experimental results validate the benefits of this approach. This part concludes with a discussion of its implications for future ML-native optical control systems.

The thesis concludes with a synthesis of the main contributions and the future research directions that can be envisioned.

# Part I

## Towards Programmable and Automated Optical Networks: From Legacy Systems to Openness, Disaggregation and Optical Network Management as Code

In this part, we present a comprehensive framework for automated optical network control based on the principles of Network Management as Code (NMaC). This paradigm introduces a declarative, software-defined approach to orchestrating both the underlying transport infrastructure and the lifecycle of SDN-based control applications. Leveraging GitOps as the core operational model, we align infrastructure and application state with version-controlled repositories, enabling deterministic, reproducible, and auditable network operations. The chapter focuses specifically on open and disaggregated optical networks, where heterogeneous hardware and evolving standards demand a flexible yet robust control plane architecture.

To this end, we introduce an Automation Engine (AE) that serves as the intelligence layer for change detection and response. The AE continuously monitors the network topology definition, and upon detecting updates—such as the addition of new nodes, changes in connectivity, or modifications in vendor-specific configurations—it triggers a Continuous Delivery (CD) workflow through a GitOps operator, in our case ArgoCD. This results in the automated deployment, upgrade, or removal of containerized SDN control functions on a Kubernetes-based microservice infrastructure.

The proposed architecture supports adaptive deployment scenarios, including the seamless transition from partially to fully disaggregated optical topologies, and integrates support for multi-vendor environments. The Automation Engine also coordinates with enhanced T-API interfaces to reflect the current topology and service context, enabling real-time synchronization between control plane components and network state.

Importantly, this work constitutes the first implementation of such a cloud-native control framework in the optical networking domain, combining GitOps, NMaC, microservices, and open standards such as T-API and OpenConfig in a unified, fully automated orchestration pipeline. Through detailed design, deployment, and demonstration, this chapter validates the ability of the NMaC framework to ensure continuous alignment between the data plane and the deployed control logic, while enabling scalable and programmable optical transport automation.

# Introduction

2

Conventional optical networks emerged during an era mainly focused on delivering reliable, high-capacity, long-distance transmission. These systems were built on the foundation of *Wavelength Division Multiplexing (WDM)*, which enabled multiple optical channels to be multiplexed over a single fiber, increasing throughput significantly. The evolution into *Dense WDM (DWDM)* architectures further amplified this capacity, pushing transmission speeds from 10 Gbps to 100 Gbps and beyond [1]. These advances supported the exponential growth of global internet traffic and cloud-based services.

Early optical transport networks emphasized transparency, with minimal electronic regeneration, and focused on deterministic point-to-point connections. While this simplified certain aspects of transport, the architecture relied heavily on proprietary hardware and management systems. Devices such as Reconfigurable Add Drop Multiplexers (ROADMs), transponders, and optical amplifiers were supplied and controlled through vertically integrated vendor ecosystems. The design was not inherently modular or programmable, making it difficult to adapt as service demands grew more diverse and dynamic [2].

Furthermore, the management and control mechanisms of these conventional systems were largely static. Legacy protocols such as Generalized Multi Protocol Label Switching (GMPLS), Simple Network Management Protocol (SNMP), and Command Line Interfaces (CLI) formed the basis of operational workflows, relying on human-in-the-loop decision-making for tasks like fault localization, topology updates, and service provisioning. This centralized, manual approach contributed to long provisioning cycles and constrained real-time responsiveness.

As the demand and scale of services increased—with requirements ranging from mobile backhaul and enterprise connectivity to Data Center Interconnect (DCI) and cloud-native workloads—traditional architectures began to show their limitations. They were not suited for elastic provisioning, telemetry-driven optimization, or integration into multi-domain orchestration systems.

Despite these constraints, conventional optical networks established the physical and operational foundations upon which modern architectures are being constructed.

Their evolution is now driven by the need for *agility, openness, and automation*, leading to a shift toward software-defined paradigms and disaggregated hardware platforms.

The following sections dig deeper into the specific limitations of monolithic optical networks, setting the stage for a comprehensive exploration of the architectural and operational transformation shaping the next generation of optical transport systems.

## 2.1 Limitations of Monolithic Optical Network Architectures

Traditional monolithic optical network architectures have long served as the foundation for global transport infrastructures. These systems are typically composed of tightly coupled, vertically integrated hardware and software components sourced from a single manufacturer. While this approach initially enabled high levels of performance optimization and stability, it introduced a multitude of limitations that are no longer viable in the context of modern networking demands, which emphasize agility, automation, and cost-efficiency.

One of the most critical drawbacks of monolithic optical networks is the phenomenon of constrained interoperability. In these systems, operators are bound to a single provider's line systems, transponders, amplifiers, and control plane software, which communicate via proprietary interfaces [3]. As a result, interoperability across different systems becomes virtually impossible without complex integration layers or costly architectural transformations. This situation, often described as *vendor lock-in*, not only inflates Capital Expenditure (CAPEX) and Operating Expenditure (OPEX) through limited market competition but also hinders innovation, since new features and capabilities must align with a proprietary development roadmap [4, 5]. Moreover, achieving end-to-end service automation across heterogeneous systems remains a significant challenge, as data models and control abstractions are rarely aligned. Even when standardization efforts define common YANG models or open APIs, slight variations in interpretation and implementation across systems can create semantic mismatches. Ensuring that two different systems interoperate exactly on the same model—without translation or adaptation—remains difficult in practice, further complicating integration.

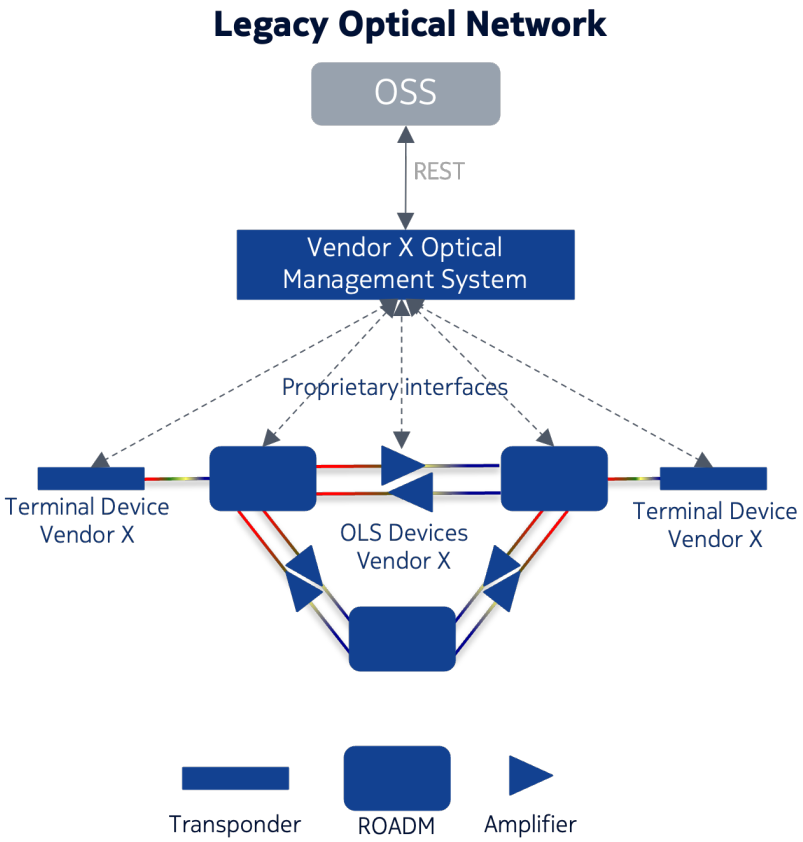



**Fig. 2.1.:** Illustration of a monolithic optical network architecture showing vertically integrated components and proprietary interfaces.

As shown in Figure 2.1, the tightly coupled nature of monolithic optical architectures produces fragmented control and management domains with limited adaptability that complicate integration, scalability, and automation. These systems generally rely on static provisioning models, where lightpaths and bandwidth resources are manually configured by network operators during the planning or commissioning phase [6]. These configurations are executed through proprietary Network Management Systems (NMS), each with its own operational logic. End-to-end provisioning often requires the operator to configure each device in the optical path separately, performing sanity checks, verifying link budgets, and ensuring protocol-specific compatibility.

The complete service lifecycle—from network planning to billing—entails numerous disjointed steps: conducting site surveys and fiber qualification; drafting transmission design based on link budgets, modulation formats, and amplifier placements; ordering and physically installing equipment; configuring devices through NMS or CLI; activating services via dedicated workflows; and finally, integrating the service

into Operational Support Systems/Business Support Systems (OSS/BSS) layers for customer management and billing. Each step is resource-intensive, heavily manual, and introduces significant latency into service delivery pipelines.

Moreover, conventional optical network design is not equipped to respond to evolving customer demands in real-time. Modifying services—be it due to bandwidth changes, protection switching, or rerouting—often entails repeating large parts of the provisioning and validation process. This lack of agility is a direct consequence of monolithic architectures' limited telemetry, closed APIs, and absence of centralized abstraction.

Another major limitation of traditional architectures is the lack of telemetry and real-time programmability. Legacy control planes, such as those based on GMPLS or proprietary management protocols, offer limited visibility into the physical layer, with minimal support for streaming telemetry or closed-loop control [7]. Operators are often forced to rely on polling mechanisms, CLI-based monitoring, or static thresholds, which provide delayed or incomplete information about network performance. The absence of real-time feedback inhibits proactive fault management and performance optimization. For instance, degradation in optical signal quality may go unnoticed until it breaches the Bit Error Rate (BER) threshold, resulting in service outages and Service Level Agreement (SLA) violations. Similarly, network planning tools must rely on worst-case design margins due to the lack of real-time Quality-of-Transmission (QoT) data, leading to underutilization of network resources [8, 9].

Monolithic systems typically employ closed software architectures that prevent the integration of third-party applications, orchestration systems, or SDN controllers. The network control and management stack is a black box, hindering external programmability and making it difficult to implement advanced functionalities [10]. Even in systems where some form of management interface is exposed, such as SNMP, the data models are inconsistent and lack the semantic richness required for automation. In contrast, modern frameworks based on standardized YANG models and NETCONF/RESTCONF protocols support model-driven management, enabling a high degree of abstraction and automation [11, 12].

The tightly coupled nature of monolithic systems makes them inherently difficult to scale or evolve. Adding new transponders, Reconfigurable Optical Add Drop Multiplexer (ROADMs), or amplifiers often requires compatibility checks, firmware updates, and sometimes even hardware replacements to ensure interworking. Such upgrades are labor-intensive, service-disruptive, and often require direct manufacturer support [13]. Furthermore, the lack of disaggregation prevents the modular

scaling of subsystems. For instance, increasing transmission capacity in a specific part of the network may necessitate changes across the entire system stack due to compatibility constraints, making the network less agile in responding to unplanned capacity demands.

These limitations position monolithic networks firmly within the lower levels of the Tele Management (TM) Forum's Autonomous Networks framework—specifically Levels 0 and 1—which are characterized by fully manual or tool-assisted operations with limited automation, no closed-loop control, and siloed domain management [14]. Progressing beyond these levels requires breaking away from proprietary constraints and embracing openness, standardization, and programmability.

In summary, monolithic optical network architectures are constrained by proprietary control stacks, static provisioning, lack of real-time telemetry, and scalability bottlenecks. While terms such as *vendor-specific*, *vendor-neutral*, or *multi-vendor* are often used to describe these challenges, they are not technical problems per se but rather business strategies by operators seeking to mitigate dependency on a single manufacturer. From a technical perspective, the real challenge is ensuring that network resource control and service management operate seamlessly across *any type of optical system*, independent of origin. This is made possible by the abstraction capabilities provided by standardized data models and open frameworks. By relying on these abstractions, optical networks can achieve interoperability and automation across heterogeneous technologies, overcoming the structural limitations of monolithic systems.

## 2.2 Key Drivers of Optical Network Transformation

The transformation of optical networks is being driven by a congregation of technological, operational, and economic demands: exponential traffic growth from 5G and cloud-native applications, dynamic service expectations, and the need for agility, cost efficiency, and automation in network operations. Modern optical networks are expected not only to scale in capacity but also to meet stringent latency guarantees, enable dynamic programmability, and support closed-loop control. This shift—from static, hardware-centric systems to dynamic, software-defined infrastructures—marks a fundamental evolution in network design and management [15, 16].

A primary motivator for this transition is the explosive increase in bandwidth demand. According to Cisco's Global Cloud Index, global IP traffic exceeded 4.8

zettabytes in 2022, driven by streaming video, hyperscale cloud interconnectivity, and emerging Artifitial Inteligence (AI) workloads [17]. Services such as enhanced Mobile Broadband (eMBB), ultra-Reliable Low-Latency Communication (uRLLC), and massive Machine-Type Communications (mMTC) associated with 5G have introduced new classes of service with diverse and unpredictable transport profiles [18]. The legacy paradigm of over-provisioning static optical connections is inadequate in the face of these dynamic demands. Network infrastructures must support elasticity, real-time reconfiguration, and telemetry-driven adaptation.

Simultaneously, advances in optical hardware have made the physical layer increasingly programmable. Coherent digital signal processing (DSP), flexible-grid ROADMs, and Sliceable Bandwidth-Variable Transceivers (S-BVTs) now enable per-channel tuning of key transmission parameters such as symbol rate, modulation format, and optical spectrum allocation [19, 20]. These capabilities strengthen the concept of Elastic Optical Networks (EONs), in which spectral resources are dynamically allocated in response to service-level requirements. Additionally, the introduction of Open Line Systems (OLS)—in which terminal equipment (e.g., transponders) and the line system (e.g., ROADMs and amplifiers) are provided from different vendors—has enabled multi-vendor deployments, reduced vendor lock-in, and created a path toward modular, software-defined optical networks [21].

This technological shift aligns closely with the architectural principle of disaggregation. In the legacy monolithic architecture, optical transport networks are built using tightly integrated systems, where optical devices and their associated control and management software are provided by a single manufacturer. These elements are designed to interoperate within a proprietary ecosystem, managed through a vendor-specific NMS that limits external access and programmability. While such vertical integration can simplify provisioning and guarantee performance, it leads to high operational costs, vendor lock-in, and limited flexibility in adopting emerging technologies or mixing equipment from different suppliers.

The partially disaggregated architecture emerged as a transitional phase to address these limitations. In this model, operators retain the vendor-specific OLS but allow the integration of transponders and terminal devices from third-party vendors. This is achieved through open optical interfaces and standardized signal specifications, enabling plug-and-play interoperability at the photonic layer [21, 22]. However, while the transponders may be configured through their own vendor-specific or standardized APIs, the OLS remains controlled by a proprietary domain controller. As a result, the management remains fragmented: the OLS and transponders are independently managed, often requiring manual coordination or custom integration

logic between systems. Although this architecture improves flexibility and enables technological innovation at the network edge, it does not yet deliver full software-defined control or unified telemetry across the end-to-end optical path.

Fully disaggregated architectures represent the next stage in this evolution by decoupling not only the hardware components but also the control and management layers. In this model, all network elements—transponders, ROADMs, amplifiers, and other optical devices—expose standardized, model-driven interfaces such as NETCONF [11], RESTCONF [12], or gRPC [23] using YANG data models. These southbound interfaces enable a logically centralized SDN controller to replace legacy vendor-specific NMS, providing unified orchestration of the entire optical domain. The controller maintains a real-time inventory and topological view of the disaggregated network, and supports service provisioning, path computation, telemetry collection, and fault management. It can also interface with upper-layer orchestrators or OSS/BSS platforms via northbound APIs, ensuring integration across the broader service management stack.

From an operational perspective, full disaggregation offers several advantages: unified control, support for heterogeneous optical systems, faster innovation cycles, and easier integration with automation frameworks and external applications. It also allows finer-grained optimization, since devices can be individually tuned or upgraded without disrupting the entire system. At the same time, this paradigm introduces new challenges, including increased effort in system integration, the need for rigorous interoperability testing, and reliance on robust SDN controllers capable of operating in multi-technology environments.

Figure 2.2 illustrates this architectural trajectory—from monolithic, vertically integrated systems to fully disaggregated environments where modularity, openness, and interoperability are foundational design principles [22].

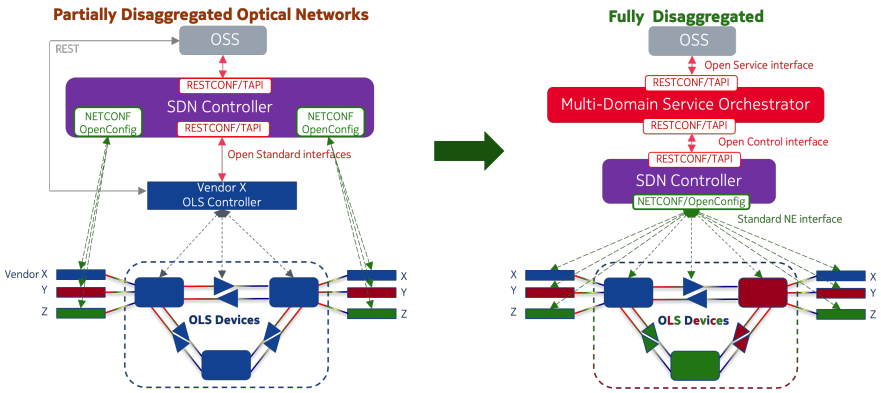



**Fig. 2.2.:** Evolution from partially to fully disaggregated optical transport architectures

To enable this vision, standardized YANG models play a central role. They provide the common information structures that controllers and orchestrators rely on to interact with heterogeneous devices and domains. These models serve different levels of abstraction depending on their purpose and scope.

OpenConfig offers vendor-agnostic control over individual device behaviors—such as wavelength assignment, optical power, and modulation formats—making it particularly suitable for programmable transponders and OLS components in white-box scenarios [24]. It supports both configuration and streaming telemetry through gRPC Network Management Interface (gNMI), enabling closed-loop monitoring and rapid fault detection.

In contrast, OpenROADM expands its scope to model topologies, operational constraints, and service layers. Its unified schema enables coordinated provisioning across ROADMs, amplifiers, and transponders while ensuring consistency across vendors [5]. OpenROADM also promotes component-level disaggregation, modeling each optical function—e.g., amplifiers, degrees, add/drop ports—as an independent network element. While this fine granularity supports detailed service provisioning and failure isolation, it significantly increases interoperability complexity across vendors and network domains. Furthermore, OpenROADM lacks native support for streaming telemetry, relying solely on polling-based retrieval of operational data.

At a higher level, the Linux Foundation Transport-API (T-API) introduces a service-oriented abstraction that is topology- and vendor-agnostic [25]. Rather than modeling individual device parameters, T-API focuses on connectivity services, logical termination points, and virtual topologies. It is designed for multi-domain service orchestration, where controllers within each domain—whether OpenROADM- or OpenConfig-based—present a harmonized view of capabilities to a centralized orchestrator. TAPI enables intent-driven operations, dynamic path computation, and end-to-end lifecycle management without requiring exposure to underlying photonic constraints [26]. The combination of OpenConfig for device-level modeling and gNMI telemetry with T-API for abstracted service orchestration effectively fulfills the functional scope that OpenROADM aims to address. This architectural separation allows high-frequency observability at the device layer and scalable, intent-driven coordination at the service layer, while minimizing integration overhead and maintaining alignment with modern cloud-native design principles.

The progressive disaggregation of optical networks provides the necessary foundation for advancing through the TM Forum's autonomous network levels. At the lower end of the spectrum, Level 0 (manual operations) reflects the traditional monolithic

model, where all service configuration, fault handling, and performance optimization are carried out manually through vendor-specific management systems. The introduction of partially disaggregated architectures, where third-party transponders operate alongside proprietary optical line systems, corresponds to Level 1 (assisted automation). Here, basic automation scripts or controller-based interfaces (e.g., CLI wrappers, domain-specific NMS) assist human operators in provisioning tasks, but end-to-end workflows still rely on manual validation and coordination.

As networks adopt open interfaces and model-driven SDN control—enabled by YANG models—the control plane becomes programmable, and telemetry data can be collected in near real-time. This capability enables the transition to Level 2 (partial automation), where intent-based provisioning, automated path computation, and proactive monitoring can be executed within a domain under SDN control. However, these systems typically lack automated correlation across domains, unified service assurance, or closed-loop feedback mechanisms.

To move to Level 3 (conditional automation), disaggregated optical systems must be integrated into end-to-end orchestration frameworks that can translate service intents into executable configurations, validate constraints, and automatically reconfigure services in response to faults or performance degradation. This demands not only a robust SDN infrastructure but also the use of telemetry-driven policies, analytics engines, and programmable assurance functions.

Level 4 (high autonomy) introduces closed-loop control mechanisms across optical domains, enabling dynamic adaptation of transponders, ROADMs, and other optical network elements based on real-time traffic or impairment data. At this stage, most control decisions are automated, but human operators still intervene for policy updates or exception handling. Achieving Level 5 (full autonomy) in optical networks implies an end-to-end architecture capable of intent-driven orchestration, self-healing behaviors, and AI-guided optimization—without manual intervention. Optical resources are continuously monitored and adjusted using machine learning models, making the transport layer responsive, resilient, and fully integrated into autonomous service delivery. A domain-specific interpretation of the TM Forum autonomous levels for optical networks is presented in Table 2.1.

In conclusion, the ongoing transformation of optical transport networks is underpinned by the confluence of exponential traffic growth, heterogeneous service-level requirements, CAPEX/OPEX pressures, and the maturing of programmable optical components. Architectures based on disaggregation and software-defined networking provide the necessary abstraction layers to decouple control from hardware and enable heterogeneous multi-vendor deployments. Meanwhile, standardized YANG

models and interoperable APIs facilitate deterministic configuration, performance monitoring, and service orchestration. Despite these foundational advances, most deployments remain limited to domain-specific automation and lack cohesive lifecycle integration. Achieving higher levels of autonomy—as defined by frameworks such as the TM Forum Autonomous Networks model—requires not only technological convergence but also alignment of operational workflows, data modeling ecosystems, and deployment pipelines. This research aims to address these systemic gaps by demonstrating how open, disaggregated, and programmable optical networks can be operationalized through declarative automation frameworks and cloud-native orchestration techniques that progressively unlock the next levels of autonomy.

The evolution of optical networks toward openness, disaggregation, and programmability represents a foundational shift in both architecture and operations. As this

**Tab. 2.1.:** Autonomous Network Levels Applied to Optical Transport (based on TM Forum autonomous network Framework)

| Level | Interpretation in Optical Networks |
|---|---|
| **0 – Manual** | Legacy monolithic systems with manual provisioning. CLI or NMS used for configuring optical elements. No telemetry or automation capabilities. |
| **1 – Assisted** | Partial disaggregation allows some device diversity. Operators use basic SDN tools or scripts for configuration. Most workflows are human-executed. |
| **2 – Partial Automation** | SDN controller manages disaggregated optical components using NETCONF/RESTCONF/gRPC. Provisioning and telemetry are automated within domains. |
| **3 – Conditional Automation** | Cross-domain orchestration supports intent-based provisioning and automated fault response. Pre-defined conditions trigger reconfiguration. Telemetry and analytics inform decisions. |
| **4 – High Autonomy** | Closed-loop control for optical topology, modulation formats, and power levels. Multi-layer coordination. Human role limited to policy setting and validation. |
| **5 – Full Autonomy** | Optical infrastructure is fully autonomous. Artifitial Inteligence (AI)/ML systems handle service assurance, self-optimization, and end-to-end orchestration. No human involvement in control loops. |

chapter has shown, the combination of traffic-driven pressures, programmable optics, and standardized SDN control is reshaping the transport layer into a more agile, vendor-neutral, and automation-ready infrastructure. Yet, despite these advances, most current deployments remain only partially automated, falling short of the higher levels of autonomy. Bridging this gap requires not only technical interoperability but also operational transformation—redefining how control, configuration, and service logic are developed, deployed, and managed. The next section addresses this critical challenge by introducing a cloud-native operational paradigm that applies DevOps principles, declarative automation, and GitOps pipelines to the control and orchestration of open and disaggregated optical transport networks.

## 2.3 Towards the Next Era of Optical Networking

The evolution of communication networks has been driven by increasing demands for high-throughput, ultra-reliable connectivity, secure infrastructure, and—more recently—energy-efficient operation [27, 28]. As digital societies evolve, optical networks have emerged as the technological foundation enabling massive data transport across metropolitan, regional, and long-haul segments. They offer unparalleled capacity, low latency, and high resilience, and today constitute the backbone of global communication systems [29, 30]. However, the escalating complexity and heterogeneity of optical network infrastructures—spanning diverse vendors, technologies, and protocol layers—has outpaced the capabilities of traditional management paradigms. These future networks will no longer be composed of static point-to-point connections, but are instead dynamic, multi-domain, and multi-layered ecosystems where transponders, ROADMs, amplifiers, and controllers must be orchestrated cohesively in real time [31, 32].

This growing complexity stems from multiple converging factors. First, the trend toward network disaggregation has led to a decoupling of hardware and software components, resulting in highly modular but inherently heterogeneous infrastructures [33]. This modularity, while promoting flexibility and vendor independence, introduces significant integration overhead, as each component may come with its own configuration mechanisms, life-cycle management processes, and operational semantics. Second, optical transport networks are no longer static infrastructures; they are expected to support dynamic service provisioning, real-time reconfiguration, and high resiliency in the face of failures or traffic surges [34]. This operational dynamism requires a control plane capable of managing rapid changes across layers and domains without human intervention. Third, as optical networks scale

across geographic and administrative boundaries, the number of interconnected elements—such as transponders, ROADMs, amplifiers, and monitoring tools—can grow exponentially [35]. Coordinating the behavior of such a large and distributed system in a consistent and reliable manner is inherently challenging. Moreover, the lifecycle of these systems spans multiple years, during which software, hardware, and protocols evolve asynchronously. Ensuring interoperability, version compatibility, and continuous service availability under these conditions represents a non-trivial engineering problem. Thus, the control and management plane must not only scale with the physical network but also adapt flexibly to new capabilities, deployment environments, and evolving operational requirements.

To address these challenges, the industry has increasingly embraced cloud-native principles in the design of control and orchestration frameworks [36]. Microservice-based architectures enable fine-grained functional decomposition, independent scaling, and fault isolation. Moreover, the incorporation of advanced software engineering practices—such as DevOps, Network-as-Code (NaC), and GitOps—has introduced a new paradigm for managing optical networks. These methodologies leverage infrastructure-as-code principles to version-control the network state and enable continuous deployment and reconciliation of control functions based on Git repositories. ArgoCD and similar tools constantly monitor the health and configuration drift of microservices deployed on Kubernetes, ensuring that the running state of the network remains aligned with its declarative definition [37]. This DevOps-driven paradigm not only reduces operational overhead but also allows operators to swiftly integrate new features, adhere to evolving open standards, and automate lifecycle management of the control plane [38].

This chapter is dedicated to unpacking the technical foundations and integration strategies of these enabling technologies. We analyze their architectural patterns, operational workflows, and implementation trade-offs, providing a cohesive view of how infrastructure-as-code and declarative network operations can transform the control of large-scale, heterogeneous systems. By decoupling intent from realization and enabling continuous alignment between them, these paradigms redefine the reliability, adaptability, and evolvability of networked infrastructures.

In later sections, we transition from the abstract capabilities of these technologies to their concrete realization in the domain of transport networks. In particular, we explore how DevOps workflows, NaC models, and GitOps operations can be applied to the orchestration of disaggregated, multi-vendor systems, then demonstrating their effectiveness in managing the complexity of modern network control.

### 2.3.1 DevOps: Bridging Development and Operations

DevOps represents a transformative paradigm that unifies software development and operational practices, with the objective of accelerating the delivery lifecycle while maintaining system reliability and service quality [39, 40]. Originally rooted in the cloud-native and enterprise IT domains, DevOps has evolved to emphasize automation, Continuous Integration and Continuous Deployment (CI/CD), version control, Infrastructure as Code (IaC), automated testing, and feedback-driven improvement. These practices have become instrumental in enabling continuous delivery of features, reducing human error, and ensuring resilience across complex system architectures [41, 42].

In large-scale distributed systems, such as those supporting financial services, e-commerce platforms, and industrial Internet of Things (IoT) applications, the adoption of DevOps has been central to achieving operational agility. These domains often involve mission-critical applications where system downtime or service disruptions incur significant costs. The iterative nature of DevOps, coupled with its emphasis on infrastructure automation and observability, allows for incremental and safe rollouts, rollback mechanisms, and performance-driven optimizations [43].

A foundational practice within DevOps is IaC, which formalizes infrastructure provisioning and management through machine-readable, declarative configuration files typically written in JavaScript Object Notation (JSON), Yet Another Markup Language (YAML), or HashiCorp Configuration Language (HCL). IaC enables consistent and repeatable deployments by abstracting hardware and platform dependencies through tooling such as Terraform, Ansible, or Pulumi [44]. It also promotes collaboration through shared repositories, peer reviews, and automated validation workflows. This ensures that infrastructure changes undergo the same rigor and traceability as software code.

These declarative models are often aligned with a higher-level abstraction known as *network intent*, which defines the desired operational state or service objective without prescribing how that state is to be achieved. Intent-based models allow developers and operators to express requirements such as connectivity, performance, or security constraints, while delegating the translation and enforcement of these requirements to automated systems. This abstraction is increasingly used in conjunction with DevOps and GitOps workflows to ensure that changes are not only syntactically valid, but also aligned with business or service-level objectives [45, 46].

CI/CD pipelines are the operational model of DevOps environments. These pipelines automate the end-to-end steps from code commits to system deployment, encompassing unit testing, static code analysis, integration testing, compliance verification, and runtime deployment [47]. Mature CI/CD pipelines are instrumented with telemetry hooks that collect performance and availability measure data, enabling observability and facilitating Root-Cause Analysis (RCA) in the case of failure [48]. This real-time feedback loop ensures that changes made to infrastructure or applications can be validated and optimized continuously, allowing teams to respond rapidly to emerging issues or performance regressions.

DevOps also emphasizes the cultural and organizational shift towards shared accountability between developers and operators. Traditional silos—where development teams handed over software artifacts to operations teams—are replaced by cross-functional teams that own services throughout their lifecycle. This integration has been shown to increase system reliability, shorten lead times for changes, and improve recovery from incidents [49]. However, achieving this alignment often requires changes not only in tooling but also in organizational mindset, workflows, and incentives.

While initially adopted in software-centric domains, the principles of DevOps have gained traction in other areas that involve complex infrastructure and high-frequency change cycles. Domains such as automotive systems, where software-defined vehicles require frequent updates [50], and telecommunication networks, where Virtualized Network Functions (VNFs) are dynamically instantiated and scaled [51], have increasingly adopted DevOps tooling and methodologies to streamline operations and ensure continuous service delivery. These use cases reveal both the potential and the challenges of DevOps in infrastructure-heavy environments: while automation and continuous deployment reduce configuration drift and accelerate delivery, they also demand stringent validation mechanisms, strong version control discipline, and robust rollback procedures to prevent cascading failures.

The application of DevOps in such domains highlights its adaptability to scenarios where safety, compliance, and distributed infrastructure play an important role. However, this same complexity introduces operational challenges, particularly in terms of managing the proliferation of automation scripts, CI/CD configurations, and interdependent pipelines. Moreover, the abstraction of infrastructure through code does not eliminate the underlying complexity—it shifts the burden toward understanding, testing, and maintaining the logic encoded in declarative models and automation flows.

By providing a systematic approach to automation, monitoring, and lifecycle management, DevOps transforms static and manually managed systems into resilient, self-correcting, and scalable platforms [39]. In the context of this thesis, DevOps provides the foundational methodology upon which further layers of automation—such as NaC, GitOps, and MLOps—are constructed. These subsequent paradigms extend DevOps principles into the domain of network and ML system orchestration, enabling scalable, version-controlled, and intent-driven management of highly dynamic infrastructures.

### 2.3.2 Network as Code: Redefining Network Management

Building upon the foundational practices established by DevOps, the concept of *Network as Code* (NaC) represents a fundamental shift in how network infrastructures are designed, provisioned, and operated. While IaC has traditionally focused on compute and storage layers, NaC extends these principles to the network domain, treating topologies, routing policies, service definitions, and device configurations as declarative, version-controlled software artifacts [52]. This approach enables reproducibility, automation, and traceability in managing complex infrastructures, aligning network operations with the agility and rigor of modern software delivery pipelines.

NaC encapsulates network state within structured, machine-readable models using formats such as YAML or JSON. These configurations define both static parameters (e.g., addressing, ACLs, BGP neighbors) and dynamic behaviors (e.g., service chaining, path selection, fault recovery) [53]. Stored in version control platforms like Git, they support workflows based on peer review, diff validation, and automated deployments—enabling team collaboration, rapid iteration, and rollback.

Model-driven architectures based on YANG schemas have further matured NaC frameworks [54], allowing orchestrators to interact with heterogeneous devices via vendor-neutral abstractions. High-level service intent—expressed in terms of availability, latency, or bandwidth objectives—is translated into device-level configurations by orchestration engines. This decoupling of intent and implementation simplifies service design and ensures compliance with operational policies [45, 46].

In dynamic environments—such as edge, multi-cloud, or disaggregated topologies—NaC enables reactive and automated updates through CI/CD pipelines. Declarative models serve as the single source of truth, and reconciliation agents ensure

that live infrastructure converges toward the desired state [55]. These pipelines typically validate changes through syntax checks, policy enforcement, simulations, and staged deployments, minimizing service disruptions and configuration drift.

By aligning configuration changes with Git-based workflows, NaC introduces a disciplined, auditable, and collaborative process to network engineering. Operators can implement changes through feature branches, submit pull requests, and trigger automated validation stages before deploying updates. These workflows improve transparency, reduce human error, and enable team-wide accountability [56]. They also facilitate integration with telemetry systems and intent frameworks that adapt the network state in response to real-time KPIs.

NaC supports modular and reusable configuration logic, which is particularly useful in heterogeneous or multi-domain networks. Whether orchestrating overlays, underlays, or containerized networks, NaC provides a unifying abstraction layer that minimizes vendor lock-in and promotes interoperability [57]. It also integrates with SDN controllers, service mesh agents, or IBN engines to support event-driven operations and autonomous policy enforcement.

Despite its benefits, NaC presents several adoption challenges. Declarative modeling requires specialized knowledge, and the transition from imperative CLI workflows to Git-based automation can face organizational resistance. Errors in configuration templates—when propagated via CI/CD—may lead to network-wide disruptions unless validation pipelines are robust. Managing complex inheritance chains, branching strategies, and schema changes in large-scale deployments also introduces tooling and governance overhead. Furthermore, even with standardized models, differences in device capabilities and semantics across vendors may necessitate custom logic or adaptation layers to ensure correctness.

In summary, NaC extends software engineering principles to networking by treating configuration as code, enabling scalable, reliable, and auditable network management. However, its effective adoption demands investment in modeling standards, validation tooling, team training, and interface normalization to fully unlock its potential in modern network environments.

### 2.3.3 GitOps: A Declarative Approach to Network Management

GitOps extends the principles of DevOps and NaC by introducing a declarative and version-controlled operational method in which Git acts as the central source of truth for system and network configurations. Rather than relying on imperative

commands or ad hoc scripts to manage infrastructure, GitOps enforces a closed-loop reconciliation process where the intended system state, expressed in machine-readable files (e.g., YAML or JSON), is automatically compared and synchronized with the actual system state [58].

In a GitOps-based workflow, configuration artifacts representing the desired network or infrastructure state—including topology definitions, routing policies, device parameters, and access rules—are committed to a version control system, such as Git. These changes are introduced via pull requests, enabling rigorous code review, audit trails, and automated validation through CI/CD pipelines. Upon approval and merging, GitOps controllers—such as ArgoCD or Flux—observe the repository for changes and reconcile them against the running system using declarative deployment strategies [59, 60]. If any drift or inconsistency is detected, the controllers automatically bring the system back to the intended state, ensuring alignment between configuration and realization.

This approach yields several operational advantages. GitOps provides strong guarantees of reproducibility and traceability, as every configuration change is recorded, versioned, and attributable. Rollbacks are simple, safe, and deterministic: reverting a misbehaving deployment involves checking out a previous Git commit, which the controller then enforces on the live system. These properties reduce manual intervention, lower the risk of configuration drift, and accelerate the mean time to recovery after failure [61].

GitOps also enhances organizational transparency and compliance. All modifications to infrastructure are made through version control system workflows, subject to fine-grained access control, code review policies, and cryptographic signing. This aligns with regulatory and governance requirements in sectors like healthcare, finance, and government IT, where auditability and operational control are essential [62]. Moreover, GitOps encourages collaboration between development, operations, and security teams by unifying them around a shared, declarative interface for managing infrastructure.

Despite its advantages, GitOps introduces several operational challenges. Declarative definitions must be carefully validated, as misconfigured files can lead to unintended system behaviors or service outages. In complex deployments, the reconciliation logic may be difficult to debug when dependencies between services or components are not well defined. Furthermore, performance and scalability considerations arise when applying GitOps to large-scale environments with high update frequency, especially when reconciling across distributed clusters or edge nodes. These issues

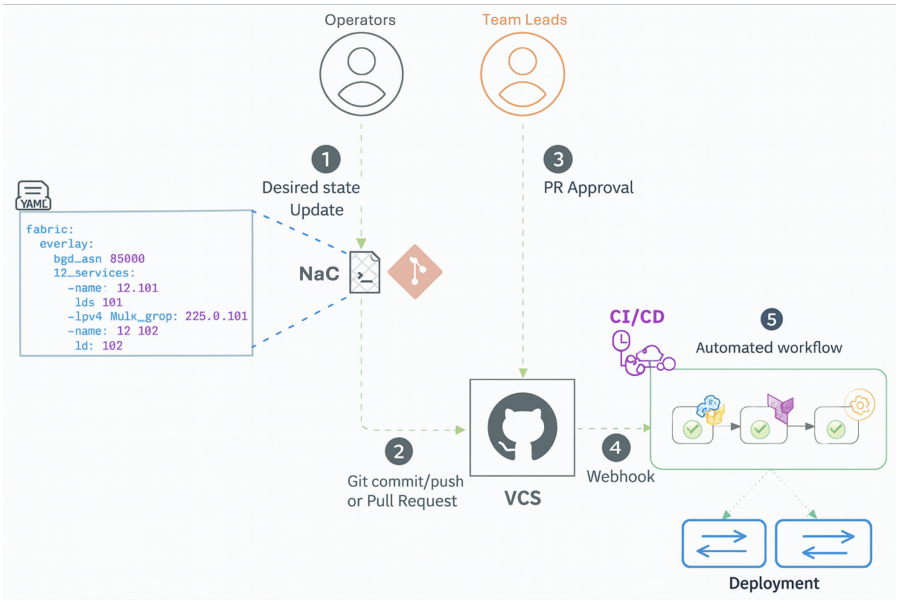



**Fig. 2.3.:** Integrated control stack combining DevOps, Network as Code, and GitOps.

demand robust pipeline testing, clear modularization of configuration, and careful management of reconciliation intervals to ensure system stability.

The relationship between DevOps, NaC, and GitOps is illustrated in Figure 2.3. DevOps defines the automation and collaboration patterns, NaC provides structured modeling of network state, and GitOps operationalizes these models by ensuring consistent deployment and self-healing behavior through continuous reconciliation. Together, these methodologies form a coherent stack for programmable and resilient infrastructure management.

GitOps has been widely adopted in cloud-native environments, particularly within Kubernetes ecosystems, due to its scalability, modularity, and ease of automation. Its principles are now being extended to other domains, including edge computing [63], 5G orchestration [64], and hybrid cloud management, where reproducible and auditable deployments are essential.

In the next section, we examine how these declarative and automated practices are further reinforced by cloud-native microservice architectures, which provide the modular runtime environment needed to scale and isolate control functions across complex infrastructure domains.

### 2.3.4 Cloud-Native Microservice Architectures

The automation-centric principles introduced by DevOps, NaC, and GitOps find their operational foundation in cloud-native microservice architectures. These architectures provide the structural and runtime environment required to implement modular, observable, and dynamically managed systems, especially in domains characterized by distributed infrastructures, heterogeneous components, and evolving service demands.

At their core, cloud-native architectures are defined by the decomposition of applications into *microservices*—independently deployable, loosely coupled components that encapsulate narrowly scoped functionalities. Each microservice is responsible for a specific domain/application (e.g., user authentication, resource scheduling, telemetry collection) and communicates with other services through lightweight protocols such as REST, gRPC, or message queues. This separation of concerns facilitates parallel development, continuous delivery, and functional isolation, in contrast to monolithic systems where components are tightly integrated and deployed as a single unit [65, 66].

Microservices are typically packaged in *containers*, which include all dependencies, runtime libraries, and configuration required for execution. These containers are orchestrated by platforms such as Kubernetes, which automate deployment, scheduling, scaling, fault recovery, and lifecycle management across clusters of compute nodes. Declarative manifests—often written in YAML—define the desired deployment state, including service replicas, resource quotas, networking rules, and health checks. Kubernetes continuously reconciles the actual system state with this declarative intent, providing a self-healing and adaptive runtime environment [67].

This paradigm is inherently compatible with NaC and GitOps workflows. Each microservice's configuration, environment, and orchestration logic is defined as version-controlled code, stored in Git repositories and subject to collaborative development workflows. Updates are delivered through CI/CD pipelines, while GitOps agents monitor the repository for changes and ensure consistent propagation to the live infrastructure. In this way, microservices become not just functional units but also policy-bound entities whose behavior and deployment are governed by declarative automation loops.

Cloud-native microservices offer several advantages. They enable independent scaling of system components, allowing operators to respond elastically to changing

workload patterns. They also support fault containment, where failures in one microservice do not cascade system-wide, increasing availability and reducing recovery time. From a development perspective, microservices promote agility: teams can build, test, and release new features in isolation, reducing coordination overhead and accelerating innovation cycles.

Industries such as finance, e-commerce, and content delivery have embraced microservices to meet the demands of high availability, low latency, and continuous feature rollout. Their success illustrates the effectiveness of this architectural model in managing distributed and mission-critical systems [65, 66, 67].

### 2.3.5 Operational Complexity Considerations

While cloud-native microservice architectures provide significant advantages in terms of modularity, scalability, and agility, their adoption in the context of disaggregated optical networks introduces a new layer of operational complexity that must be carefully examined.

One of the primary challenges arises from the large number of microservices that must be managed. In large-scale deployments, it is not uncommon to have hundreds of microservices, each responsible for a specific function such as topology discovery, telemetry ingestion, path computation, or policy enforcement. This increases the complexity of service orchestration, lifecycle management, and dependency tracking. Each microservice may have its own configuration, versioning scheme, and resource requirements, leading to a highly heterogeneous environment. Ensuring consistency across these components, particularly during upgrades or rollbacks, becomes a non-trivial task that requires sophisticated orchestration frameworks and strict version control mechanisms.

Furthermore, inter-service communication introduces additional operational overhead. Microservices typically interact through APIs, message brokers, or event-driven mechanisms, which increases the number of communication paths within the system. As the number of services grows, the communication graph becomes increasingly complex, leading to challenges in debugging, performance optimization, and fault isolation. Latency and reliability of inter-service communication can also impact the overall system performance, especially in time-sensitive network control loops.

Another critical aspect is observability. In monolithic systems, monitoring and troubleshooting are relatively straightforward due to the centralized nature of the application. However, in a distributed microservice architecture, achieving

end-to-end visibility requires the integration of distributed tracing, logging, and metrics collection across all services. This implies the deployment of additional monitoring infrastructure and the adoption of standardized telemetry frameworks. The correlation of events across multiple microservices to diagnose issues becomes significantly more complex, particularly in scenarios involving cascading failures.

Resilience and fault management also become more challenging in such environments. While microservices are designed to be independently deployable and fault-isolated, failures in one service can propagate through dependencies and affect other components. Designing robust failure handling mechanisms, such as circuit breakers, retries, and fallback strategies, is essential but adds further complexity to the system design. Additionally, the dynamic nature of cloud-native environments, where services can be scaled up or down and instances can be frequently created or terminated, complicates state management and consistency guarantees.

Security is another dimension where complexity increases. Each microservice exposes interfaces that must be secured, leading to a larger attack surface compared to monolithic systems. Implementing consistent authentication, authorization, and encryption mechanisms across all services requires a comprehensive security framework and careful policy management. The need to manage secrets, certificates, and access controls at scale further contributes to the operational burden.

Finally, the human and organizational aspects should not be overlooked. Operating a cloud-native microservice architecture requires new skill sets, including expertise in containerization technologies, orchestration platforms such as Kubernetes, and DevOps practices. Teams must adopt new workflows and tools to manage continuous integration and continuous deployment (CI/CD) pipelines, infrastructure as code, and automated testing. This transition can introduce a steep learning curve and necessitates significant changes in operational processes.

In summary, while cloud-native microservice architectures are a key enabler for programmability and automation in disaggregated optical networks, they inherently introduce substantial operational complexity. Addressing these challenges requires a holistic approach that combines advanced orchestration, observability, and automation techniques with well-defined operational practices. These considerations are fundamental for ensuring that the benefits of microservices can be effectively realized without compromising system reliability and manageability.

In the context of programmable networks—explored in the next sections—cloud-native microservice architectures serve as a foundational enabler. They allow traditionally monolithic control-plane functions to be modularized into independently

developed, deployed, and scaled services. When combined with GitOps workflows and declarative configuration models, this architecture supports robust orchestration of network services with software-level flexibility, reliability, and automation.

### 2.3.6 Convergence of DevOps, NaC, GitOps, and Optical Networks

The convergence of DevOps, NaC, and GitOps with optical network control marks a decisive evolution in the way transport infrastructures can be managed, transitioning from rigid, vendor-specific practices toward programmable, automated, and software-driven paradigms. These methodologies—originally developed to support agile software delivery and infrastructure management in cloud-native environments—offer a coherent framework for managing the growing complexity of disaggregated, multi-vendor optical networks [68, 69].

Optical transport networks are increasingly composed of heterogeneous elements—such as transponders, ROADMs, amplifiers, and performance monitors—sourced from different vendors and integrated across layered protocol stacks. This heterogeneity poses significant operational challenges: configurations must be precisely synchronized across diverse interfaces; service provisioning must account for physical-layer impairments; and fault recovery requires coherent control-plane behavior spanning multiple administrative and technological domains. Traditional management systems, often based on manual workflows and proprietary tooling, lack the flexibility and automation needed to handle such dynamic and distributed environments.

DevOps introduces automation and reproducibility into this landscape by enabling optical network control functions and configurations to be treated as code. Using IaC practices, operators can define optical network topologies, service intents, and device parameters as structured data models—typically encoded in YAML or JSON—and commit them to version-controlled repositories. These artifacts are then validated, tested, and deployed using CI/CD pipelines [70]. This approach transforms the management of optical control planes into a systematic, traceable, and automated process aligned with modern software engineering workflows.

NaC extends these principles beyond traditional infrastructure to include detailed, domain-specific configurations of optical devices. Power settings, modulation formats, channel spacing, protection schemes, and service templates are abstracted into reusable, declarative models that can be integrated into CI/CD pipelines and orchestrated across vendors. The use of standardized data modeling languages, such as YANG, facilitates interoperability by exposing consistent APIs across heterogeneous systems [71]. NaC thus provides a unifying layer that abstracts vendor-specific

behavior, reduces configuration drift, and promotes consistency in service deployment.

GitOps operationalizes this model by tightly coupling configuration state with version control. Every change to the optical control stack—whether in topology, policies, or service definitions—is made through Git commits, reviewed via pull requests, and applied through automated reconciliation mechanisms driven by tools such as Argo CD. This ensures that the runtime state of the network always reflects the declared state, enabling rapid rollbacks, drift detection, and auditability [72]. In the context of optical systems, this is particularly valuable, as misconfigurations can result in degraded optical signal quality or service disruption across wide areas.

Integrating these methodologies into the optical domain brings numerous benefits. The control and management plane becomes modular, observable, and responsive. Service activation becomes faster and less error-prone. Operators gain full traceability of changes, and the platform becomes more resilient through automated validation, testing, and rollback. These capabilities are essential for meeting the growing demands for agility, high availability, and service-level assurance in modern optical transport infrastructures.

Nonetheless, unique challenges remain. Unlike software-only systems, optical networks must operate under strict physical constraints—such as optical reach limitations, chromatic dispersion, nonlinear impairments, and power budgets—that are difficult to model and validate using generic DevOps toolchains. Furthermore, the diversity of vendor data models and the lack of fully standardized APIs complicate cross-domain orchestration. Continuous integration pipelines must therefore incorporate domain-specific knowledge, such as optical feasibility engines or impairment-aware provisioning logic, to ensure correctness and robustness.

To address these gaps, we propose a comprehensive implementation that integrates DevOps, NaC, and GitOps into a cohesive control framework tailored for open and disaggregated optical networks. The resulting system, referred to as *Network Management as Code* (NMaC), unifies topology modeling, control-plane orchestration, and declarative service provisioning within a GitOps-based automation workflow. It enables both static and dynamic reconfigurations through code-driven interfaces, supports vendor-agnostic deployments via standard APIs, and ensures continuous synchronization between intent and infrastructure through automated pipelines.

The next sections introduce the architectural design and technical components of the NMaC framework. We describe how declarative models, micros-ervice orchestration,

and continuous deployment mechanisms are instantiated in a real-world optical testbed, showcasing the feasibility and scalability of software-defined, automation-native control in production-like environments.

This implementation demonstrates a significant step toward intent-based, self-adaptive transport infrastructures, where control logic is abstracted, composable, and verifiably correct.

# 3 Contribution 1. Open Disaggregated Optical Network Control with Network Management as Code

## 3.1 Introduction

The transition from traditional vendor-specific to disaggregated optical transport networks leads to the expansion of equipment inventory. Thus, operators need to have a unified mechanism to manage the topology and device information since various vendors use different approaches to report their network elements. Besides, open interfaces such as T-API [25] allow network control functions to migrate from monolithic software stack on purpose-built hardware to more fine-grained microservice components deployed in the cloud infrastructure [73]. Therefore, softwarization plays an important role in Optical Network Automation [74]. Nevertheless, there exists a requirement on the specific versions of data models implemented in the functions, which hinders the interoperability between them. Adaptive and efficient software practices are needed to cope with the constant evolution of Open Optical Software Defined Networking (OO-SDN) and Virtualization technologies. In addition, both automated and manual deployment procedures are being applied which may cause a potential version mismatch if they are not tracked properly.

To tackle these challenges, we introduce a novel application of GitOps [75]—an operational model that integrates version control, Infrastructure as Code (IaC), and Continuous Delivery (CD)—into the management and orchestration of optical transport networks. To the best of our knowledge, this work is the **first application of GitOps principles to open and disaggregated optical networks**, establishing a new paradigm for automating both control-plane deployment and topology management in this domain.

GitOps builds on the concept of IaC, which is the process of managing and configuring cloud infrastructure under a definition file. Any changes to the file will be monitored

by an IaC tool (i.e., Terraform). Consequently, the state of the infrastructure is kept synchronized with the one described in the file. In addition, GitOps incorporates the functionality of Git repositories, Merge Requests (MRs), and Continuous Delivery (CD) to further unify software deployment and infrastructure operations. Any change to infrastructure is committed to the git repository along with application changes. GitOps leverages Git [76] as a single source of truth for both infrastructures and applications.

In this chapter, we introduce Network Management as Code (NMaC) in open disaggregated optical networks by adopting GitOps techniques. The principles of NMaC are to automate i) optical topology management and ii) network control function deployment. Particularly, an Automation Engine (AE) is designed to analyze topology deviations and make decisions on the suitable control plane architecture. In combination with AE, ArgoCD [77] acts as GitOps operator to form the NMaC tool, in order to trigger the CD procedure of SDN applications on a Kubernetes cluster. We validate this solution in the autonomous migration scenario from partially to fully disaggregated network. Thanks to the implementation of T-API version 2.4, seamless multi-vendor network control is achieved.

## 3.2 NMaC Framework Description and Implementation

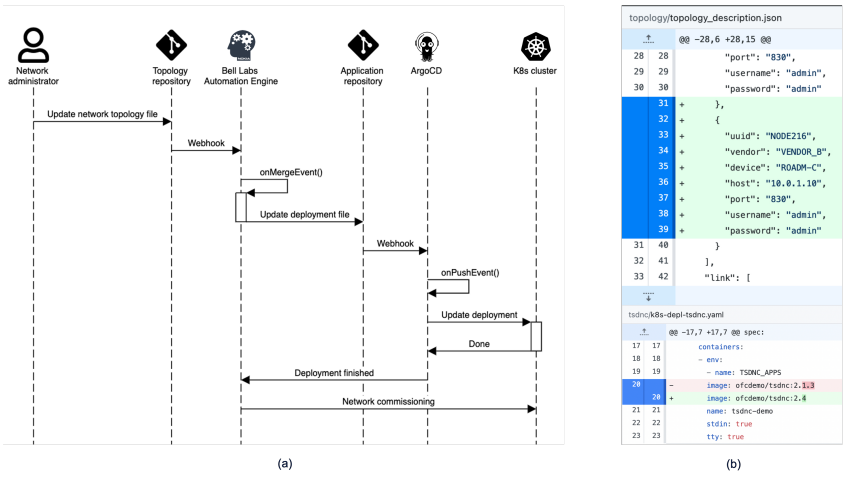



**Fig. 3.1.:** (a) NMaC Workflow. (b) Network Topology (top) and Application Deployment (bottom) file changes.

We showcase the complete autonomous management workflow of network infrastructure and control function deployment in open disaggregated optical networks using the NMaC concept, as depicted in Fig. 3.1a. NMaC architecture consists of an

AE, a GitOps CD tool, a Docker registry, a Kubernetes cluster and two Git repositories. To check the consistency of the deployed optical network, we adopt GitOps operations. Given the distinct purposes of topology management and control function deployment in open disaggregated optical networks, two repositories, namely Topology and Application, are created. These repositories represent the central source of truth of each NMaC principle. The network infrastructure is described in a topology definition file and stored in the Topology repository. On the other hand, deployment manifests are specified in the Application repository.

The AE acts as a reconciliation mechanism between the Topology and Application repositories. We configure the Topology repository with a webhook on merge events to notify AE of topology changes events automatically. When the network topology file is updated by a network administrator (see Fig. 3.1b), the AE analyzes the network infrastructure and makes the decision to upgrade/downgrade the version of the SDN Controller (SDNC) application and to add/remove network control functions in the deployment file (see Fig. 3.1b). We use ArgoCD as the GitOps CD tool to ensure the automatic deployment on the Kubernetes cluster. It synchronizes the desired state as declared in the repository with the actual state installed on the Kubernetes cluster. A webhook on push event is used to notify ArgoCD when there is a new commit on the deployment manifest for faster reaction time. Then, it will automatically sync the deployment to match the desired state. Subsequently, network administrators do not need to interact with the cloud infrastructure but only with the Topology repository interface. When the topology file is modified, the AE updates the deployment YAML of the Application repository by upgrading/downgrading the version of the Transport-SDNC (T-SDNC) application and by adding/removing network control functions. These new changes are captured by Argo CD. Argo CD will report the differences between the currently deployed infrastructure and desired state of the deployment YAML and will automatically sync the deployment to match the desired state. The state of the deployment can be monitored with ArgoCD's Graphical User Interface (GUI). Finally, ArgoCD informs AE once the deployment is finished so that it can launch the network commissioning process.

We consider two Git operations for code updates in two repositories. In the Topology repository, the network administrator acts as a contributor and forks the shared project to their own one. They can submit the modified topology definition file and open a *Pull Request* (PR) to notify other maintainers of the change that they want to merge to the original repository. Involved contributors, such as technicians and operation engineers, can review and validate the PR. This operation provides a mechanism to safely verify the integrity of the network among the collaborators. Upon the approval, PR can be merged to the main repository and consequent actions

are applied accordingly. On the other hand, the *Push* operation is applied for the Application repository since control functions deployment has been well studied by the AE and code approval procedure could be mitigated.

Additionally, we introduce for the first time a T-API v2.4 implementation in the SDNC. TAPI v2.4 provides new features over previous releases (i.e., T-API v2.1.3) such as equipment profiles, optical impairment parameters and event streaming. This enhancement enables the SDNC to manage directly Optical Impairment (OI) parameters in the topology, connectivity and path computation services without the need of intermediate OLS controllers. Thus, a mapping between the OpenConfig based device configuration and T-API v2.4 is implemented. On the Northbound, every control function interfaces with the SDNC using T-API v2.4. The external OI-aware Path Computation Engine (PCE) application performs routing, modulation scheme and spectrum assignment on the fully disaggregated network based on the T-API topology state. In addition, the database and (GUI) are upgraded to store and visualize the new T-API v2.4 context. Also, Kafka broker is used to stream T-API event notifications between the SDNC and network applications.

To validate the proposed NMaC architecture and its integration into real-world optical infrastructures, we design and deploy a comprehensive demonstration that showcases autonomous control plane orchestration, multi-vendor disaggregated topology management, and dynamic service restoration. The demonstration leverages GitOps principles for lifecycle automation, and is structured around two operational scenarios: a *Partially Disaggregated Network* and a *Fully Disaggregated Network*.

All control and orchestration components are deployed as microservices within a Kubernetes cluster, while the optical infrastructure is hosted on a remote physical lab composed of Nokia 1830 PSS [78] and transponders from Vendor-A and Vendor-B. Network configurations, topologies, and deployment manifests are stored in Git repositories and are version-controlled using Git. Topology changes are committed as pull requests, and infrastructure changes are triggered via ArgoCD and reconciled by our AE.

Figure 3.2 illustrates the transition from the initial partially disaggregated state to a fully disaggregated optical transport architecture, automatically orchestrated using the NMaC framework.

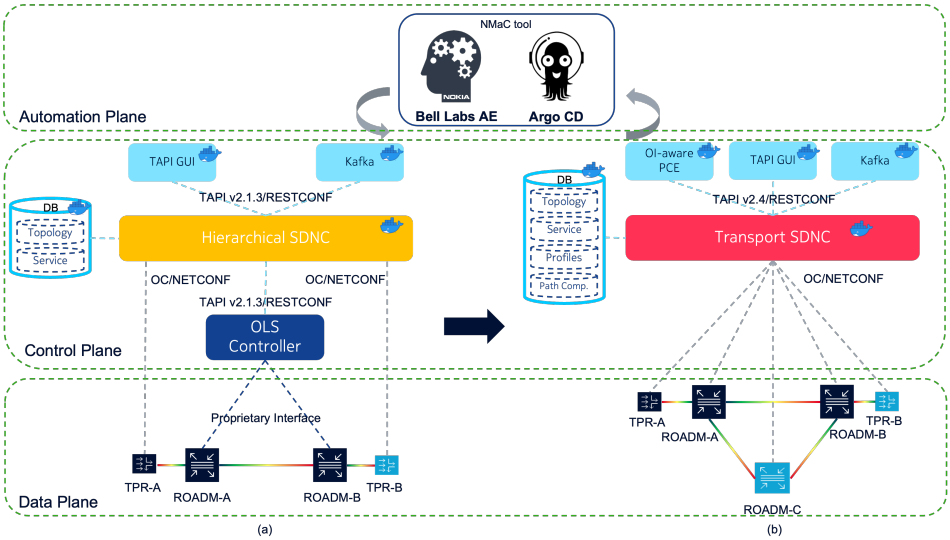



**Fig. 3.2.:** Transition in Topology and Control Plane from (a) Partially to (b) Fully Disaggregated Network managed by NMaC

### 3.2.1 Key Components of the NMaC Architecture

#### Transport SDNC

As previously mentioned, it is required to adopt an open architecture able to cope with a multi-vendor ecosystem. For that reason, we selected an open source controller designed for controlling disaggregated optical networks: OpenDayLight/Transport-PCE (T-PCE)[79]. T-PCE is an Optical SDN controller that automatically creates, edits, and deletes optical connectivity services in a multi-vendor optical network. It relies on "OpenDayLight" as a development framework and "OpenROADM" for service, network, and device data models [5], ensuring interoperability through standardized YANG schema definitions.

However, OpenROADM data models lack full adoption from the open-source community and are not aligned with Nokia's internal product strategy. That is why we decided to extend the capabilities of T-PCE by developing additional modules to support both broader southbound interoperability and more expressive northbound abstractions. Specifically, we implemented two major enhancements.

First, on the South Bound Interface (SBI), we introduced support for OpenConfig-based transport devices. This enhancement enables the controller to manage devices that rely on OpenConfig YANG models, such as `terminal-device`, `optical-channel`, and `platform` modules, via NETCONF. Since OpenConfig focuses solely on device-level modeling and does not include native support for network topology or service

abstractions, we implemented a mapping layer that converts OpenConfig device data into T-API topology constructs. This mapping allows OpenConfig-managed nodes to be fully represented in the controller's unified topology and to participate in service provisioning workflows. Conceptually, OpenROADM models can be seen as combining the functionality of both OpenConfig (device modeling) and T-API (topology and service modeling), which is why the T-API and OpenConfig approach offers a modular and interoperable alternative.

Second, on the North Bound Interface (NBI), we implemented support for the ONF T-API standard to expose abstracted views of the network topology, connectivity services, and path computation capabilities. T-API acts as a convergence layer, allowing orchestration systems and external SDN applications to interact with the network infrastructure in a vendor-agnostic manner. Throughout this process, we actively participated in the ONF T-API working group calls to ensure our implementation was aligned with ongoing community discussions and specifications. Our Transport SDNC now fully supports the T-API service model for topology discovery, connectivity provisioning, path computation, and event notification using both REST and gRPC interfaces.

A major contribution in this context was our proposal and prototype implementation of T-API model extensions for online OI-Aware Routing, Modulation and Spectrum Assignment (RMSA). These extensions introduced support for the advertisement and consumption of optical impairment parameters (e.g., Optical Signal-to-Noise Ratio - OSNR, chromatic dispersion, non-linear penalties) at the photonic media layer (Layer 0), as well as enhanced link state and quality metrics required for dynamic service computation. These proposals were discussed and validated within the T-API working group and ultimately adopted in release 2.4 of the standard. We extended the T-PCE to serve the enriched T-API v2.4 context, enabling real-time collaboration with external applications such as the GNPy-based PCE, Kafka-based monitoring systems, and GUI front-ends.

All code contributions related to the T-API plugin—including service, topology, and notification support—have been fully integrated into the official T-PCE open-source repository, further reinforcing its maturity and openness. The new architecture and T-API integration work were also presented to the broader open-source networking community during a session at the Linux Foundation Networking forum, showcasing the feasibility and readiness of T-PCE as a controller for Open and Disaggregated Optical Networks [80, 81].

Figure 3.3 illustrates the resulting architecture of the T-SDNC with the developed modules. The controller is now capable of realizing end-to-end optical services

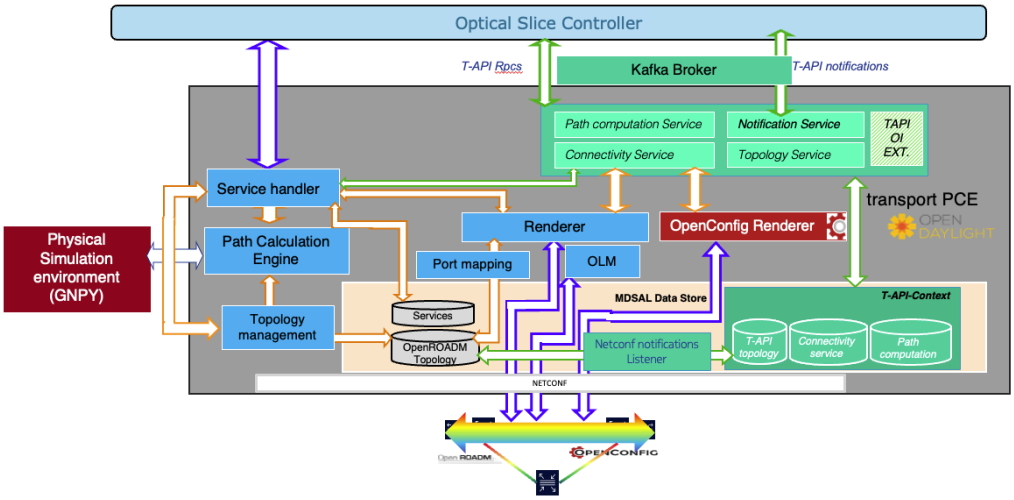



**Fig. 3.3.:** Nokia Bell Labs T-SDNC architecture

across multiple domains and vendor ecosystems, dynamically adapting its control plane capabilities in response to changes in the data plane and the evolution of open industry standards.

Thanks to the defined modular architecture, additional internal/external modules could be easily integrated to extend the functionalities of the controller. Each module is designed to perform a specific task.

For example, the **Service Handler/Connectivity Service** receives and manages service requests (e.g., service-create, service-delete, service-reconfigure...) from a parent controller or an orchestrator through the NBI as defined by the OpenROADM and T-API service models. It is in charge of triggering the corresponding modules to fulfill the service request.

The **PCE/Path Computation Service** is responsible for calculating a valid path across an optical domain managed by T-SDNC. A path calculation request can be received from both the service handler and an external component. To keep the PCE aligned with the latest changes in the topology, an interface to the Topology Management module is provided. The implementation of the internal PCE module follows the Shortest Path First Fit (SP-FF) allocation scheme, which finds the shortest route and the first available frequency slot over the end-to-end spectrum profile of the route. However, optical non-linearities can significantly affect optical services. Therefore, additional constraints must be taken into account when calculating the route. For this reason, we decided to interface our Transport SDNC with an optical impairement aware PCE, which will compute a set of Quality of Transmission (QoT) metrics for the pre-computed path and will validate the request of the PCE.

Next, the **Topology Management/Service** builds multi-layer topologies based on each of the supported models (OpenROADM and T-API). The topology is built dynamically as nodes are connected/disconnected to the controller. All configuration data is retrieved through the NETCONF protocol and stored in the database of the controller.

The **Renderer** serves the requests coming from the Service Handler. The Service Handler provides to the Renderer the path description calculated by the PCE. This description is based on the abstracted topological entities (node, links, termination points...), therefore the Renderer converts them into device resources (circuit packs, ports, interfaces...). Once the conversion is performed, the Renderer checks the existing interfaces (OTS – Optical Transmission Section, OMS – Optical Multiplex Section, etc) on the ports of the different nodes that the path description contains. If any required interface is missing, the Renderer will create it through a NETCONF command. Then, the Renderer establishes the connections between nodes. After all configuration commands have been successfully sent to the optical nodes described in the path description, the Renderer relies on the parameters computed by GNPy to set the power levels.

Finally, the **Notification Service** allows clients to subscribe to and filter autonomous notifications from the controller for events such as resource/service state change, failure or degradation, and telemetry.

### Kafka

Apache Kafka is used to implement a distributed, real-time event delivery system for inter-service communication. Kafka acts as the messaging backbone for publishing and subscribing to T-API event notifications. These include:

- **Connectivity Service Lifecycle Events**: Created, deleted, failed, restored.
- **Topology Changes**: Node/link addition, removal, degradation.
- **Telemetry or Alarm Events**: Streaming notifications tied to physical link conditions or service state.

Kafka decouples service producers (e.g., the SDNC) from consumers (e.g., GUI, monitoring agents) and guarantees reliable, ordered event delivery at scale. This enables a responsive and resilient control ecosystem.

### T-API GUI

The T-API GUI is a full-stack application that provides a user-friendly interface for visualization and control of the optical network. It is composed of:

- **Backend**: Built using Django, it interfaces with the T-API NBI exposed by the SDNC and performs REST calls for topology queries, service creation, and state polling.
- **Frontend**: Built using ReactJS, it renders the discovered topology graph, service paths, and node status. It supports visual service provisioning and displays real-time status.
- **Kafka Integration**: Subscribes to relevant topics to receive T-API event notifications (e.g., service status, alarms), updating the GUI automatically.

This component allows operators to interact with the system at a high level, monitor topology changes, provision services, and track service lifecycle states in real time.

### Optical Impairment aware PCE

Our OI aware PCE is built around GNPy, an open-source physical layer modeling engine developed by the Telecom Infra Project (TIP) under the Open Optical & Packet Transport (OOPT) initiative. GNPy, short for "Gaussian Noise model in Python," is designed to perform QoT estimation based on analytical models rooted in the Gaussian Noise approximation. It enables multi-vendor and transparent estimation

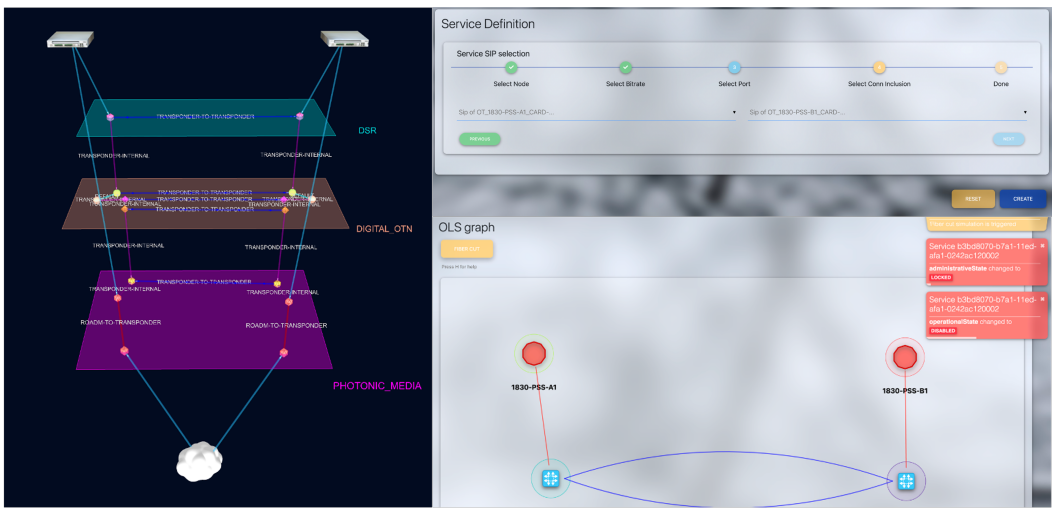



Fig. 3.4.: T-API GUI

of optical feasibility by simulating signal propagation through optical fiber links using a vendor-neutral methodology [82, 83, 84].

GNPy operates on abstract optical topologies represented in JSON or YAML format and simulates the physical impairments encountered by optical signals. These include linear effects like chromatic dispersion, polarization mode dispersion, and attenuation, as well as nonlinear effects such as self-phase modulation, cross-phase modulation, and four-wave mixing. In addition, it considers amplifier noise characteristics and span-level properties like length, slope, and loss. The result is an accurate estimation of signal quality metrics such as OSNR and non-linear penalties.

Within our proposed architecture, the GNPy engine is deployed as an external microservice that interacts with the SDNC to offload optical impairment validation. During the RMSA procedure, the SDNC exports a topology that follows the T-API model, augmented with physical impairment data. This data includes span loss, amplifier gain, and fiber-specific attributes necessary for accurate modeling. The GNPy service receives the proposed paths and configuration constraints—such as baud rate, channel spacing, or modulation format—and computes whether the end-to-end optical performance satisfies required thresholds.

If the proposed path meets the quality criteria, GNPy returns the validated route along with the corresponding QoT metrics. Otherwise, it returns a rejection due to optical infeasibility, preventing the SDNC from provisioning sub-optimal services. This process is instrumental in disaggregated optical network environments where vendor-specific QoT estimation mechanisms are either incompatible or unavailable. It ensures that end-to-end paths are not only logically consistent but also physically feasible.

The use of an external OI-aware PCE introduces several operational benefits. First, it separates the logical service orchestration from the physical performance validation, leading to more modular and maintainable control architectures. Second, it enhances restoration procedures by allowing dynamic recalculation of feasible paths during failure events, taking into account real-time impairment constraints. Third, it aligns with open networking principles by avoiding reliance on closed-form vendor-specific algorithms. As disaggregated transport networks evolve, impairment-aware orchestration will become a critical capability for maintaining service assurance, minimizing unnecessary over-provisioning, and ensuring efficient spectrum utilization.

### 3.2.2 Design Trade-offs, Scalability Considerations, and Critical Assessment

While the proposed NMaC architecture demonstrates the feasibility of automating control and management functions in open and disaggregated optical networks, a deeper examination of the underlying design choices is necessary to fully assess its applicability in large-scale and production environments.

A first important design decision lies in the adoption of a GitOps-based workflow as the central mechanism for triggering network operations. This approach provides strong benefits in terms of traceability, reproducibility, and version control of network configurations. By treating network intent and configurations as code artifacts stored in repositories, the system enables auditable change management and facilitates collaboration across teams. However, this choice also introduces inherent latency in the control loop, as changes must propagate through repository commits, validation pipelines, and deployment stages before being applied to the network. While acceptable for planned provisioning and configuration tasks, this model may not be suitable for time-critical control operations requiring near real-time responsiveness, such as fast restoration or dynamic adaptation to rapidly changing network conditions.

Another key architectural component is the automation engine responsible for interpreting repository changes and orchestrating the deployment of corresponding network functions. The decoupling between intent definition and execution enhances modularity and extensibility, allowing new automation workflows to be introduced without modifying the core system. Nevertheless, this abstraction layer introduces additional complexity in terms of dependency management and execution ordering. As the number of managed services and workflows increases, ensuring deterministic behavior and avoiding conflicting actions becomes more challenging. In particular, concurrent updates affecting shared resources (e.g., spectrum allocation or topology state) may lead to race conditions or inconsistent network states if not properly synchronized.

Scalability is another critical aspect that must be carefully considered. The NMaC architecture relies on a microservice-based control plane deployed over cloud-native infrastructure. While this enables horizontal scaling of individual components, it also introduces overhead associated with inter-service communication, state synchronization, and orchestration. In large-scale optical networks with frequent topology updates and high volumes of telemetry data, the control plane may experience bottlenecks in processing and propagating state information. Moreover, the reliance

on centralized repositories as the source of truth can become a limiting factor, both in terms of access contention and latency, particularly in geographically distributed deployments.

From a data consistency perspective, the architecture must balance between strong consistency and eventual consistency models. While strong consistency ensures correctness of network operations, it can negatively impact performance and scalability due to synchronization overhead. On the other hand, eventual consistency improves scalability but may lead to transient inconsistencies that could affect decision-making processes, especially in closed-loop control scenarios. The choice of consistency model therefore represents a fundamental trade-off that depends on the specific operational requirements of the network.

The integration with external systems, such as SDN controllers and domain-specific management platforms, also raises important considerations. While the use of standardized interfaces (e.g., YANG/NETCONF, gRPC) promotes interoperability, it introduces dependencies on the performance and reliability of these external components. Variability in response times, data models, or supported features across vendors may impact the overall system behavior and limit the portability of the NMaC approach across heterogeneous environments.

Finally, the current evaluation of the NMaC framework primarily focuses on demonstrating functional capabilities in a controlled environment. While this is an essential first step, a more comprehensive assessment would require quantitative analysis of key performance indicators, such as deployment latency, system throughput, scalability limits, and fault recovery times. In particular, evaluating the system under stress conditions (e.g., high frequency of configuration changes or large-scale topology updates) would provide valuable insights into its robustness and operational boundaries.

In summary, while the NMaC architecture represents a promising approach for enabling programmable and automated control of disaggregated optical networks, its practical adoption requires careful consideration of design trade-offs related to latency, scalability, consistency, and interoperability. Addressing these challenges is essential for transitioning from proof-of-concept demonstrations to production-grade deployments capable of supporting the stringent requirements of modern optical transport networks.

### 3.2.3 Demo Scenario Description

**Scenario 1: Partially Disaggregated Network**

The network includes two ROADMs (ROADM-A and ROADM-B) managed via a proprietary OLS controller. Transponders from Vendor-A (TPR-A) and Vendor-B (TPR-B) are installed and interconnected. The physical and logical topology, including nodes, links, and device interfaces, is declared in a YAML-based topology definition file and committed to the *Topology Repository*, as shown in Fig. 3.1b (top).

**Step 1: Control Plane Deployment.** Upon submission and merging of the pull request in the Topology Repository, a webhook event notifies the AE, which analyzes the new topology and determines the necessary network control functions based on the new topology. The AE then generates a deployment manifest specifying the set of containerized control applications required for this network scenario. This manifest is pushed to the *Application Repository*, triggering another webhook to notify Argo CD. Argo CD continuously monitors the Application Repository as the Git-based source of truth, and upon detecting the new commit, initiates a sync process to reconcile the desired state (as declared in Git) with the live state on the Kubernetes cluster.

ArgoCD proceeds to deploy the corresponding applications, which in this case includes a Hierarchical-SDNC (H-SDNC), Kafka broker, T-API GUI, and the underlying SDNC database. The H-SDNC is configured to manage the transponders via OpenConfig/NETCONF and communicates with the proprietary OLS controller through T-API v2.1.3. ArgoCD also tracks the health status of each microservice by periodically querying their liveness and readiness probes. If a container fails or becomes unresponsive, ArgoCD automatically reports the deviation in state and re-triggers a redeployment to restore consistency. This ensures the control plane remains aligned with the intended system configuration described in the Git repository.

**Step 2: Service Provisioning.** Once deployment is complete and the Kubernetes environment reaches a healthy state, the AE performs a commissioning process to discover devices and populate the SDNC database. The T-API GUI queries the SDNC's NBI to retrieve the network topology and visualize it. The OLS domain is abstracted as a single T-API node. The operator uses the GUI to create a connectivity service between TPR-A and TPR-B, which the H-SDNC provisions by configuring the transponders and coordinating with the OLS controller to establish the Network

Media Channel (NMC). Kafka is used to stream real-time state notifications about the connectivity service lifecycle to the GUI, ensuring end-to-end observability.

### Scenario 2: Migration to Fully Disaggregated Network

To demonstrate dynamic control plane adaptation and multivendor interoperability, a third node (ROADM-C) from Vendor-B is introduced into the topology. This new node is modeled using OpenConfig data models and connected through physical links to ROADM-A and ROADM-B.

**Step 3: Automated Migration.** The updated topology is committed to the Topology Repository and merged into the main branch via pull request. The AE detects the topology change and interprets it as a migration from a partially to a fully disaggregated architecture, since the newly added ROADM-C cannot be managed by the proprietary OLS Controller. This topology change is the key triggering event for the control plane evolution and is visually depicted in Fig. 3.2.

The AE modifies the Application Repository by replacing the H-SDNC and OLS controller with the Transport-SDNC (T-SDNC), enabling direct multi-vendor control through OpenConfig/NETCONF. Simultaneously, the AE introduces support for more advanced and expressive open standards—specifically, it upgrades the interface model from T-API v2.1.3 to T-API v2.4. This upgrade is crucial to support the enhanced topology abstraction, impairment-aware routing, and real-time event streaming required in a fully disaggregated optical network.

In addition, the AE deploys the OI-aware PCE (based on GNPy) and a new version of the T-API GUI capable of parsing and visualizing the richer T-API v2.4 context. ArgoCD monitors the Application Repository for this update, detects the new commit, and automatically reconciles the control plane. It terminates the outdated applications and deploys the updated set of microservices while continuously monitoring their health.

Once the new control plane is active and the system reaches a healthy state, the AE re-commissions all devices to populate the T-SDNC database with a T-API v2.4 topology view. This migration highlights the system's ability not only to update the control plane microservices but also to adapt to new open standard versions—driven by changes in the data plane such as vendor diversity and model upgrades.

**Step 4: Failure and Service Restoration.** A fault is simulated on the optical link between ROADM-A and ROADM-B (e.g., fiber cut). The T-SDNC detects the event through optical alarms and immediately publishes a notification to Kafka.

It simultaneously triggers a dynamic service restoration process. The OI-aware PCE is invoked to compute a new feasible path via ROADM-C, validating the route against QoT constraints. Once confirmed, the T-SDNC reconfigures the affected transponders and reroutes the Network Media Channel accordingly. Kafka broadcasts the restoration event, which is visualized in real-time by the T-API GUI. This closed-loop reaction illustrates the adaptive, automated, and standards-aligned capabilities of the NMaC framework.

This implementation demonstrates the feasibility and operational maturity of applying GitOps-driven automation, declarative infrastructure modeling, and cloud-native microservice orchestration to open disaggregated optical networks. Through the introduction of NMaC, we enable a reproducible and version-controlled workflow where infrastructure definitions and control functions are managed entirely through Git, ensuring traceability, consistency, and minimal human intervention.

The integration of the AE with ArgoCD forms a closed-loop system where any topology change triggers the automatic (re)deployment of the appropriate SDN control functions. This capability supports seamless transitions between partially and fully disaggregated configurations, introduces topology-aware orchestration, and enhances multi-vendor interoperability via T-API 2.4. The system achieves near real-time reaction to infrastructure changes, including impairment-aware service restoration, without requiring manual coordination across operational domains.

By abstracting human workflows into declarative code and automating both deployment and lifecycle management, the solution aligns with Level 3 ("Autonomous Operations") of the TM Forum's Autonomous Networks framework. At this level, control functions are self-configuring, self-healing, and operate with limited supervisory input.

This work constitutes the first practical realization of such an architecture in the optical transport domain, validating the convergence of DevOps, GitOps, and open SDN principles in delivering agile, interoperable, and future-proof network control.

# 4 Conclusion

Part I has presented a comprehensive exploration of the technologies enabling the shift from monolithic, vendor-specific systems to open, disaggregated, and fully automated optical transport networks. We examined how modern software engineering methodologies—specifically DevOps, NaC, GitOps, and cloud-native microservice architectures—can be effectively adapted to address the operational complexities of next-generation optical infrastructures. Each of these paradigms contributes a critical capability: DevOps brings continuous integration and deployment; NaC provides structured and auditable configuration management; GitOps ensures state consistency through declarative control and version tracking; and microservices offer modularity, scalability, and fault isolation.

Building upon these principles, we introduced the first complete implementation of *Network Management as Code* (NMaC) in the optical domain, marking a major milestone in the evolution of control plane automation. The proposed architecture brings together a topology-aware Automation Engine, GitOps tooling (ArgoCD), and modular SDN control components to enable a fully declarative and event-driven workflow. Control functions are dynamically instantiated, reconfigured, or retired based solely on changes committed to version-controlled network models—without the need for manual intervention. This demonstrates the feasibility of autonomously transitioning between partially and fully disaggregated network scenarios while maintaining alignment between the infrastructure and the control plane.

The contribution of this work lies not only in the application of cloud-native principles to optical networks, but in the creation of a reproducible, modular, and extensible platform capable of addressing real-world operational complexities. By supporting T-API version 2.4, including optical impairment parameters and notification mechanisms, the system demonstrates vendor-agnostic interoperability and readiness for advanced use cases such as QoT-aware provisioning. The integration of GitOps-based workflows provides full traceability, auditability, and rollback capabilities—critical attributes for safe automation in multi-domain environments.

Several key insights were gained through this implementation. First, declarative and automated network management significantly reduces the operational overhead

typically associated with optical transport systems. Second, version-controlled workflows mitigate the risks of configuration drift, enabling consistent deployments even in rapidly evolving infrastructures. Third, by decoupling topology representation from control logic, the architecture provides a flexible substrate for intent-based networking.

Nevertheless, certain challenges remain before transitioning to full operational maturity. These include validation of declarative models across heterogeneous vendors, robustness against configuration errors or misalignments, and improved observability of real-time network behaviors during dynamic changes. The current system operates reliably under controlled conditions but will require further hardening and scalability assessments to reach production deployment standards.

In terms of autonomy, the proposed NMaC framework demonstrates alignment with **Level 3** of the TM Forum Autonomous Networks framework—*partial autonomy*. At this level, the system achieves closed-loop automation for specific workflows (e.g., topology-based control plane deployment), reduces human involvement to supervisory actions, and provides full transparency into changes and decisions. It lays the groundwork for future extensions into Levels 3 and 4, where ML-assisted decision-making and real-time self-optimization become central.

In this context, the next part investigates how ML can elevate this architecture by introducing predictive, adaptive, and self-optimizing behaviors. While this part laid the foundation for software-defined automation, the next logical step is to embed cognitive intelligence into the control loops—enabling networks that not only react to changes, but anticipate and adapt to them proactively.

# Part II

## Machine Learning Foundations and Applications in Optical Networks

This part investigates the application of Machine Leaning (ML) techniques to the management and control of optical transport networks, building on the architectural foundations laid in previous chapters. As networks become increasingly dynamic, disaggregated, and service-driven, traditional deterministic approaches struggle to provide the responsiveness and adaptability required. In this context, ML offers a data-driven paradigm shift capable of addressing complex decision-making tasks under uncertainty and dynamic conditions.

We begin by reviewing the fundamental ML methods relevant to optical network management and control, addressing both predictive modeling and decision-making tasks. Particular attention is given to approaches capable of capturing temporal dynamics, spatial correlations, and adapting to changes in network behavior over time, which are essential for ensuring robustness and performance in dynamic and heterogeneous optical environments.

Then we present concrete research contributions that apply ML to address critical challenges in real-world optical network scenarios. These include a supervised learning approach for transponder parameter configuration, enabling predictive inference of optimal modulation formats and launch powers; DeepSF-PCE, a Deep Reinforcement Learning (DRL)-based Path Computation Engine (PCE) capable of making Spectrum Fragmentation (SF)-aware spectrum allocation decisions in a single-agent setting; MAGC-RSA, a Graph Convolutional Network (GCN)-based Multi Agent Reinforcement Learning (MARL) framework for distributed and topology-aware Routing, Modulation and Spectrum Assignment (RMSA) optimization; and DAEQoT, a Drift-Adaptive Ensemble (DAE) classifier designed to maintain robust Quality of Transmission (QoT) estimation under dynamic traffic patterns and evolving network states. All proposed solutions are empirically validated through detailed experimentation, highlighting measurable improvements in spectrum utilization, request blocking probability, and model resilience over time.

Through these studies, we illustrate not only the feasibility but also the operational value of ML-driven network intelligence in modern optical systems. At the same time, we critically discuss remaining challenges, including the impact of model drift, generalization limitations, and the integration of learning agents into latency-sensitive control loops. These observations set the stage for further exploration into scalable ML deployment strategies in future chapters.

# 5 Introduction

## 5.1 Machine Learning Paradigms for Network Intelligence

Artificial Intelligence (AI) represents a broad spectrum of computational methods aimed at replicating human cognitive processes, such as learning, reasoning, perception, and decision-making. These methods have become increasingly relevant as modern communication infrastructures grow in complexity, scale, and operational dynamism. AI techniques enable systems to process vast quantities of data, identify latent patterns, adapt to changing environments, and optimize decisions with minimal human intervention—capabilities that are critical in the pursuit of intelligent and autonomous network operations.

Within the field of AI, Machine Learning (ML) has emerged as a particularly influential discipline, offering data-driven models that learn from experience rather than relying on static rule-based logic. By analyzing historical data or real-time telemetry, ML algorithms can approximate complex mappings, uncover statistical dependencies, and make informed predictions or control decisions. This shift from explicit programming to model-based learning has opened the door to new forms of automation and adaptability across various domains, including networking, cloud infrastructure, and systems management.

ML approaches differ based on how data is presented to the algorithm and what type of feedback is available during learning. Broadly speaking, the field encompasses paradigms such as supervised learning, where models are trained on labeled datasets; unsupervised learning, which seeks to uncover structure in unlabeled data; and reinforcement learning (RL), which focuses on learning decision-making policies through interaction with an environment. Each paradigm is suited to different types of tasks and brings distinct algorithmic strategies and modeling assumptions.

In the context of network intelligence, these ML paradigms provide a foundational toolkit for addressing challenges in observability, control, fault tolerance, and resource optimization. As networks continue to evolve toward software-defined, virtualized, and distributed architectures, the ability to embed learning capabilities

into network functions becomes a fundamental enabler for automation at scale. This chapter introduces the theoretical foundations of these ML paradigms, discusses their modeling principles, and provides a conceptual basis for understanding their applicability to intelligent network systems. The concepts presented here will serve as a precursor to the more domain-specific applications explored in the following chapters.

### 5.1.1 Supervised Learning

Supervised learning constitutes one of the most foundational paradigms in the machine learning (ML) landscape. It is based on the assumption that a dataset of labeled instances is available, each consisting of a feature vector $\mathbf{x} \in \mathbb{R}^n$ and a corresponding ground truth label $y$. The learning algorithm's objective is to approximate a function $f : \mathbf{x} \rightarrow y$ that generalizes well to new, unseen data. This mapping can involve either discrete-valued labels (classification) or continuous-valued targets (regression). The overall supervised learning pipeline, illustrated in Figure 5.1, encompasses data preprocessing, model training using loss minimization, and inference.

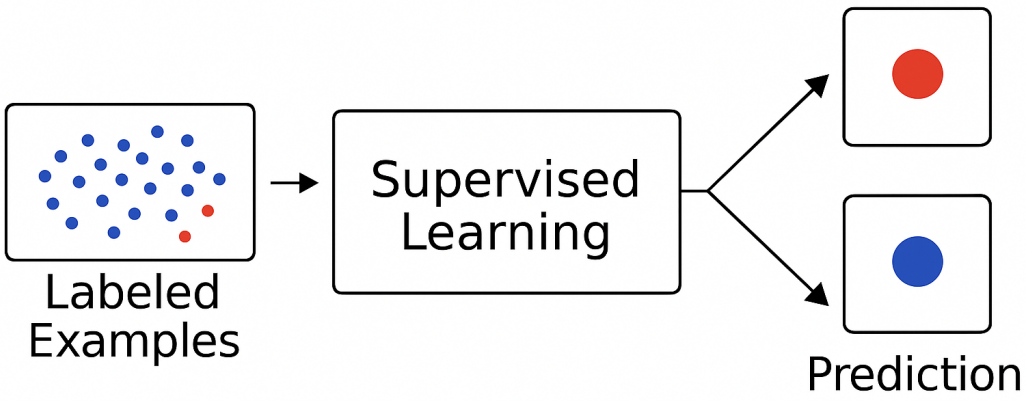



**Fig. 5.1.:** General architecture of supervised learning

The training process seeks to minimize a chosen loss function $\mathcal{L}(y, \hat{y})$ over the dataset, where $\hat{y}$ is the model's prediction. Loss functions such as mean squared error (MSE) and cross-entropy are commonly used for regression and classification tasks, respectively. Optimization is typically performed using stochastic gradient descent (SGD) or its variants (e.g., Adam, RMSProp), with regularization techniques like L1 and L2 penalties to control overfitting. In many applications, k-fold cross-validation is employed to ensure that the model achieves good generalization performance on unseen data.

Supervised learning is particularly suited to domains where labeled datasets are available or can be reliably generated. In telecommunications, this includes predictive maintenance using fault logs, traffic classification based on flow statistics, and quality estimation from link-level telemetry. In such domains, the input features often consist of statistical aggregates, protocol metadata, or signal characteristics derived from time-series measurements.

A wide spectrum of supervised learning algorithms has been developed, each offering distinct trade-offs in terms of accuracy, interpretability, computational complexity, and scalability. Linear models such as ordinary least squares regression and logistic regression provide fast and interpretable solutions, but their capacity to model non-linear relationships is limited. Tree-based models—including decision trees, random forests, and gradient boosting machines—offer more flexibility and robustness to feature interactions. They are particularly effective in tabular domains where feature importance and decision logic are crucial for operational validation.

Support Vector Machines (SVMs) are another prominent class of models that construct optimal separating hyperplanes in high-dimensional feature spaces. When combined with kernel methods, SVMs can capture complex, non-linear decision boundaries with relatively few parameters. Their use, however, can be computationally intensive on large-scale datasets.

Instance-based learning algorithms, such as k-Nearest Neighbors (k-NN), rely on local decision boundaries determined by distance metrics. These models are easy to implement and can perform well in low-dimensional settings, but scale poorly in terms of computation and storage as the dataset grows. On the other hand, probabilistic methods like Naive Bayes classifiers and Bayesian networks model the generative process of the data, allowing for the incorporation of prior knowledge and uncertainty quantification.

In recent years, artificial neural networks (ANNs) and deep learning have gained substantial traction for supervised learning tasks involving high-dimensional, un-

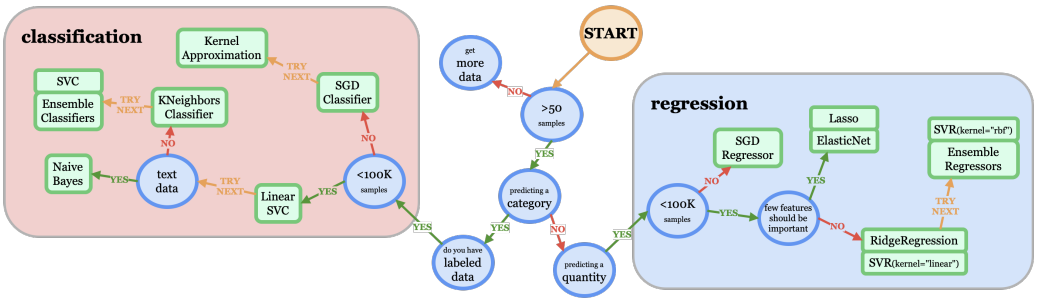



**Fig. 5.2.:** Popular algorithms used in supervised learning categorized by type.

structured data. Deep neural networks (DNNs), which consist of multiple layers of interconnected neurons, are capable of learning hierarchical feature representations through backpropagation. Activation functions such as ReLU, sigmoid, and tanh introduce non-linearities that enable the approximation of complex functions. Specialized architectures, including Convolutional Neural Networks (CNNs) for spatial data and Recurrent Neural Networks (RNNs) for sequential data, have further extended the applicability of supervised learning in domains such as audio processing, anomaly detection, and user behavior modeling [85, 86].

Despite its broad applicability and strong empirical performance, supervised learning has several limitations. One of the principal challenges is the reliance on large volumes of labeled data. In many real-world scenarios, obtaining labeled examples is time-consuming, expensive, or impractical. This issue motivates the development of semi-supervised, weakly-supervised, and active learning methods. Moreover, supervised models are often sensitive to distribution shifts between training and deployment environments, a phenomenon known as dataset shift or concept drift [87].

To mitigate these issues, recent advances have focused on techniques such as transfer learning, where models pre-trained on large generic datasets are fine-tuned on task-specific data, and data augmentation, which artificially inflates training data diversity. Automated Machine Learning (AutoML) tools have also gained popularity by automating feature selection, hyperparameter tuning, and model ensembling, thus lowering the barrier to entry for non-experts [88, 89]. Explainability and interpretability remain ongoing research concerns, particularly in regulated industries. Tools like SHAP and LIME are increasingly used to make complex models more transparent and auditable [90].

Figure 5.2 summarizes common supervised learning algorithms and their categorization based on learning strategy and model assumptions. Selecting the appropriate

model often requires a careful balance of accuracy, robustness, interpretability, and computational cost.

In conclusion, supervised learning remains a cornerstone of ML-driven network intelligence. Its ability to model complex relationships from labeled data makes it highly effective for a range of prediction and classification tasks. However, its reliance on labeled data and vulnerability to non-stationary environments underscore the importance of model validation, drift detection, and lifecycle management when deployed in operational settings.

### 5.1.2 Unsupervised Learning

Unsupervised learning represents a core paradigm in the ML landscape, wherein the objective is to uncover latent patterns, relationships, or structures in input data without the guidance of labeled outputs. This learning setup is particularly valuable in domains where manual annotation is costly, infeasible, or inherently ambiguous, and it facilitates the analysis of large-scale, heterogeneous datasets for tasks such as clustering, dimensionality reduction, anomaly detection, and density estimation.

As illustrated in Figure 5.3, the general unsupervised learning pipeline begins with raw, unlabeled inputs and typically produces high-level representations, groupings, or probabilistic models. Unlike supervised learning, which optimizes predictive accuracy with respect to ground truth labels, unsupervised algorithms operate by optimizing intrinsic properties of the data, such as similarity, variance, or reconstruction fidelity.

One of the most prominent objectives in unsupervised learning is clustering, where the aim is to partition data into groups such that intra-cluster similarity is maximized and inter-cluster similarity is minimized. Algorithms such as $k$-means, hierarchical agglomerative clustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and Gaussian Mixture Models (GMMs) provide different inductive biases and assumptions about the data distribution. For instance, $k$-means assumes isotropic cluster shapes and minimizes within-cluster variance, whereas DBSCAN can detect non-convex clusters and isolate noise without requiring the number of clusters a priori [91].

Dimensionality reduction constitutes another major branch of unsupervised learning. The goal is to transform high-dimensional data into a lower-dimensional representation that preserves essential information. Principal Component Analysis (PCA), a linear projection method, identifies orthogonal directions of maximum variance,

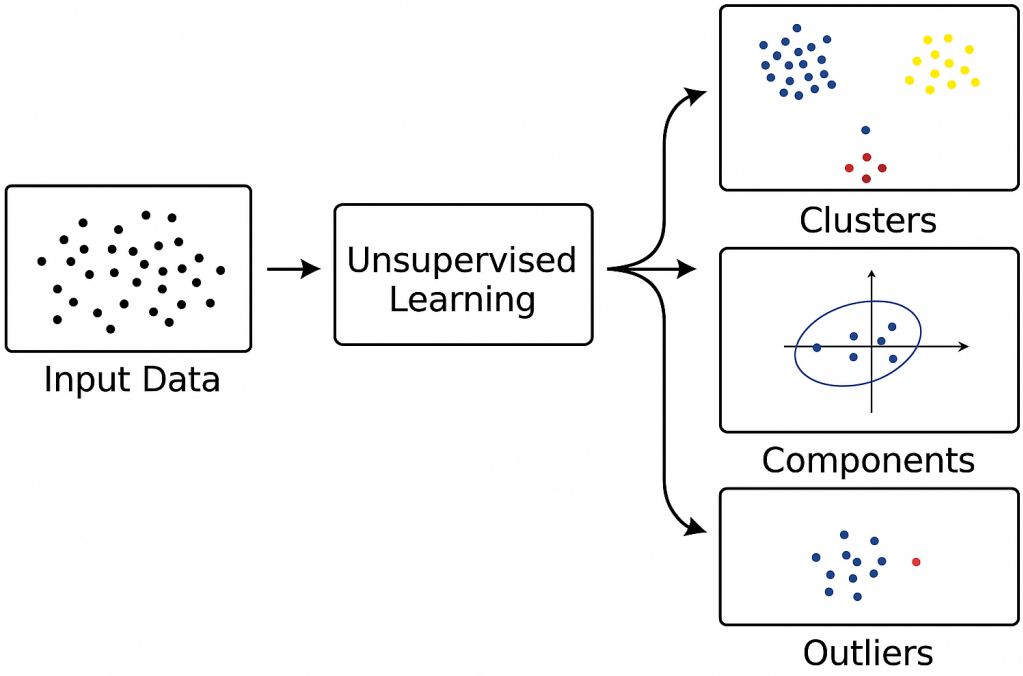



**Fig. 5.3.:** General framework of unsupervised learning

while nonlinear techniques like t-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) aim to preserve local neighborhoods or global topological structures in the projected space. These techniques are widely used for visualization, noise filtering, and feature compression in domains with multivariate measurements [92, 93].

Density estimation methods model the underlying probability distribution of the data. Approaches such as Kernel Density Estimation (KDE) or GMMs are particularly useful for identifying regions of high probability mass and for detecting outliers—samples that deviate significantly from the expected distribution. Such capabilities are crucial for early warning systems, anomaly detection, and probabilistic decision-making.

Recent advances in neural networks have introduced powerful unsupervised architectures such as autoencoders, which learn compressed latent representations by reconstructing inputs from a lower-dimensional encoding. Variants such as sparse autoencoders, denoising autoencoders, and Variational Autoencoders (VAEs) further enhance representation learning by introducing regularization terms or probabilistic assumptions over the latent space [94, 95]. These models have proven effective for tasks ranging from feature extraction and anomaly detection to generative modeling.

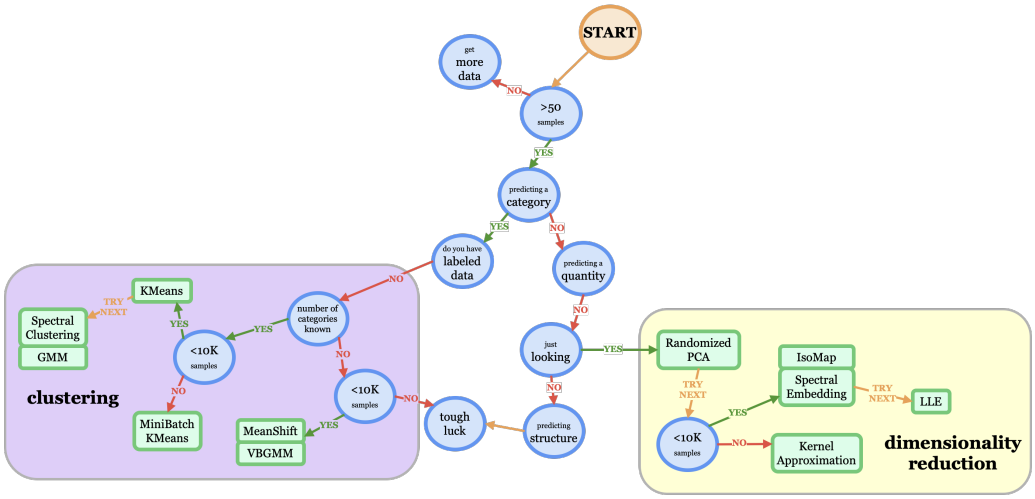



**Fig. 5.4.:** Popular algorithms used in unsupervised learning categorized by type

A particularly impactful direction in unsupervised learning is the emergence of self-supervised learning (SSL), which bridges the gap between supervised and unsupervised paradigms. In SSL, pretext tasks are designed to generate supervisory signals from the data itself, such as predicting missing parts of input, solving contrastive tasks, or reordering sequences. Methods like SimCLR, MoCo, and BYOL have demonstrated strong performance on representation learning benchmarks by leveraging contrastive objectives to maximize similarity between differently augmented views of the same instance while minimizing similarity across instances [96, 97, 98].

Figure 5.4 categorizes prominent unsupervised learning methods based on their functional characteristics. Clustering, dimensionality reduction, density modeling, and representation learning serve as the foundational pillars of this paradigm.

The application of unsupervised learning in real-world systems offers several advantages. It enables the discovery of intrinsic data structure without the need for annotated datasets, allows for more exploratory and adaptive analytics, and can serve as a pre-processing step to improve the performance of downstream supervised or reinforcement learning models. It is particularly advantageous in high-dimensional data regimes, where redundant or irrelevant features may obscure meaningful patterns.

Nonetheless, unsupervised learning also presents significant challenges. The absence of ground truth makes it difficult to evaluate and compare model performance quantitatively. Model selection, hyperparameter tuning, and interpretability often

rely on heuristic criteria or domain expertise. Furthermore, many unsupervised algorithms are sensitive to initialization, metric choice, or scale normalization, which can significantly influence their output [99].

As the volume and complexity of unlabeled data continue to grow, unsupervised learning plays an increasingly critical role in building scalable, autonomous, and intelligent systems. In fields such as cybersecurity, telecommunications, and bioinformatics, it supports applications including traffic segmentation, anomaly detection, behavior profiling, and latent feature discovery. With ongoing advancements in deep representation learning and scalable optimization techniques, unsupervised learning is poised to remain a cornerstone of data-driven intelligence.

### 5.1.3 Reinforcement Learning

Reinforcement Learning (RL) is a foundational ML paradigm aimed at solving sequential decision-making problems where the optimal strategy must be learned through interaction with a dynamic environment. Unlike supervised learning, which relies on labeled datasets, or unsupervised learning, which infers structure from unannotated data, RL operates through an agent–environment feedback loop. At each time step, the agent observes the current state of the environment, selects an action, transitions to a new state, and receives a scalar reward signal. The core objective is to learn a policy that maximizes the cumulative reward over time, often under uncertainty or partial observability.

Mathematically, RL problems are typically modeled using a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s'|s, a)$ represents the state transition probabilities, $R(s, a)$ is the reward function, and $\gamma \in [0, 1]$ is a discount factor that prioritizes immediate or future rewards. The goal is to identify an optimal policy $\pi^*(a|s)$ that maximizes the expected discounted return over trajectories induced by agent-environment interactions.

Figure 5.5 provides a conceptual illustration of the RL loop, where the agent iteratively improves its behavior based on observed rewards and updated value estimates.

Various algorithmic strategies have been developed to solve MDPs, with the principal families being value-based, policy-based, and actor-critic methods.

Value-based methods, such as Q-learning and SARSA, operate by learning an estimate of the action-value function $Q(s, a)$, which encodes the expected return

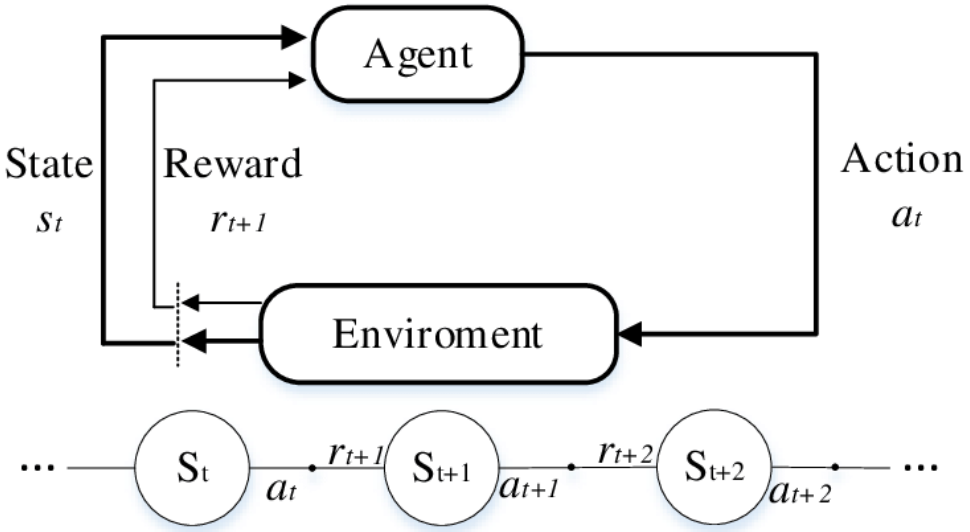



**Fig. 5.5.:** Standard reinforcement learning loop: an agent interacts with an environment to maximize cumulative rewards.

of performing action $a$ in state $s$ and subsequently following the optimal policy. Q-learning, an off-policy algorithm, iteratively updates $Q(s, a)$ using the Bellman optimality equation. These methods are conceptually simple and sample-efficient in small or tabular environments. However, they struggle with generalization and scalability in high-dimensional or continuous spaces, as value function estimation becomes unstable or intractable.

Policy-based methods, on the other hand, directly optimize the policy $\pi_\theta(a|s)$ via gradient ascent on the expected return. Algorithms such as REINFORCE use Monte Carlo estimates of the policy gradient but suffer from high variance, especially in sparse-reward or long-horizon tasks. Despite this, policy-gradient methods are naturally suited for environments with continuous action spaces and allow for stochastic policies, which can aid exploration and robustness.

Actor-critic algorithms combine the strengths of both approaches by maintaining two separate models: an actor that proposes actions according to a policy, and a critic that evaluates the action using a learned value function. This hybrid strategy reduces the variance of gradient estimates and stabilizes training. Advantage Actor-Critic (A2C) introduces the advantage function $A(s, a) = Q(s, a) - V(s)$ to quantify the relative value of an action compared to the average expected return from state $s$. Asynchronous variants such as A3C leverage parallel environments to accelerate learning and improve exploration diversity.

The advent of deep neural networks has given rise to Deep Reinforcement Learning (DRL), where neural architectures are used to approximate value functions, policies, or both. One of the seminal contributions in this domain is the Deep Q-Network (DQN) [100], which demonstrated that convolutional neural networks (CNNs) could be trained end-to-end to perform control from high-dimensional visual input. DQN introduced key stabilization techniques, including experience replay and target networks, that remain central to DRL algorithms today.

Policy-optimization algorithms have also evolved to address stability and efficiency concerns. Proximal Policy Optimization (PPO) [101] is a widely adopted DRL algorithm that constrains policy updates using a clipped surrogate objective. This prevents large, destabilizing updates to the policy parameters and enables stable training across a wide range of environments. PPO strikes a balance between sample efficiency and implementation simplicity, making it the algorithm of choice in many applied DRL scenarios, including robotics [102], recommendation systems [103], and intelligent traffic control [104].

Another important axis of RL research is the distinction between single-agent and multi-agent reinforcement learning (MARL). Traditional RL assumes a single agent interacting with a stationary environment. This formulation is well-understood and supported by convergence guarantees under specific conditions. However, many practical systems involve multiple agents acting concurrently, either collaboratively or competitively. In MARL settings, each agent optimizes its policy based on local observations and rewards, while the environment evolves due to the actions of all agents, thus violating the stationarity assumption.

To cope with the non-stationarity and coordination challenges of MARL, several algorithmic frameworks have been proposed. Centralized Training with Decentralized Execution (CTDE) offers a paradigm where agents are trained with global information (e.g., joint states or actions) but operate independently during deployment. Algorithms such as Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [105] extend deterministic policy gradient methods to multi-agent systems by incorporating opponent modeling and shared critics. Similarly, QMIX [106] employs a monotonic mixing network to combine individual agent value functions into a global value function, allowing for tractable learning in cooperative scenarios.

Figure 5.6 depicts the architecture of a typical MARL system, emphasizing the need for communication, synchronization, and shared objectives across agents.

Despite its demonstrated potential, RL remains a challenging methodology to apply in real-world settings. One of the main limitations is the high sample complexity:

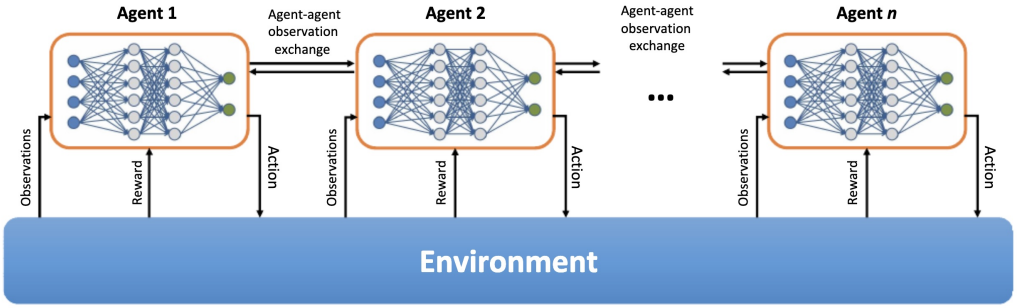



**Fig. 5.6.:** MARL diagram where agents exchange observations and learn coordinated policies to maximize long-term rewards.

learning optimal policies often requires millions of interactions with the environment, which is impractical or unsafe in many critical domains. Additionally, RL algorithms are sensitive to hyperparameters, reward shaping, and exploration strategies. Training instabilities, policy oscillations, and catastrophic forgetting are common pitfalls, particularly in sparse-reward or partially observable environments.

To address these limitations, a number of emerging research directions are being explored. Safe RL focuses on ensuring that learned policies respect predefined safety constraints during training and deployment. Offline RL leverages historical interaction data to learn policies without direct environment interaction, while sim-to-real transfer techniques attempt to bridge the reality gap between simulated and physical environments. Moreover, the use of high-fidelity digital twins and emulators facilitates controlled training and evaluation of RL models in complex systems.

Reinforcement learning offers a principled framework for autonomous control, planning, and decision-making under uncertainty. As advances in model architectures, training protocols, and safety mechanisms continue to evolve, RL is becoming increasingly viable for real-time, adaptive systems in domains such as telecommunications, robotics, finance, and beyond.

### 5.1.4 Integrating Machine Learning Approaches for Network Intelligence

This section has presented a comprehensive exposition of the fundamental paradigms underpinning machine learning, namely supervised learning, unsupervised learning,

and reinforcement learning. Each of these categories addresses distinct problem formulations and leverages specific algorithmic frameworks to extract knowledge from data. Supervised learning excels in scenarios where labeled datasets are available, enabling predictive models to approximate complex input-output relationships. Unsupervised learning, conversely, focuses on discovering latent structures and patterns in unlabeled data, playing a critical role in dimensionality reduction, clustering, and anomaly detection. Reinforcement learning represents a third paradigm wherein agents interact with dynamic environments to learn optimal policies through delayed reward signals, supporting sequential decision-making under uncertainty.

Across all paradigms, advances in algorithmic strategies—ranging from decision trees and support vector machines to deep neural networks and policy gradient methods—have significantly enhanced the representational power and generalization capabilities of learning systems. The rise of deep learning has further enabled the integration of machine learning into high-dimensional and temporally complex domains, while self-supervised and hybrid approaches have begun to bridge the gap between purely supervised and unsupervised learning regimes. However, challenges persist in terms of data efficiency, interpretability, robustness, and scalability, especially in operational contexts where real-time decision-making and limited ground truth are common.

As such, the development and deployment of machine learning systems require careful consideration of the underlying assumptions, computational constraints, and domain-specific requirements. Methodologies such as transfer learning, active learning, and continual learning are increasingly relevant for adapting models to evolving environments with minimal supervision. Additionally, the use of model-agnostic interpretability techniques and uncertainty estimation has become essential for ensuring trustworthy and accountable AI systems in critical infrastructures.

The exponential growth in bandwidth demand, the densification of network architectures, and the transition toward open and disaggregated optical infrastructures have significantly increased the operational complexity of transport networks. In this evolving landscape, the limitations of traditional algorithmic and rule-based management approaches are becoming increasingly apparent. These legacy techniques often fail to scale with the dynamic and multi-dimensional characteristics of modern optical systems, where decisions must be taken in real-time under uncertainty and across heterogeneous layers and technologies.

ML offers a compelling approach to addressing these challenges by enabling data-driven, adaptive, and predictive control in optical networks. Leveraging historical

data and real-time telemetry, ML algorithms can uncover latent patterns, model complex nonlinear relationships, and support high-dimensional decision-making with minimal human intervention. These capabilities are especially advantageous in optical environments, where physical-layer impairments, constrained resources, and heterogeneous topologies demand context-aware and fine-grained optimization strategies.

The application of ML to optical networks spans multiple functional domains and architectural layers. At the physical layer, ML is used to predict signal quality and diagnose impairments based on channel characteristics and hardware parameters. At the service and control layers, ML supports functions such as traffic forecasting, dynamic resource allocation, failure detection, and routing optimization. Techniques from supervised, unsupervised, and reinforcement learning are each applied depending on the availability of labels, the nature of the problem, and the operational constraints of the network.

The next section presents a comprehensive and in-depth analysis of how Machine Learning is being leveraged to enhance optical network control and management. It begins by reviewing the principal ML models and methodologies applied across the optical stack, identifying the core problem formulations and associated learning paradigms.

## 5.2 Taxonomy of ML Applications Across Optical Network Layers

ML has emerged as a critical technology for advancing the intelligence and autonomy of optical transport networks. The growing complexity of multi-layer, dynamic, and open optical infrastructures—combined with the increasing heterogeneity of transponders, modulation formats, and transmission channels—poses significant challenges for traditional rule-based or heuristic control schemes. In contrast to static models or vendor-specific algorithms, ML offers a data-driven approach capable of learning from the network's own operational telemetry, enabling adaptive, real-time optimization and management. This section provides a structured and technical overview of the principal use cases of ML in optical networks, focusing on both physical-layer and network-layer applications. Each task is described in terms of its operational context, traditional solutions, and recent ML-driven advances.

### 5.2.1 ML Applications at the Physical Layer

The physical layer of optical networks encompasses the transmission system, including optical transponders, amplifiers, and links. ML has shown strong potential in enhancing the visibility, control, and performance of these components.

#### Quality of Transmission Estimation

**QoT Estimation** is a critical task in the operation of transparent optical networks, as it enables the assessment of whether a LightPath (LP) can satisfy the physical-layer performance requirements—typically expressed in terms of Bit Error Rate (BER), Optical Signal-to-Noise Ratio (OSNR), or Q-factor—without the need for intermediate regeneration. These QoT metrics are inherently influenced by a multitude of transmission parameters, including but not limited to the modulation format, baud rate, coding rate, channel spacing, and the total propagation length of the LP. The highly nonlinear interplay among these factors renders the manual configuration and validation of LPs impractical in large-scale or dynamic networks.

Conventional QoT estimation methodologies may be broadly categorized into two principal classes. The first class leverages analytical modeling approaches, which utilize vendor-specified static parameters of optical components—such as amplifier gain, noise figure, fiber attenuation, and connector losses—to analytically approximate the QoT of a candidate LP. Although these models, such as the Gaussian Noise (GN) model, offer initial deployment insights, their accuracy is often compromised over time due to device aging, unmodeled impairments, or deployment deviations in disaggregated network environments. The second class of methods incorporates real-time telemetry collected from the operational network, such as signal power, OSNR, and noise figures measured via Erbium-Doped Fiber Amplifiers (EDFAs) and optical channel monitors. These telemetry-driven approaches offer greater adaptability to dynamic network conditions; however, they remain susceptible to uncertainties introduced by varying spectral load conditions and cross-channel interference, especially in elastic or open optical systems.

ML techniques offer a robust and scalable alternative, enabling the development of data-driven models that capture complex, nonlinear relationships between observable network features and QoT indicators. These features typically include topological attributes (e.g., hop count, path length), physical-layer parameters (e.g., baud rate, launch power, modulation format), and traffic descriptors (e.g., spectral

occupancy, co-propagating channel characteristics). Unlike traditional analytical approaches, ML-based QoT estimators do not rely on rigid vendor-specific assumptions, making them particularly well-suited for deployment in open and disaggregated optical transport networks.

A wide array of ML algorithms has been explored for this task. Support Vector Machines (SVMs) have been successfully employed for binary QoT feasibility classification, demonstrating high predictive accuracy across different network conditions [107]. Ensemble learning methods such as Random Forests provide improved generalization and robustness by aggregating multiple decision trees trained on distinct feature subsets [108]. Probabilistic models, notably Gaussian Process Regression (GPR), offer the added advantage of predictive uncertainty quantification, facilitating confidence-aware decision-making in QoT estimation [109]. Deep learning approaches, including fully-connected Deep Neural Networks (DNNs), have further advanced the field by capturing complex hierarchical relationships among input features, achieving superior performance when trained on sufficiently large and diverse datasets [110].

In addition to these techniques, Case-Based Reasoning (CBR) frameworks have been proposed. These models infer the QoT of a new LP by retrieving the most similar historical LP instance—based on normalized Euclidean distance across feature vectors—and extrapolating its QoT outcome. This form of cognitive modeling allows for interpretable and adaptive decision-making and has been particularly effective in early ML-driven network control architectures.

Several comparative studies have examined the efficacy of different ML classifiers for QoT-E. Results indicate that while SVMs may yield the highest classification accuracy, they often incur higher computational complexity compared to simpler models like Random Forests or K-Nearest Neighbors (KNN). Moreover, GPR has been shown to generalize well across heterogeneous optical network configurations, with input features including signal power, symbol rate, channel spacing, and link length.

The applicability of ML-based QoT estimators has been validated not only in simulation studies but also through integration into experimental testbeds and field trials. These real-world validations highlight their capacity to enable fully automated QoT prediction, thereby facilitating agile LP provisioning, reducing operational costs, and improving network reliability in highly dynamic environments.

In summary, the integration of ML into QoT estimation frameworks represents a significant step forward in achieving intelligent, autonomous optical networking. The ability of ML models to generalize across heterogeneous infrastructures, adapt

to real-time conditions, and support high-frequency decision-making underscores their essential role in the evolution of open and disaggregated optical networks.

### Optical Amplifier Control

**Optical Amplifier Control** is a fundamental task in the management of dynamic optical transport systems. As internet traffic patterns grow increasingly heterogeneous and elastic—driven by fluctuations in service demands and the proliferation of cloud applications—the ability to dynamically configure and reconfigure LPs becomes essential. Such reconfigurations, however, directly affect the power levels within optical spans, particularly in systems comprising multiple cascaded EDFAs. Since EDFAs amplify signals based on the total spectral load they experience, the activation or deactivation of LPs often introduces power transients, which can degrade signal quality, lead to bit errors, or in extreme cases, cause complete LP failure.

Traditionally, EDFA settings such as gain, tilt, and output power are either statically preconfigured based on worst-case assumptions or adjusted via slow-acting control loops. These conventional methods lack the agility required in modern reconfigurable optical networks, especially those supporting Colorless, Directionless, and Contentionless (CDC) architectures and Wavelength Selective Switches (WSS). To overcome these limitations, recent research has focused on leveraging ML models for real-time amplifier control, enabling more accurate and adaptive configuration of EDFA operating points.

ML-based optical amplifier control techniques aim to model and predict the EDFA behavior as a function of input signal characteristics and network state variables. Several cognitive approaches have been proposed. For instance, the work in [111], introduced a cognitive gain control scheme based on CBR, wherein a historical database of EDFA gain settings, path lengths, and OSNR readings is used to retrieve and adapt prior amplifier profiles for new LP requests. This method enables fast configuration by selecting and perturbing similar known cases, with the updated OSNR values estimated and stored for future reuse.

In parallel, data-driven approaches using neural networks have demonstrated promising results in capturing the nonlinear and multivariate dependencies that characterize amplifier operation. [112] proposed an Artificial Neural Network (ANN)-based controller trained on experimentally collected EDFA response data. The model was capable of predicting appropriate gain settings with interpolation errors below 0.5 dB, achieving fast adaptation to load changes without the need for exhaustive look-up tables or slow convergence.

Further experimental validation has been carried out in [113], who demonstrated a hybrid control architecture combining ML-based estimators with feedback-based error correction loops. Their testbed implementation showed significant reduction in power fluctuations during LP activation and deactivation events in multi-span transmission systems, underscoring the practical viability of ML-enhanced amplifier control.

More broadly, these ML-based strategies differ in their modeling granularity. Regression models—linear or nonlinear—are used to emulate the EDFA transfer function, predicting output power or gain tilt given a set of input signals. Ensemble methods, such as gradient boosting or random forests, have also been explored for their robustness in generalizing to unseen load configurations.

The integration of intelligent amplifier control mechanisms into Software-Defined Networking (SDN) environments further expands their utility. By exposing EDFA control interfaces to centralized controllers, ML models can be used not only for local optimization but also for network-wide power equalization and impairment-aware routing.

In summary, ML-driven EDFA control represents a paradigm shift from reactive, hardware-specific tuning to proactive, context-aware configuration. These approaches significantly enhance the stability of optical channels, mitigate signal degradation due to dynamic reconfigurations, and reduce operational complexity—key requirements for the next generation of fully automated and disaggregated optical networks.

### Optical Performance Monitoring

**Optical Performance Monitoring (OPM)** is a fundamental task in modern optical transport systems, enabling real-time tracking of transmission quality and early detection of physical-layer impairments. It provides the foundation for numerous adaptive network functions such as dynamic signal power equalization, modulation format adjustment, spectrum reallocation, and fault localization. Within SDN frameworks, OPM plays a pivotal role by continuously feeding controllers with accurate physical-layer state information.

Traditionally, OPM relied on dedicated monitoring hardware devices—such as OSNR meters, dispersion analyzers, or BER testers—deployed at strategic network points to individually assess transmission impairments such as chromatic dispersion, polarization mode dispersion, polarization-dependent loss, and nonlinear distortions. These

tools, however, suffer from several limitations: they are expensive, difficult to deploy at scale, require careful calibration, and often provide limited temporal or spatial granularity. Moreover, the independent measurement of each impairment type is analytically challenging due to the nonlinear interaction of signal degradations across optical spans.

To address these challenges, recent research has proposed leveraging ML techniques for generalized, data-driven OPM. The key insight is that complex signal degradations manifest as observable patterns in sampled waveform data—such as eye diagrams, constellation plots, or power histograms—which can be used as feature inputs to cognitive engines that learn to infer the underlying physical impairments. This approach enables scalable, cost-effective, and multipurpose monitoring using fewer physical probes.

Most ML-based OPM systems rely on supervised learning models, trained to estimate specific transmission parameters (e.g., OSNR, chromatic dispersion, BER) or classify signal quality levels. [114] and [110] reviewed several architectures where features extracted from asynchronous eye diagrams (e.g., eye opening, jitter, crossing amplitude) or from sampled amplitude histograms are fed into neural networks, SVMs, or decision trees for impairment identification. These methods reduce the need for intrusive taps or expensive instrumentation, while maintaining high accuracy in realistic network conditions.

Deep learning models have also been proposed to remove the feature engineering step by learning directly from raw waveform data. Convolutional Neural Networks (CNNs) and autoencoders have been trained on time-domain signals or digitized eye diagrams to simultaneously predict multiple impairments or detect anomalies. Although this approach increases computational complexity and demands larger datasets, it offers the advantage of model generalization and end-to-end learning.

Unsupervised learning techniques, including clustering and Self-Organizing Maps (SOMs), have been applied for anomaly detection and impairment grouping without requiring labeled datasets. These models are particularly valuable in open and disaggregated optical networks, where device behavior is less predictable, and labeled impairment data is difficult to obtain.

The growing interest in ML-based OPM also aligns with the integration of coherent detection and Digital Signal Processing (DSP) in optical transponders, which facilitates access to rich signal statistics in real-time. When combined with in-line telemetry and network-wide data aggregation, ML models can offer centralized or distributed performance monitoring with sub-second latency.

In summary, ML-enhanced OPM systems present a scalable and hardware-agnostic alternative to conventional techniques. They enable accurate monitoring of multiple physical-layer impairments using low-cost measurements and data-driven inference, unlocking new capabilities for proactive control, fault diagnosis, and autonomous network operation in evolving optical infrastructures.

### 5.2.2 ML Applications at the Network Layer

At the network layer, ML is leveraged to address dynamic resource management, failure mitigation, and service assurance tasks. These problems are typically combinatorial and non-convex, and ML provides effective heuristics learned from data.

#### Traffic and Optical Resource Prediction

Traffic prediction is a fundamental enabler for proactive provisioning, virtual topology reconfiguration, and dynamic bandwidth allocation in optical networks. Accurate forecasting of traffic demand allows network operators to adapt infrastructure resources efficiently, reducing over-provisioning while ensuring Service-Level Agreement (SLA) compliance. ML models are particularly well-suited for traffic prediction tasks due to their ability to learn complex temporal and spatial correlations from historical data.

Traditional time-series forecasting techniques such as Auto-Regressive Integrated Moving Average (ARIMA) and Kalman filters have been applied for traffic prediction during the planning phase of Virtual Network Topologies (VNTs) [115, 116]. However, their applicability is constrained by their inherent linear assumptions and limited adaptability to highly non-stationary traffic trends observed in real-world optical networks. In contrast, supervised ML models such as Feedforward Neural Networks (FNNs), Support Vector Regression (SVR), and more recently deep learning models, including Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have demonstrated superior performance for learning non-linear and long-range temporal dependencies in traffic matrices [117, 118].

In particular, LSTM networks are widely used due to their memory cell architecture, which allows capturing both short- and long-term trends in source-destination traffic demand. These models support predictive reconfiguration strategies by feeding forecasted traffic into decision-making modules that trigger VNT adaptation and resource reassignment. Some studies extend the use of deep learning techniques to CNNs and attention mechanisms for spatiotemporal traffic learning [119].

Beyond node-to-node traffic demand, optical resource prediction focuses on estimating the state of the spectrum within the network. This includes predicting spectrum utilization levels, available contiguous slot ranges, and fragmentation metrics, which are critical for efficient Routing, Modulation and Spectrum Assignment (RMSA) [120, 121]. Since the spectrum resource in Elastic Optical Networks (EONs) is fine-granular and dynamically allocated, real-time prediction of spectrum state aids in reducing fragmentation and blocking probabilities.

ML-based models for optical resource prediction typically leverage features such as spectrum occupancy histograms, path history, modulation formats, and demand characteristics. For instance, in [122], random forests and gradient boosting machines are trained to estimate spectrum availability for candidate paths, helping accelerate the decision-making in dynamic RMSA. Another study [123] proposes a DRL agent that learns to predict future spectrum states and proactively reserves contiguous slots for upcoming demands.

Clustering and matrix factorization techniques have also been applied to analyze historical traffic matrices and estimate likely spectrum usage across the network. Collective non-Negative Matrix Factorization (C-NMF), for example, allows decomposing multiple traffic snapshots into shared basis components that characterize underlying spectrum usage patterns [124].

Furthermore, optical resource prediction has been extended to support spectrum defragmentation strategies. Predictive models anticipate when fragmentation is likely to impair resource allocation efficiency and trigger re-optimization routines accordingly. In [125], supervised learning is applied to detect high fragmentation scenarios based on real-time spectrum allocation data.

Overall, ML-driven traffic and optical resource prediction represents a pivotal component in the intelligent control loop of modern optical networks. These methods support efficient planning, allocation, and adaptation of spectral resources, enhancing overall network utilization and resilience to traffic volatility.

### Failure Prediction and Localization

Failure prediction and localization play a pivotal role in ensuring the robustness and availability of optical networks. As optical infrastructures grow in scale and complexity, network operators are increasingly adopting ML techniques to anticipate and mitigate potential disruptions. Predictive models enable timely fault identification,

reduce Mean Time To Repair (MTTR), and support dynamic reconfiguration, thus maintaining high Quality of Service (QoS) and meeting SLAs.

ML-based approaches to failure prediction typically rely on large volumes of telemetry data, such as BER, received signal power, amplifier current, and component temperature. These inputs are used to detect abnormal conditions or performance degradations indicative of incipient failures. Classification models such as SVMs, decision trees, and Bayesian classifiers have been successfully deployed for identifying early signs of degradation and categorizing different fault types [126, 127].

One notable example is the FEELING framework [128], which utilizes a combination of SVMs and decision trees to diagnose failures based on spectral power readings and BER signatures. Similarly, the TISSUE system [128] analyzes the slopes of BER deviations across network nodes, comparing observed trends to theoretical expectations. If deviations surpass defined thresholds, a failure alert is triggered. These systems operate in near real-time, facilitating proactive rerouting and resource reallocation.

Advanced techniques integrate probabilistic reasoning to address uncertainty in the presence of incomplete data. Bayesian networks, often combined with Expectation-Maximization (EM) algorithms, are employed to estimate failure probabilities when some telemetry measurements are unavailable [127]. These methods have been demonstrated to yield accurate fault localization across multi-layer optical topologies.

Regression-based approaches have also gained traction for continuous fault prediction. For instance, neural network-based regression models are trained to predict deviations in BER, signal gain, or power readings as functions of environmental variables or equipment aging. In [129], a statistical neural regression model was applied to predict failure events using features such as amplifier current and module temperature, achieving high predictive accuracy across various operating conditions.

Ensemble learning methods further enhance fault classification robustness. Random forests and gradient boosting machines leverage multiple decision trees to model diverse failure modes and improve generalization on unseen fault scenarios [127]. These models are particularly effective in handling heterogeneous datasets with high-dimensional features.

In addition, unsupervised learning methods such as clustering and autoencoders are used to detect novel or previously unseen failure patterns. Deep autoencoders can

reconstruct healthy signal profiles and identify anomalies through reconstruction error thresholds. Such models are particularly valuable in disaggregated environments where component behavior may deviate from vendor-defined baselines [127].

Finally, spatial statistical models like Kriging have been adapted for fault localization, leveraging historical failure data to estimate the likelihood of failure at different network segments. By modeling spatial correlations, Kriging-based predictors can pinpoint failure origins even in sparse monitoring scenarios [127].

Collectively, these ML-driven techniques enable a paradigm shift from reactive to proactive fault management in optical networks. By continuously learning from operational data and adapting to evolving network conditions, ML models improve resilience, reduce downtime, and support autonomous optical network control.

### Routing Modulation and Spectrum Assignment

The RMSA problem represents one of the most computationally intensive and strategically significant tasks in the operation of elastic optical networks (EONs). The goal of RMSA is to jointly determine the optimal path through the network and allocate the appropriate spectrum resources (i.e., frequency slots) to lightpath requests, while minimizing the blocking probability and avoiding spectrum fragmentation. As optical networks evolve to support higher bandwidth demands and dynamic service provisioning, traditional routing algorithms are no longer sufficient to ensure efficient and flexible spectrum utilization.

The RMSA problem is known to be NP-hard due to the combinatorial nature of its constraints. Specifically, any valid solution must satisfy the following three key spectrum constraints:

- **Spectrum continuity:** The same set of frequency slots must be available along the entire route.
- **Spectrum contiguity:** Allocated slots for a lightpath must be adjacent in the spectrum.
- **Non-overlapping:** Allocated slots must not conflict with those of existing lightpaths.

These constraints are depicted in Figure 5.7, where a connection request is evaluated against the current spectrum occupancy at two distinct time instances. At time T1, sufficient contiguous and continuous spectrum is available along the path between

nodes N1 and N4, making the request feasible. In contrast, at time T2, fragmentation and existing allocations render the request unsatisfiable.

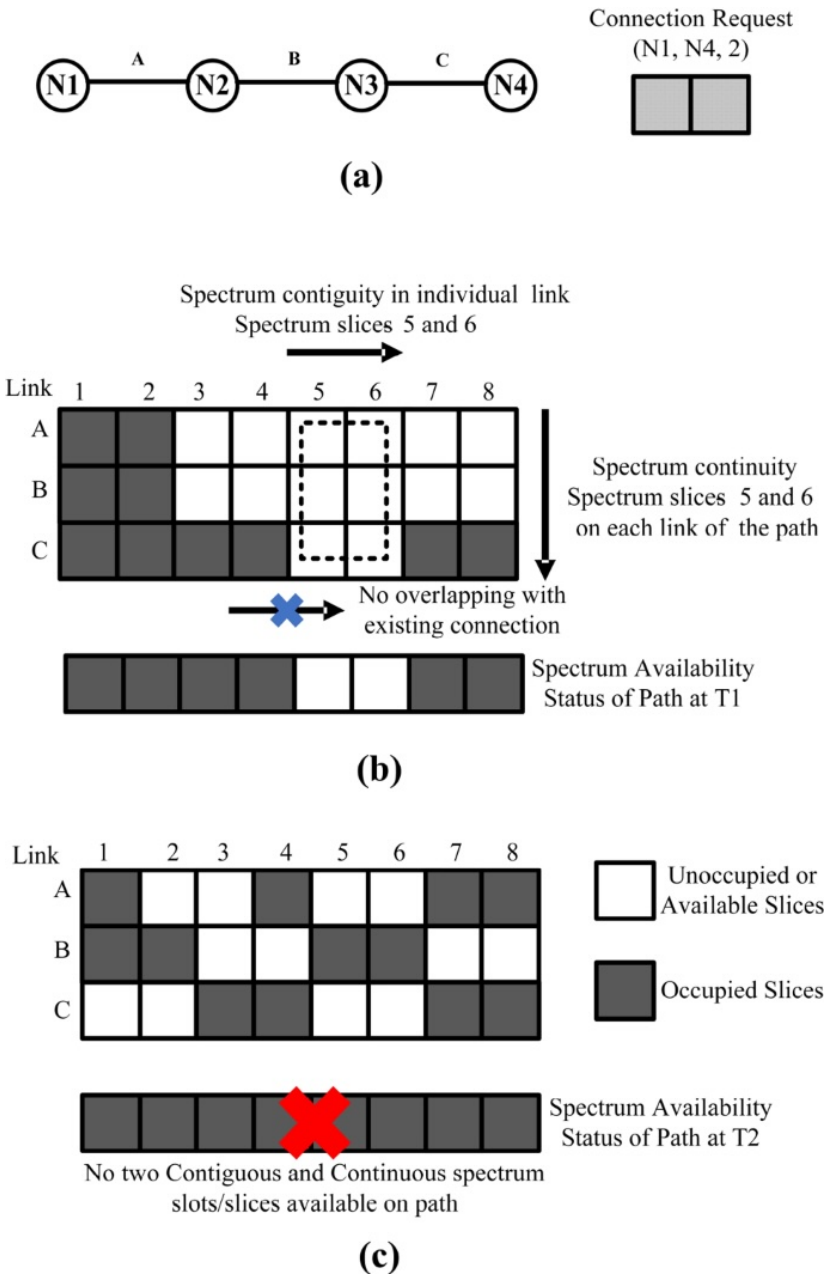



**Fig. 5.7.:** RMSA constraint illustration. (a) A connection request from node N1 to node N4 requiring two contiguous FS; (b) Spectrum allocation scenario at time T1, where the selected path satisfies the RMSA constraints, including spectrum continuity, contiguity, and non-overlapping allocation; (c) Spectrum state at time T2, where no available contiguous and continuous spectrum slices exist along the path to accommodate the connection request. **Figure reproduced from** [130].

Conventional solutions to RMSA, such as heuristic approaches and Integer Linear Programming (ILP), struggle to scale with network size and dynamicity. Heuristics often rely on pre-defined cost functions or static rules, which may lead to suboptimal spectrum allocation, especially in highly variable traffic scenarios. ILP-based approaches, while optimal for small topologies, suffer from prohibitive computational complexity in large-scale networks or real-time environments.

To address these limitations, ML techniques have been widely adopted. Supervised learning models, including Decision Trees, SVMs, and Deep Neural Networks (DNNs), are trained on historical RMSA instances to predict feasible routes and slot assignments. These models reduce computation time at runtime by replacing exhaustive search with inference from trained policies [131]. Unsupervised learning techniques such as clustering and NMF have been used to extract traffic patterns and spectrum usage trends, aiding in proactive spectrum allocation decisions [132, 133].

However, RL has emerged as one of the most promising frameworks for solving the RMSA problem in dynamic optical networks. By formulating RMSA as a Markov Decision Process (MDP), RL agents are trained to learn optimal routing and spectrum allocation policies by interacting with the environment. The state space typically encodes network topology, spectrum usage, and fragmentation levels, while actions correspond to selecting paths and spectrum blocks.

In particular, DRL techniques such as Deep Q-Network (DQN) [134], Proximal Policy Optimization (PPO) [101], and Advantage Actor-Critic (A2C) [135] have been successfully applied to learn RMSA policies in high-dimensional, continuous state-action spaces. These methods leverage deep neural networks to approximate Q-values or policy gradients, enabling generalization across a wide range of network scenarios.

Multi-Agent Reinforcement Learning (MARL) schemes further extend this paradigm by enabling distributed decision-making among autonomous agents. Each agent may be responsible for a subset of the network (e.g., specific nodes or domains) and cooperatively learns policies through shared experiences or limited message passing. MARL techniques have demonstrated improved scalability and robustness in decentralized environments [136].

ML and RL-based RMSA strategies are increasingly critical for managing spectrum resources efficiently in EONs. They enable real-time adaptability to traffic variability, reduce blocking and fragmentation, and facilitate scalable orchestration across disaggregated infrastructures. As optical networks become more dynamic and

heterogeneous, intelligent RMSA solutions will remain a cornerstone of efficient network operation and automated service provisioning.

## 5.3 Practical Challenges for Deploying Machine Learning in Optical Networks

While ML and DRL techniques have demonstrated significant potential for improving control and management in optical networks, their deployment in carrier-grade environments remains subject to several critical operational constraints. Beyond algorithmic performance, practical considerations such as computational overhead, scalability, integration with existing control planes, and robustness under real-world conditions play a decisive role in determining their applicability.

One of the primary challenges is the computational complexity associated with advanced ML models, particularly DRL agents. These models often rely on deep neural networks with large parameter spaces, requiring substantial computational resources for both training and inference. While training is typically performed offline using high-performance computing infrastructure, inference must often be executed within the control loop of the network. In carrier-grade environments, where decisions may need to be taken within strict time constraints, the latency introduced by model inference can become a limiting factor. For example, in dynamic resource allocation or path computation scenarios, delays of even a few seconds may lead to suboptimal decisions or service degradation. This is particularly critical when DRL agents are embedded in centralized controllers that already process large volumes of network state information.

In addition to latency, the scalability of ML-based solutions adds significant challenges. Optical networks at realistic scale involve thousands of nodes, links, and channels, each contributing to a high-dimensional state space. DRL algorithms, in particular, are known to struggle with scalability due to the exponential growth of the state-action space. Techniques such as state abstraction, dimensionality reduction, or hierarchical learning can mitigate this issue, but they often introduce trade-offs between model accuracy and computational efficiency. Moreover, the training of such models requires large datasets that accurately represent network conditions, which are not always readily available in operational environments.

Another important obstacle is the generalization capability of ML models when transitioning from simulated or laboratory environments to real-world optical net-

works. Many proposed solutions are validated using simplified network topologies or synthetic traffic patterns, which do not fully capture the complexity and variability of production networks. In practice, optical networks exhibit heterogeneous equipment, vendor-specific behaviors, and dynamic traffic patterns that may differ significantly from training data. This domain shift can lead to degraded performance or even incorrect decisions when models are deployed in live networks. Ensuring robust generalization therefore requires careful dataset design, continuous model validation, and mechanisms for online adaptation.

Integration with existing control and management systems also represents a non-trivial challenge. Carrier-grade optical networks rely on established protocols, standardized data models, and strict operational procedures to ensure reliability and interoperability. Embedding ML models into this ecosystem requires well-defined interfaces and compatibility with existing control loops. In particular, ML-driven decisions must be explainable and verifiable to meet operational requirements, especially in scenarios where incorrect actions could impact service-level agreements (SLAs). This raises additional challenges related to model interpretability and trustworthiness.

Reliability and fault tolerance are further critical considerations. Unlike traditional rule-based systems, ML models may exhibit unpredictable behavior when encountering unseen conditions. In a carrier-grade context, where availability requirements are extremely stringent, it is essential to incorporate safeguards such as fallback mechanisms, confidence thresholds, and hybrid control strategies combining ML-based and deterministic approaches. These mechanisms, however, add complexity to the overall system design and may limit the autonomy of ML-driven control loops.

Finally, operational deployment requires alignment with DevOps and MLOps practices. Managing the lifecycle of ML models—including data collection, training, validation, deployment, and monitoring—introduces additional overhead compared to traditional network functions. Continuous integration and deployment pipelines must be extended to handle model versioning, performance tracking, and retraining processes. This is particularly important in optical networks, where environmental conditions and traffic patterns evolve over time, necessitating periodic model updates to maintain performance.

In summary, while ML and DRL offer powerful tools for enhancing optical network control and management, their deployment in carrier-grade environments is constrained by challenges related to computational overhead, scalability, generalization, integration, and operational reliability. Addressing these challenges requires a holistic approach that combines algorithmic innovation with system-level

design considerations, paving the way for practical and robust ML-driven network automation.

These operational constraints provide essential context for understanding both the opportunities and the limitations of data-driven approaches in optical networks. The integration of ML into optical network control and management—as detailed through its applications in QoT estimation, optical amplifier control, performance monitoring, traffic forecasting, failure prediction, and RMSA—demonstrates the growing role of data-driven intelligence in enabling scalable, autonomous, and adaptive infrastructures. At the same time, the aforementioned challenges highlight the need for carefully designed models that balance accuracy, computational efficiency, and deployability in real-world environments.

These foundational concepts and technical challenges have directly motivated the research contributions developed throughout this PhD work. In particular, we investigate supervised learning for QoT estimation using Gaussian Process Regression with physical impairment awareness; deep reinforcement learning frameworks for solving the RMSA problem through both single-agent and multi-agent paradigms; and a novel Drift-Adaptive Ensemble-based QoT classifier capable of responding to non-stationary traffic and network dynamics. The following chapter presents a comprehensive account of these contributions, highlighting their design, implementation, and performance evaluation across synthetic and real-world optical network scenarios.

# 6 Contribution 2. Design and Evaluation of ML Models for Optical Network Control

This section presents the practical implementation and evaluation of ML techniques applied to the management and optimization of optical networks. Building upon the foundational concepts introduced in the previous chapters, we focus on the development, integration, and performance analysis of ML models tailored to key operational tasks within both the physical and network layers. These tasks include QoT estimation, optical transponder configuration, and dynamic resource orchestration via RL-based RMSA.

The section is structured around a series of research contributions conducted during the course of this Ph.D. Each contribution is presented in the form of an independent experiment or case study, corresponding to peer-reviewed publications. For each work, we detail the ML models employed, the dataset characteristics, training methodology, and performance metrics used for evaluation. Where applicable, results are derived from emulated testbeds, real network telemetry, or simulation environments that mimic the conditions of open and disaggregated optical networks.

By examining these contributions, we aim to provide a comprehensive assessment of how ML models can be effectively deployed to enhance network automation, adaptability, and fault tolerance. The results support the hypothesis that data-driven approaches not only reduce the operational complexity associated with manual configuration but also yield measurable gains in spectral efficiency, failure mitigation, and service availability.

## 6.1 Contribution 2.1. Impairment-aware Path Computation and ML-aided Transponder Configuration Optimization

In this work, we demonstrated an online impairment-aware path computation engine by leveraging a machine learning model to recommend the most appropriate path and optical parameters combination for a connectivity service request in a fully disaggregated multi-vendor optical transport network.

As transponders have a rapid innovation cycle, network operators consider the partially disaggregated network as a pragmatic approach, which decouples this equipment from the Open Line System (OLS) to eliminate the vendor lock-in issue. The choice of transponder providers is open to which could meet the service requirements (e.g., capacity, transmission range). However, it is still obliged to use vendor-proprietary solutions such as network planning tools, PCE, power optimization, etc., within the OLS; due to the complexity of optical transmission with physical impairments, the lack of standardization at layer 0.

Besides, the evolution towards multi-vendor fully disaggregated network architecture requires a vendor-agnostic mechanism to act as a path computation element, as well as estimate the QoT of an optical path traversing different network elements from various vendors. The Gaussian Noise Python (GNPy) has the potential to perform such multi-vendor optical network planning tasks. However, GNPy has some limitations:

- optical parameters (e.g., transceiver operational mode, transmit power, etc.) must be specified in the path request, which does not ensure an optimal QoT;
- the use of non-standardized data models (i.e., network topology, path computation, equipment configuration) restricts its association with other SDN applications. If the transmit power is not set properly, the computed path from GNPy may not have a good OSNR value at the receiver. Thus, this requires a power control loop optimization mechanism. In traditional single-vendor systems, each vendor has a proprietary mechanism to adjust the power. However, this proprietary solution is no longer applicable in the multi-vendor scenario.

In [137], a telemetry-based automated solution is proposed to enhance the operational mode selection of transponders in the partially disaggregated network. The study in [138] experiments with various ML models on the retrieved data from a running network to predict the QoT of an unestablished optical path in another

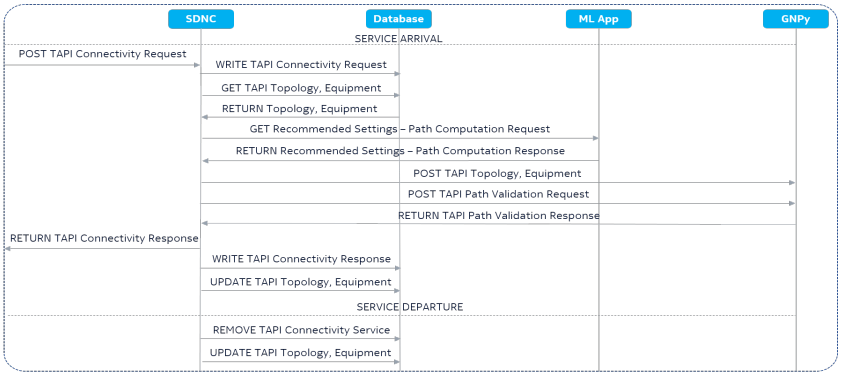



**Fig. 6.1.:** L0 Impairment-aware Path Computation Sequence Diagram

agnostic network. Nevertheless, the question of provisioning online layer 0 services with optimal settings in the multi-vendor fully disaggregated scenario has not been resolved.

For this reason, we propose a ML model to recommend the most favorable optical configurations and paths to reduce the time complexity in the multi-vendor power control loop optimization procedure. This ML model can cover all equipment parameters in the network topology thanks to the centralized SDN paradigm. We make use of the container-based microservice SDN control platform from 3 to demonstrate the capability to perform online impairment-aware layer 0 path computations.

The implementation of GNPy for path computation in optical networks relies on custom data models for network topology, equipment configuration, and path request processing. However, these proprietary models necessitate a translation layer to integrate with standardized NorthBound Interface (NBI) models within an SDN Controller (SDNC). The TransportAPI (T-API) serves as the most suitable NBI due to its extensive adoption and demonstrated interoperability across various use cases. Consequently, a mapping is performed between GNPy's data models and corresponding T-API interfaces, including T-API topology, T-API equipment, and T-API path computation. As T-API was not originally designed for impairment-aware networking, an extension is introduced to enhance its interface for this functionality. Later, this extension was adopted by the standard and introduced in a later release.

In a dynamic optical network, resources such as transponder types, central frequency, slot width, and Optical Multiplex Section (OMS) links must be dynamically allocated and released based on service demands. Additionally, operators must be able to reserve bandwidth in advance, securing network resources for upcoming service requests. The online L0 impairment-aware path computation procedure is described in Figure 6.1.


**(a)** Example of the dataset **(b)** ML model

**Fig. 6.2.:** Dataset and ML model

GNPy requires detailed optical parameters in the path request to perform path computation and validation. While user-defined parameters such as source, destination, and capacity are straightforward, other critical parameters—including transponder type, operational mode, and transmit power—are more complex due to the vast range of optical configurations and the network state's variability. A service is deemed feasible only if its estimated Generalized OSNR (GOSNR) surpasses the sum of the OSNR threshold and system margin at the receiver.

Traditionally, a conservative approach has been employed, where predefined optical parameters with high margins are used for all services sharing similar characteristics. This, however, results in extended optimization periods due to unnecessary monitoring adjustments and inefficient resource utilization. A brute-force search of all possible configurations is impractical given the extensive operational range of network elements and the real-time constraints of service provisioning.

To address these challenges, an ML-based application is developed to recommend optimal optical parameter settings and routing paths, leveraging real-time network states and service request information. The core of this application is an ML model that interacts with the SDNC via the T-API interface, facilitating autonomous and intelligent path selection.

The training data for this ML model is gathered from a multi-vendor optical network testbed. During each service request, optical parameters are fine-tuned, and corresponding monitored values are recorded. To enrich the dataset, randomized service requests are generated with varying optical parameter configurations, and GNPy is used to compute the GSNR values. The input feature set (X) comprises optical parameters and network state metrics such as spectrum occupancy, while the output label (Y) is the GOSNR value as shown in Figure 6.2a.

Finding an explicit function to model this relationship is highly complex. Instead, the ML model constructs a surrogate function to approximate the optimal parameter-to-GOSNR mapping, as shown in Figure 6.2b. Gaussian Process Regression (GPR) is

chosen as the foundational technique for modeling the complex relationship between input optical parameters and the resulting GOSNR values. GPR is a non-parametric, probabilistic model that defines a distribution over functions, making it particularly suited for scenarios where the underlying data relationships are highly nonlinear and uncertain. The key advantage of GPR is its ability to provide not only point predictions but also uncertainty estimates, allowing network operators to gauge the confidence in each prediction. Given a set of training data, GPR models the distribution of functions that could fit the data while maintaining a measure of confidence. The covariance function (or kernel) plays a crucial role in defining the smoothness and shape of the learned function. A Radial Basis Function (RBF) kernel is typically employed to capture fine-grained variations in optical transmission impairments.

Despite its advantages, GPR suffers from scalability issues when applied to large datasets due to its cubic complexity in the number of data points. To mitigate this, Bayesian Optimization (BO) is utilized as an enhancement mechanism. BO is an iterative optimization strategy that efficiently searches for the best hyperparameters or configurations by constructing a probabilistic surrogate model of the objective function. It employs an acquisition function to determine the next best sample point, balancing exploration (sampling in regions of high uncertainty) and exploitation (sampling in regions predicted to yield optimal performance).

The integration of BO into GPR fine-tunes the predictive model by adaptively selecting the most informative training samples. Initially, the model has a wide confidence interval, reflecting high uncertainty. As more data points are sampled and incorporated, BO systematically refines the surrogate function, converging towards an optimal representation of the system's behavior. This enables faster convergence in determining optimal optical configurations, significantly reducing the computational overhead compared to exhaustive search methods.

The ML model is implemented using TensorFlow and scikit-learn 0.23.2, enabling efficient training and inference. The surrogate function allows for rapid optical parameter selection and path computation, significantly reducing the time complexity associated with conventional trial-and-error methods. Additionally, power adjustments occur faster, leading to enhanced network performance. The ML-driven application can function independently of traditional network planning tools, with GNPy serving as a path validator to confirm the feasibility of recommended routes and configurations.

The demonstration focuses on the difference in the power adjustment of two layer 0 services. One uses the default optical parameter settings, and the other applies the recommended settings from the ML application. Details are specified as follows:

- Step 1: L0 T-API Connectivity Service Request - from the GUI, we show the T-API topology context, current network state as well as detailed optical parameters descriptions of equipment from the database. Two T-API connectivity service requests are generated with the same Source, Destination, and Capacity inputs. For other required fields, one uses the default optical parameters; likewise, the other invokes the ML application for a more appropriate suggestion.
- Step 2: ML-aided Optical Parameters and Paths Recommendation - Given the Source, Destination, Capacity, and current network state inputs, the ML application returns the most suitable optical parameters (such as the transponder type, operational mode, transmit power) and their corresponding path for the second request.
- Step 3: Service Provisioning - After composing two requests, SDNC sends to GNPy for path computation (request 1) and validation (request 2). If paths are validated, network elements are configured accordingly.
- Step 4: Power Control Loop Observation - SDNC performs power adjustment based on a simple centralized power optimization algorithm. We observe from the monitored data that (see Figure 6.3), with the recommendation from the ML application, the transmit power is close to the optimal value. On the other hand, the initial configuration is implemented with a high margin to avoid service disruption. Thus, SDNC monitors the transmission performance to regulate the transmit power to reach the optimal value with regards to OSNR and BER.

This work introduces an ML-assisted impairment-aware path computation framework for fully disaggregated optical networks, enabling intelligent selection of transponder parameters and routing paths based on real-time network states. By employing GPR with Bayesian BO, the system approximates the complex mapping between optical configurations and GOSNR, offering predictive capabilities with quantified uncertainty. This data-driven surrogate modeling approach significantly improves decision-making efficiency, reduces provisioning latency, and optimizes power control loops without resorting to exhaustive search or conservative overprovisioning.

The integration of the ML model with the standardized T-API northbound interface enables seamless interaction with SDN control platforms. This compatibility ensures

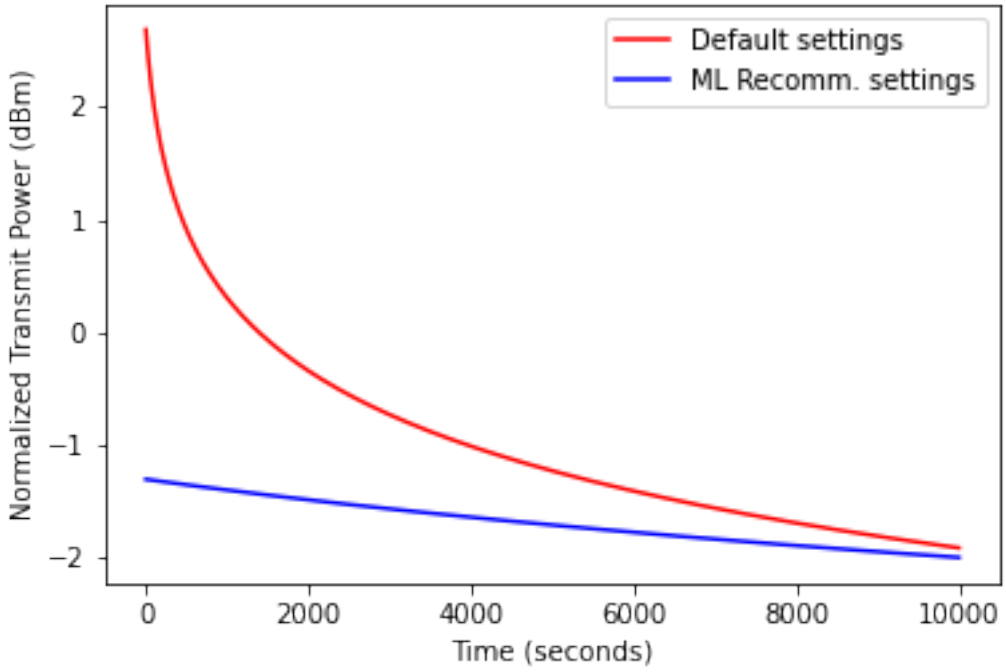


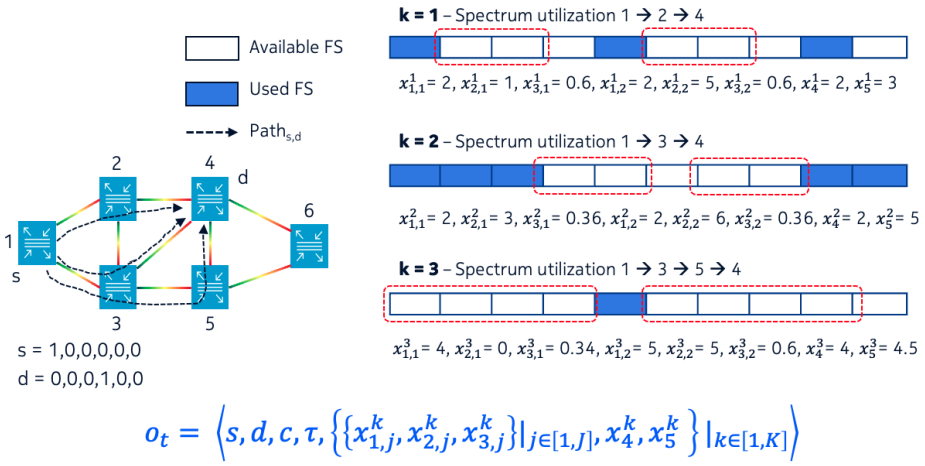

**Fig. 6.3.:** Power adjustment over time using power loop control

that the solution can be embedded within the microservice-based control framework introduced in the previous chapters, supporting modular and vendor-agnostic orchestration in open optical environments.

While the results demonstrate substantial improvements in provisioning speed and configuration accuracy, the approach remains constrained by the scalability limits of GPR and its dependence on comprehensive training data. Future work should explore sparse or online variants of GPR and dynamic retraining strategies to ensure adaptability in evolving network scenarios.

## 6.2 Contribution 2.2. Deep Reinforcement Learning aided Routing, Modulation format, and Spectrum Allocation

This work presents the design and implementation of *DeepSF-PCE*, a single-agent DRL-based Spectrum Fragmentation (SF) aware PCE developed to intelligently




$$o_t = \left\langle s, d, c, \tau, \left\{ \left\{ x^k_{1,j}, x^k_{2,j}, x^k_{3,j} \right\} |_{j \in [1,J]}, x^k_4, x^k_5 \right\} |_{k \in [1,K]} \right\rangle$$

**Fig. 6.4.:** DeepSF-PCE state representation

solve the RMSa problem in EONs. The system addresses the pressing need for autonomous and scalable decision-making frameworks in optical transport networks, where dynamic traffic conditions, spectrum fragmentation, and QoT constraints make real-time resource allocation increasingly complex.

By training a DRL agent to operate over a representation of the network state—including current spectrum occupancy and fragmentation metrics—DeepSF-PCE learns to select paths, modulation formats, and contiguous spectrum slots that not only meet connectivity constraints (i.e., continuity, contiguity, and non-overlapping rules), but also minimize long-term spectrum fragmentation. Unlike heuristic or rule-based RMSA methods, this agent develops a policy that generalizes to unseen network states and adapts to the evolving network context through reinforcement learning.

To ensure practical integration within a disaggregated optical control architecture, DeepSF-PCE is encapsulated as a modular network function and exposed through REST APIs that conform to the T-API standard, enabling direct interaction with software-defined controllers introduced in earlier chapters. This work thus contributes an intelligent, modular, and standards-aligned decision-making engine capable of autonomously improving network performance under dynamic operational conditions.

The objective of DeepSF-PCE is to learn optimal RMSA policies maximizing the number of served requests. It involves three components: the network state (i.e., the observation vector), the set of actions that can be performed by the DRL-agent (i.e., action space), and the reward function that gives feedback to the agent.

- The **action space**, $A$ is a set of discrete actions of size $k \times J + 1$, where $k$ is the number of candidate paths for each source-destination pair in the network and $J$ determines the number of candidate Frequency Slot (FS) blocks in each path. The additional action represents the possibility of rejecting a request.

- The **observation vector**, $o_t$ contains the source-destination pair in 1-hot encoding $(s, d)$, a normalized value of the service capacity requirement ($c$ in Gbps), the service duration ($\tau$) and the spectrum utilization of $k$ shortest paths (SP) between source and destination. For each path, we compute the starting indices $x^k_{1,J}$, the size of the FS block $x^k_{2,J}$, the resulting fragmentation state $x^k_{3,J}$ in case of selecting that particular FS block, the number of FS needed $x^k_4$ and the average FS block size $x^k_5$. For calculating the fragmentation state, we use the Shannon entropy ($H_{frag}$) at two different levels: link-based and path-based, as proposed in [139]. The objective is to provide to the agent the resulting spectrum fragmentation measure after allocating the service request. Higher level of SF lead to large values of $H_{frag}$, therefore we normalize $H_{frag}$, between [0-1]: ($e^{-H_{frag}}$). Thus, a common scale is preserved for every parameter in the observation vector. Fig. 6.4 shows an example of the state representation using the path-based fragmentation approach.

- The **reward function** considers the resource allocation capability of the agent for a given request and the resulting fragmentation caused by the allocation. If the agent can accommodate the service request, it will get a positive reward $+1$ plus an additional fragmentation-aware metric. If not, it will receive a negative reward $-1$. For that reason, we introduce the difference in fragmentation ($\Delta H_{frag}$) between $t$ and $t-1$. The reward function is defined as follows:

$$r_t = \begin{cases} 1 + e^{-\Delta H_{frag}} & \textit{successfully allocated} \\ -1 & \textit{otherwise} \end{cases} \quad (6.1)$$

The DRL agent was trained using two algorithms: Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO) provided by the stable-baselines library. Both agents are modeled with two fully-connected Deep Neural Networks (DNNs): Actor and Critic. The input layer size is equal to the length of $s_t$. 5 hidden layers, each one of 128 neurons, and the output layer consisting of $k \times J$ neurons for the Actor-network and 1 neuron for the Critic network. The difference between A2C and PPO lies in the learning phase. PPO estimates the policy gradients using the ratio between the new and old policy instead of using the logarithm of the new policy as in A2C [101]. We selected a discounted factor $\gamma = 0.96$ and a learning rate $\alpha = 10^{-4}$ as they produced the best results. The simulations consider an extended dynamic

network environment from [140], where service requests arrive following a Poisson process with arrival rate $\lambda = 10$ and a mean service duration of 25 units of time. The source-destination pair is randomly selected, and the capacity demand follows a uniform distribution between [10-200] Gbps. The C-band is considered for each link of the network topology (384 FSs of 12.5GHz). The distance-adaptive based impairment aware model explained in [141] is used to select the modulation format and, subsequently, determine the number of FS needed to accommodate the service request. The agents were trained for 1000 episodes, where each episode consisted of 1000 service requests. An episode begins with a fully available spectrum and ends when all service requests have been processed i.e., provisioned or rejected, by the agent. The objective is to prove the benefit of including spectrum fragmentation-related information when quantifying the service-blocking probability.

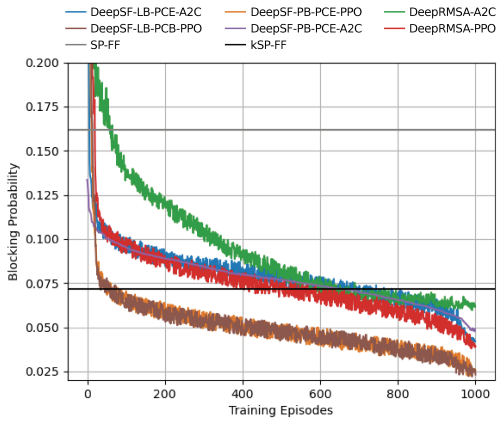
### 6.2.1 Performance Evaluation

The performance of the proposed DeepSF-PCE framework was assessed using the widely adopted NSFNET topology, composed of 14 nodes and 22 bidirectional fiber links. Each fiber supports the full C-band spectrum, discretized into 384 FSs of 12.5 GHz granularity. The evaluation is conducted using a dynamic traffic scenario, where connection requests arrive following a Poisson process and have exponentially distributed holding times, simulating realistic network conditions under time-varying traffic demands.

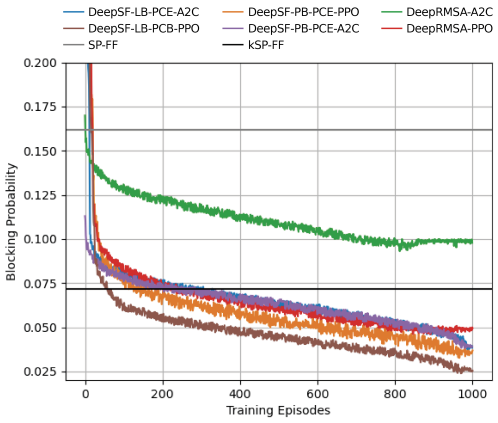
The learning environment is parametrized with $k = 5$ candidate shortest paths per source-destination (s-d) pair. For each path, the agent evaluates a discrete set of $J \in \{2, 3, 5\}$ spectrum block options, leading to an action space of cardinality $|A| = k \cdot J + 1$. The additional action accounts for the rejection of unsatisfiable requests. This design allows the agent to explore multiple allocation options while preserving computational tractability.

Figure 6.5 reports the request blocking probability as a function of training episodes for three distinct configurations of the spectrum allocation parameter $J$. As $J$ increases, the agent is exposed to a broader decision space, improving its ability to explore alternative allocations that reduce contention and fragmentation. However, this comes at the cost of increased observation and action space dimensionality, which directly impacts convergence time and learning stability.

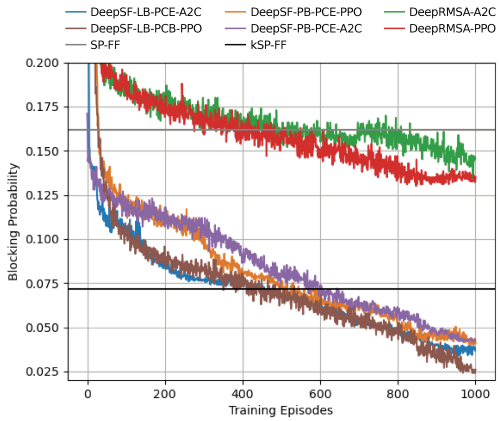
The observed trends confirm that DeepSF-PCE consistently and significantly outperforms baseline methods, including DeepRMSA [134], SP-FF (Shortest Path - First Fit),




(a) Blocking probability for $J = 2$




(b) Blocking probability for $J = 3$




(c) Blocking probability for $J = 5$

**Fig. 6.5.:** Blocking probability over training episodes for different values of $J$.

and kSP-FF (k-Shortest Paths - First Fit). Specifically, the proposed model achieves a relative blocking probability reduction of 28.5% for $J = 2$, 50% for $J = 3$, and 80.7% for $J = 5$ compared to DeepRMSA. These performance gains are attributed to two core innovations in DeepSF-PCE:

1. **Fragmentation-aware Observation Design:** By embedding the expected fragmentation variation $e^{-H_{\text{frag}}}$ into the observation vector for each candidate allocation, the agent learns to avoid spectrum allocations that lead to spectrum fragmentation, a major contributor to future blocking events. This proactive consideration of global spectrum health enables the agent to make decisions that optimize long-term spectrum utilization, rather than short-term acceptance rates.

2. **Reward Shaping with Entropy Variation:** The reward function penalizes allocations that increase entropy in the FS distribution, thus biasing the learning process toward spectrum-preserving decisions. This is particularly impactful in elastic optical networks, where fragmentation leads to stranded resources, i.e., small contiguous blocks of FS that cannot accommodate new demands despite sufficient total capacity.

Furthermore, the experiments reveal important insights into model behavior as a function of $J$:

- For smaller values of $J$ (e.g., $J = 2$), the agent converges rapidly due to the limited action space and lower-dimensional observation vectors (see Fig. 6.5a). However, the constrained number of allocation options limits the agent's capacity to find spectrum-efficient configurations under high-load scenarios.
- For intermediate values ($J = 3$), DeepSF-PCE demonstrates a strong balance between learning complexity and allocation flexibility (Fig. 6.5b). It achieves significant reductions in blocking with relatively fast convergence, making it a suitable configuration for many operational scenarios.
- For larger values ($J = 5$), the agent achieves the lowest blocking probability overall (Fig. 6.5c). However, the expanded state-action space requires longer training duration and more stable policy updates. This configuration is ideal for offline training environments or networks requiring highly optimized provisioning under heavy load.

The comparison between link-based and path-based entropy estimation further reveals the effectiveness of localized fragmentation awareness. While path-based measures provide a holistic view of contiguous spectrum availability across a full path, they fail to capture per-link bottlenecks that often constrain end-to-end provisioning. LB-based fragmentation metrics enable more granular and accurate estimation of spectrum health, leading to better policy learning and decision-making—especially in high $J$ configurations where nuanced spectrum dynamics emerge.

Additionally, PPO consistently outperforms the Advantage Actor-Critic (A2C) algorithm across all test scenarios. PPO's clipped objective function and adaptive step-size control mechanisms result in more stable training dynamics, improved sample efficiency, and reduced sensitivity to hyperparameter tuning. This confirms PPO's suitability for complex environments such as RMSA in EONs, where noisy and high-dimensional observations pose significant challenges for traditional policy gradient methods.

While the previous results have demonstrated the effectiveness of the DRL-based RMSA approach in terms of blocking probability and resource utilization, it is also important to analyze the learning dynamics of the agent to better understand its behavior during training.

In reinforcement learning, the evolution of the reward signal provides key insights into the convergence properties of the agent. In our case, the reward function is designed to capture both successful service provisioning and efficient spectrum usage. During training, the reward typically exhibits an initial phase of high variability, corresponding to exploration of the state-action space. This phase is followed by a progressive stabilization as the agent converges towards a policy that balances acceptance ratio and spectral efficiency.

Although a full convergence analysis is beyond the scope of this work, observations from the training process indicate that the agent reaches a stable performance regime after a sufficient number of episodes, with diminishing improvements in cumulative reward. This suggests convergence towards a locally optimal policy, which is consistent with the stochastic nature of DRL algorithms and the complexity of the RMSA problem.

It is important to note that convergence speed and stability are strongly influenced by hyperparameters such as learning rate, exploration strategy, and reward shaping. In large-scale optical network scenarios, the size of the state-action space can significantly slow down convergence, requiring careful tuning.

### 6.2.2 Conclusion

The proposed DeepSF-PCE framework introduces a deep reinforcement learning-based Path Computation Element capable of effectively reducing the blocking probability in dynamic RMSA scenarios. Through the integration of fragmentation-aware state features and reward functions, the agent learns to internalize the long-term effects of spectral fragmentation, resulting in significantly improved spectrum allocation decisions. Empirical evaluations across multiple configurations confirm that DeepSF-PCE outperforms traditional heuristics and baseline DRL models—such as SP-FF, kSP-FF, and DeepRMSA—achieving up to 80.7% reduction in blocking probability in high-complexity settings.

These improvements have several strategic implications. First, DeepSF-PCE enables greater service provisioning reliability under bursty or high-load traffic conditions,

thereby enhancing overall throughput. Second, its ability to minimize fragmentation leads to improved spectrum efficiency and reduces the operational overhead associated with spectrum defragmentation procedures. Third, by embedding fragmentation metrics directly into both the observation vector and reward structure, the framework introduces a form of anticipatory control that supports sustainable and long-term resource management. Finally, the agent's implementation as a T-API compliant PCE supports interoperability and real-world deployment within open, multi-vendor SDN-enabled optical infrastructures.

Despite these advantages, the single-agent reinforcement learning approach adopted in DeepSF-PCE presents inherent scalability limitations. As the optical network grows in size and complexity, the expansion of the state and action spaces increases the computational burden during training and slows convergence. Moreover, a centralized agent may struggle to capture distributed and localized network state variations, leading to suboptimal decisions in scenarios that require fine-grained control granularity. These limitations highlight the need for future extensions toward scalable multi-agent architectures that preserve the intelligent behavior of DeepSF-PCE while enhancing its adaptability in large-scale, heterogeneous environments.

## 6.3 Contribution 2.3. Multi-Agent Graph Convolutional Reinforcement Learning for RSA

As mentioned previously, the single-agent DRL approach may lead to a long training period. Moreover, relational features, such as non-fragmented spectrum blocks between links in a path, need to be extracted by manually applying an equivalent kernel.

In this work, we propose the multi-agent Graph Convolutional Reinforcement Learning approach, called *MAGC-RSA*, to solve the Routing and Spectrum Assignment (RSA) problem in a distributed manner. Leveraging the similarity of network topology and graph structure, GCN models each DRL agent as a network node in the topology. Based on the global view provided by SDNC and the observation of its neighbors, the DRL agent, which corresponds to the source node of the request, decides to select the optimal path and spectrum resources. Moreover, we adopt the Attention mechanism [142] as the kernel in the convolutional layer to extract latent features for faster convergence even in a large observation space of many nodes. Additionally, DRL agents are trained with a fragmentation-aware reward function, which leads to better spectrum utilization. To the best of our knowledge,

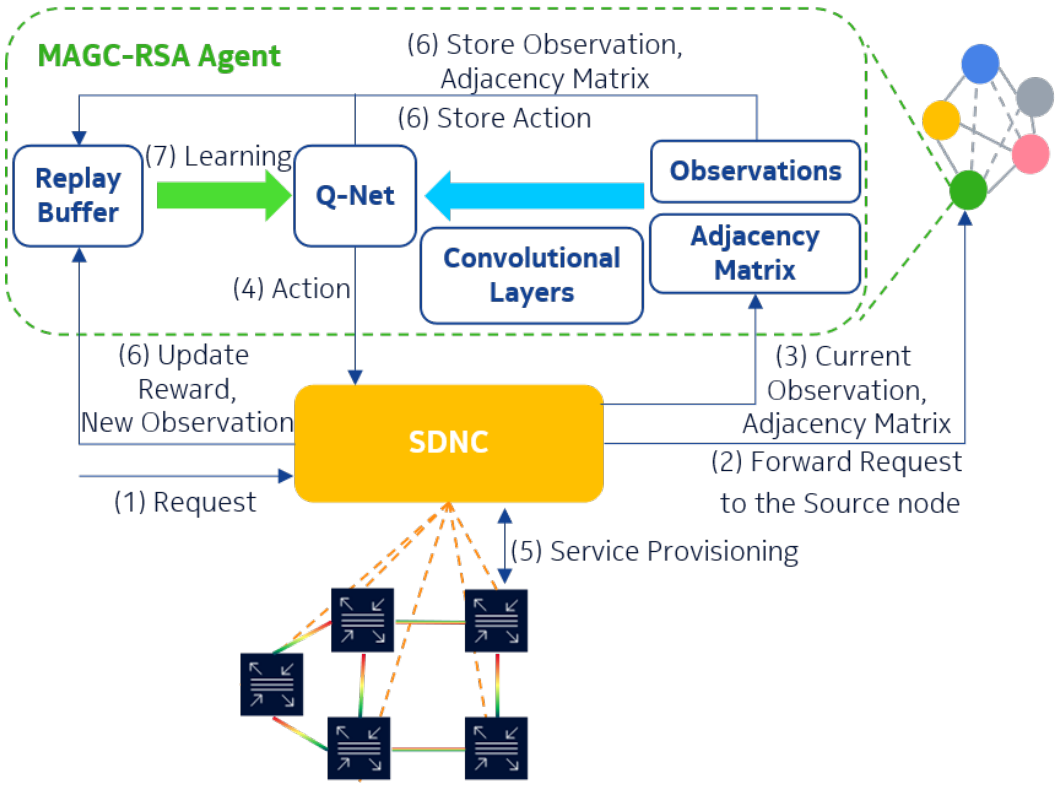



**Fig. 6.6.:** Operation principle of MAGC-RSA

this centralized training and distributed execution method is applied for the first time to solve the RSA problem in EONs. Our approach achieves a lower blocking probability as compared to the heuristic K-Shortest Path First-Fit and another MADRL (Multi-Agent DRL) solution.

We model the MAGC-RSA algorithm as a graph $G = (N, E)$, where agents represent network nodes and edges express the connection between them. For every service request, the agent that corresponds to the source node is in charge of finding the optimal path and spectrum resources. Fig. 6.6 describes the operation principle of our proposed solution.

MAGC-RSA architecture consists of three main components: the Encoder, the Convolutional Layer, and the Q Network. They are illustrated in Fig. 6.7 and described as follows:

- *1) Observation Encoder:* The local observation of agent $a_i$ at time step $t$ is denoted as vector $o_i(t)$. Each observation vector $o_i(t)$ contains the one-hot encoding of the source and destination of the request, the number of requested frequency slots, the spectrum utilization of $k$ candidate shortest paths between the source and destination, the spectrum utilization of $d$ links originating from

$K$ neighbors of $a_i$. The value of $d$, $K$ may vary between environments. In this paper, we choose $d = 3$, $K = 4$. Once constructed, each observation vector will be encoded by the Multi Layer Perceptron (MLP) to create the feature vector $f_i$ of agent $a_i$.

The feature vector $f_i$ is defined by multiplying two matrices $M_{Adj} \times F(t)$. $M_{Adj}$ is a $N \times N$ matrix, where $m_{i,j} = 1$ if agent $a_i$ has a connection towards $a_j$, $m_{i,j} = 0$ otherwise. $F(t)$ is the feature matrix that aggregates all feature vectors of all agents at time $t$.

- *2) Convolutional layers:* These layers take as input the feature vector $f_i$, which is the output of MLP, and an adjacency matrix $M_{Adj}$ of the agent.

  The convolutional layer, namely the Kernel relation, is responsible for generating the latent features. In this context, the convolutional kernel uses multi-head dot-product attention to compute interactions between agents. Considering an agent $a_i$ and the set $\mathcal{E}_i$ of its $K$ neighbors, an input feature is expressed in a query, key and value representation by each independent attention head [142].

  In order to model the relationship between each agent and its neighboring agents (i.e., between $i$ and $j \in \mathcal{E}_i$) for an attention head $m$, we use the following formula:

$$\alpha_{ij}^m = \frac{exp(\delta \,.\, W_q^m f_i \,.\, (W_k^m f_j)^T)}{\sum_{e \in \mathcal{E}_i} exp(\delta \,.\, W_q^m f_i \,.\, (W_k^m f_e)^T)} \tag{6.2}$$

  where: for each agent $a_i$, there are a set of entities $\mathcal{E}_i$ ($K$ neighbors and the node $n_i$ itself) in the local region, and $\delta$ is a scaling factor.

  For each attention head, the value representations of all the input features are weighted by the relation and summed together. Subsequently, for agent i the outputs of the M attention heads are concatenated and fed into the $\sigma$-function (a MLP with ReLU non-linearities) in order to produce the convolutional layer output. Therefore, the latent feature vector $l_i$ is formulated as follows:

$$l_i = \sigma(concatenate[\sum_{j \in \mathcal{E}_i} \alpha_{ij}^m W_v^m f_j, \forall m \in M]) \tag{6.3}$$

- *3) Q-Network:* For each agent, the Q-network is applied on the concatenated features of the previous layers, allowing to take into consideration the cooperation at different scopes.

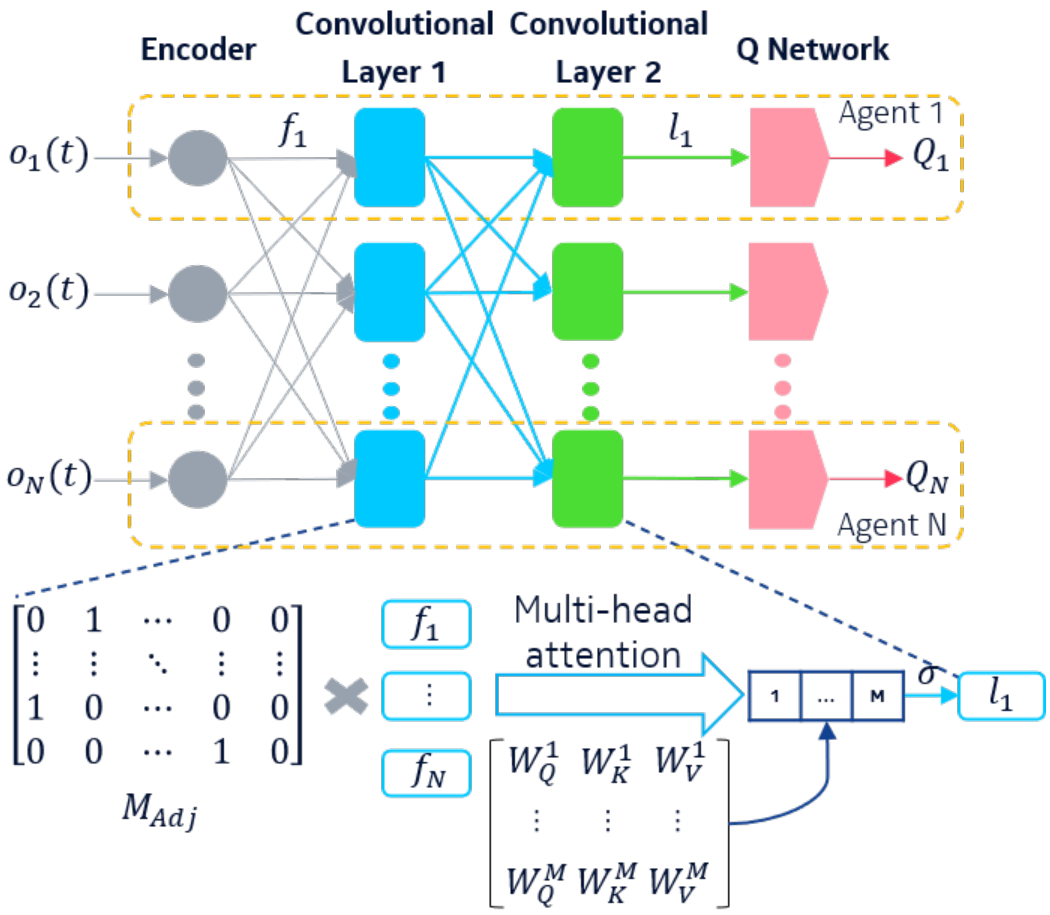



Fig. 6.7.: MAGC-RSA architecture.

The Q-network aims here to set the actions by calculating the Q-values. Therefore, during the training phase, the tuple ($O$, $A$, $O'$, $R$, $M_{Adj}$) is stored into the buffer $B$ at each time step $t$; where: $O$= ($o_1$, ... $o_N$) is the set of observations, $A$= ($a_1$, ... $a_N$) is the set of actions, $O'$= ($o'_1$, ... $o'_N$) is the set of next observations, $R$= ($r_1$, ... $r_N$) is the set of rewards, and $M_{Adj}$= ($M_{Adj_1}$, ... $M_{Adj_N}$) is the set of adjacency matrix. Next, a random mini-batch of size S is sampled from $B$ and hence, we minimize the following loss:

$$Loss_Q(\theta) = \frac{1}{S}\sum_{S}\frac{1}{N}\sum_{i=1}^{N}((r_i + \gamma max_{a'}Q(O'_i, a'_i; \theta')) - Q(O_i, a_i; \theta))^2 \quad (6.4)$$

With: $\gamma$ is the discount factor, and $\theta$ is the parametrization of the $Q$ function.

It is worth noting that during the computation of Q-loss in the learning phase, the underlying graph can change over time, which prevents the convergence of Q and leads to some learning instabilities. To deal with the latter issue, $M_{Adj}$ is kept unchanged in two successive time steps. Therefore, in order to update the parameters of the latter scheme, the Q-loss gradients of all agents are accumulated. Each agent minimizes not only its own Q-loss but also the Q-loss of the other agents it collaborates with. Each agent communicates only with its $K$ neighbors which makes the scheme easily applicable to large-scale GC-MARL systems.

Deep-Q-learning (DQL) is implemented to train our model where the future value estimation is used as the target for the current estimation. In multi-agent environments, the decision-making of actions is complicated, especially when the environment is highly dynamic. Therefore, the temporal relations regularization facilitates action-taking by adopting a cooperative behavior providing coherent actions in the long term to maximize the final reward. Thus, the relation representation (i.e., the attention weight distribution over the neighboring agents produced by the relation kernel) should also be consistent and stable for a short period.

To make the learned attention weight distribution stable over time-steps, we propose temporal relation regularization. Inspired by temporal difference learning, we use the attention weight distribution in the next state as the target for the current attention weight distribution. We adopt the Kullback-Leibler (KL) divergence to measure how the current attention weight distribution is different from the target attention weight distribution. Minimizing the KL divergence as a regularization will encourage the agent to form a consistent relation representation and hence consistent cooperation.

It is worth noting that we only use the target network to produce the target Q-value. For the calculation of KL divergence between relation representations in two time-steps, we apply the current network to the next state to produce the target relation representation. This is because relation representation is highly correlated with the weights of feature extraction. However, the update of such weights in the target network always lags behind that of the current network, making the relation representation produced by the target network not consistent with that produced by the current network.

Given the observation vector and the adjacency matrix described previously, a MAGC-RSA agent takes an action based on the output of the Q-Network. It selects the path and the first index of the contiguous frequency slots block among $k \times J$ actions in

**Tab. 6.1.:** Hyperparameters

| Parameters | Values |
|---|---|
| Learning rate | $10^{-4}$ |
| Batch size | 64 |
| # neighbors | 4 |
| # attention head | 4 |
| # hidden layers | 2 |
| Hidden layers dimension | 64 |
| Discount factor $\gamma$ | 0.96 |

the discrete action space; where $k$ is the number of candidate shortest paths, and $J$ is the number of sufficient frequency slots blocks.

As in DeepSF-PCE, the reward function considers not only the resource allocation capability of the agent for a given request but also provides useful information concerning the impact of this assignment concerning the effect of fragmentation:

$$r_t = \begin{cases} 1 + e^{-\Delta H_{frag}} & \textit{successfully allocated} \\ -1 & \textit{otherwise} \end{cases} \quad (6.5)$$

### 6.3.1 Performance evaluation

We conducted the experiments by extending the environment in [140]. In our implementation, all agents cooperate and interact with one single environment. Thus, after performing an action, all agents update their observations and adjacency matrix, based on the network information provided by SDNC. The simulation considers the dynamic scenario where requests arrive following a Poisson process with an arrival rate of 10 and have a mean service duration of 25 units of time, which follows the exponential distribution. Each request's source-destination pair is randomly selected and its bandwidth demand is evenly distributed between [2-4] frequency slots. The agents were trained in 1000 episodes. Each episode consists of 10000 services. The hyperparameters are specified in Table 6.1.

The performance of MAGC-RSA is experimented on the 14-node NSFNET topology for small-scale problems and the 28-node Pan-European topology for larger-scale problems.

Figures 6.8a and 6.8b illustrate the evolution of the request blocking probability over the training episodes. MAGC-RSA consistently outperforms baseline approaches

such as kSP-FF, SP-FF, and DeepRMSA. Notably, the blocking probability is reduced by 16.62% in NSFNET and by 58.33% in Pan-European networks compared to DeepRMSA, confirming the superior generalization capability of our multi-agent architecture in diverse network conditions. These results affirm the efficacy of MAGC-RSA in managing spectral resources and adapting to fluctuating demand.

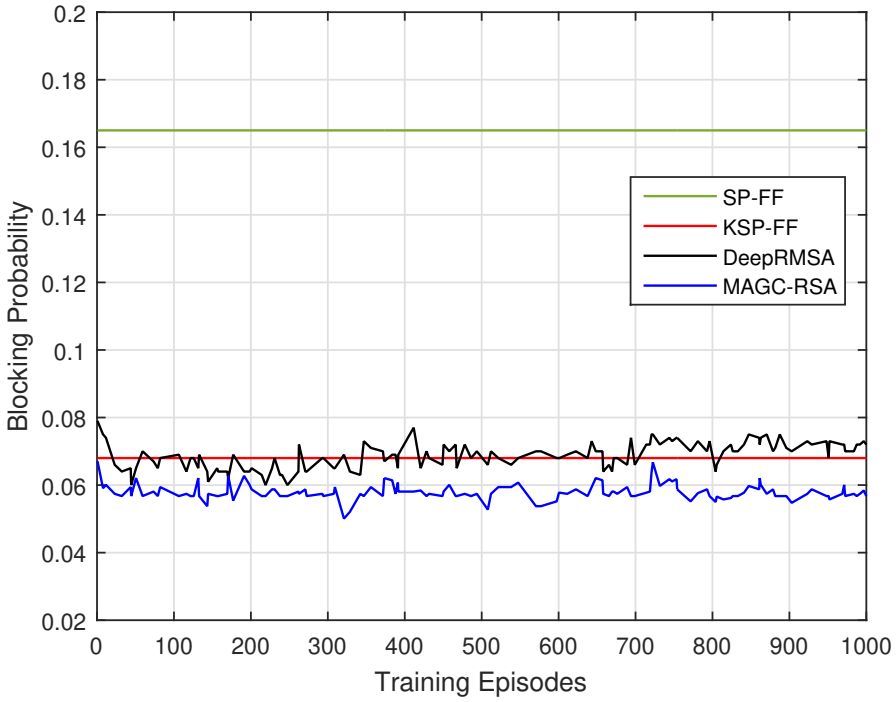
Compared to DeepSF-PCE, which adopts a single-agent paradigm, MAGC-RSA demonstrates multiple advantages. First, by distributing decision-making across network nodes, it alleviates the scalability bottlenecks observed in DeepSF-PCE where a central agent must learn a policy over a large action and observation space. In MAGC-RSA, agents exploit their local view and neighborhood observations to make decentralized yet coordinated decisions, significantly accelerating convergence and improving policy specialization. This results in a more scalable architecture suitable for future large-scale elastic optical networks.

Moreover, the incorporation of GCNs with multi-head attention mechanisms in MAGC-RSA enables relational inductive biases that traditional feedforward DRL models cannot capture. GCNs allow agents to learn latent topological embeddings by leveraging the network's adjacency structure, effectively modeling the interdependencies between neighboring nodes. This capability is crucial for capturing path continuity and spectral contiguity constraints across the network. The multi-head attention kernel, acting as a dynamic and adaptive filter, further enhances the model's expressiveness by enabling context-sensitive feature aggregation. This facilitates robust representation learning and empowers agents to focus on spectrum blocks that maximize long-term utility while minimizing fragmentation.

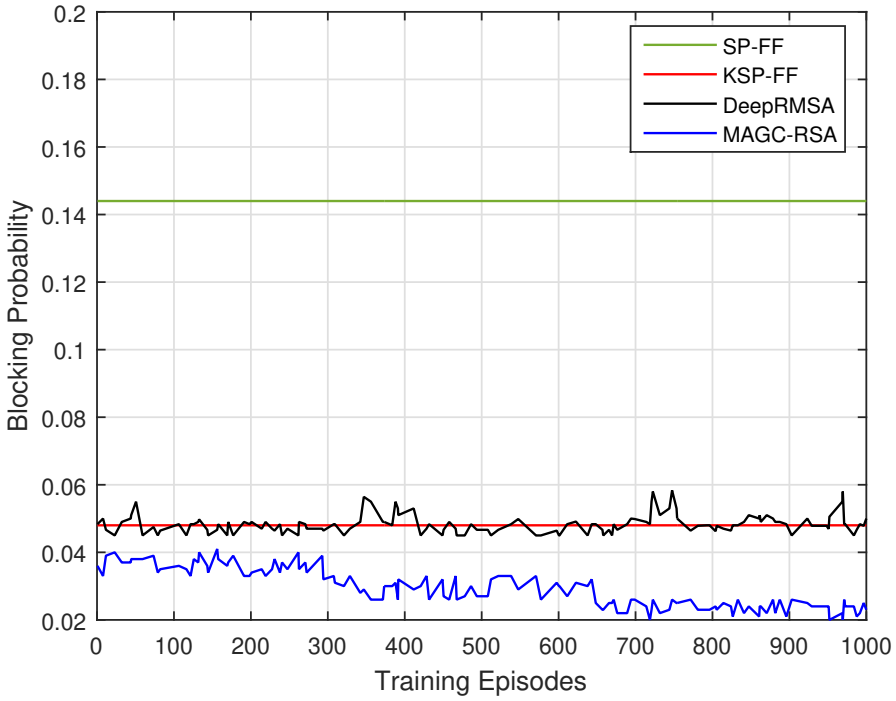
In addition, MAGC-RSA integrates a fragmentation-aware reward function, similar to DeepSF-PCE, yet improves upon it by enabling agents to collaboratively minimize fragmentation across distributed decision points. This results in enhanced spectrum compactness and defers the need for expensive defragmentation operations. Furthermore, temporal relation regularization stabilizes inter-agent cooperation over time, promoting consistent and interpretable policies in dynamic environments.

### 6.3.2 Conclusion

The proposed multi-agent graph convolutional reinforcement learning framework leverages distributed decision-making to address the scalability limitations of centralized approaches. However, this paradigm introduces additional considerations related to inter-agent cooperation and communication overhead.




**(a)** Request blocking probability with NSFNET topology.




**(b)** Request blocking probability with Pan-European topology.

**Fig. 6.8.:** MAGC-RSA blocking probability over training episodes

In the proposed architecture, agents implicitly cooperate through the exchange of information embedded in graph representations, enabling each agent to capture both local and global network state features. This distributed perception allows agents to make coordinated decisions without requiring a fully centralized controller. Nevertheless, the degree of coordination depends on the frequency and richness of information exchange.

From an operational perspective, inter-agent communication introduces overhead in terms of bandwidth consumption, synchronization requirements, and processing latency. In large-scale optical networks, where the number of agents may grow proportionally with the number of nodes or domains, the cumulative communication cost can become significant. This creates a trade-off between the level of coordination and system scalability.

Compared to centralized approaches, the multi-agent framework reduces the computational burden on a single entity and improves fault tolerance, as decisions are distributed across agents. However, it may also lead to suboptimal global decisions if agents operate with partial or outdated information. Ensuring consistency across agents therefore requires mechanisms for synchronization or consensus, which may further increase communication complexity.

Additionally, training multi-agent systems introduces challenges such as non-stationarity, where the environment perceived by each agent changes as other agents update their policies. This can impact convergence stability and requires careful design of training strategies, such as centralized training with decentralized execution or parameter sharing.

Overall, the multi-agent approach provides improved scalability and flexibility compared to centralized solutions, but at the cost of increased communication overhead and coordination complexity. The choice between centralized and distributed control therefore depends on the specific requirements of the network, including size, latency constraints, and reliability considerations.

In conclusion, RL holds immense promise for advancing the automation and intelligence of optical networks. However, despite these advancements, significant challenges remain in deploying RL-based solutions in real-world optical networks. One of the primary barriers is the requirement for vast amounts of high-quality training data. Optical networks do not naturally generate sufficient labeled datasets, and obtaining such data from real deployments is both expensive and time-consuming. Furthermore, RL models require extensive exploration to learn optimal policies, which is infeasible in operational networks where errors can lead to severe service

disruptions. Unlike simulated environments, real-world optical networks must prioritize reliability and service continuity, making it difficult to deploy RL models without extensive safety mechanisms.

To bridge the gap between RL research and real-world deployment, hybrid approaches combining RL with traditional optimization techniques and heuristic-based fallback mechanisms should be explored. Moreover, integrating offline learning with transfer learning strategies may allow RL models to generalize better across diverse network conditions, minimizing the risks of real-time learning failures.

## 6.4 Contribution 2.4. Dynamic Drift-Adaptive Ensemble-based QoT Classification

Quality of Transmission (QoT) classification using ML models has become a hot topic since it shows more effective and accurate than numerical and analytical approaches (i.e., the split-step Fourier method, the Gaussian Noise model). We consider the requirement to verify the feasibility of a new lightpath before the provisioning phase. The QoT violation assurance imposes extra complexity because there might be many active connections that need QoT value verification. Most QoT ML models are implemented with the assumption that the training and testing datasets follow the same pattern. However, this is no longer the case in the dynamic Optical Transport Network (OTN) scenario, where data distribution in the target environment may differ from the learning one. Thus, QoT ML models could experience performance degradation or drift. In addition, the lack of a drift adaptation mechanism prevents the model from dynamically improving its performance, which causes a complete update on new samples. Model retraining requires an advanced scheme to organize new online data, tune hyperparameters, and record and serve various versions of the model.

Model drift occurs when the performance of an ML model degrades over time. There are various causes, such as data distribution shift, model objective change, etc. There are two main categories: data drift and concept drift. Data drift occurs when the distribution of the features in prediction is different from the training phase. For example, the traffic matrix of connectivity requests could be shifted from a high number of low-capacity demands (i.e., 100Gbps, 200 Gbps) to more high-capacity ones (i.e., 300Gbps, 400Gbps) due to the bandwidth requirement of 5G and Beyond 5G services. Concept drift indicates that there is a change in the underlying relationships between features and outcomes. Particularly, a defective

optical receiver would report a failure, although the receiving Optical Signal-to-Noise ratio (OSNR) is well above the threshold. Hence, labels are reported incorrectly, which leads to the degradation of the ML model's performance

There are two main methods to detect drift: data monitoring (or statistical method) and performance monitoring (or learner-based method). Statistical or hypothesis tests are used to detect actual changes in data distribution. There are popular methods such as the Population Stability Index (PSI), Kullback-Leibler (KL) divergence, Jensen-Shannon (JS), Kolmogorov-Smirnov (KS) test, etc. Particularly, PSI measures how much a population has shifted over time. In addition, the KS test quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples. Monitoring model performance metrics is the most fundamental approach for drift detection. There are common metrics to monitor such as confusion matrix, accuracy, recall, F1 score, etc. Some approaches focus on the error rate and use the error rate-based drift detection method. For example, the Drift Detection Method (DDM) algorithm can be used to detect any significant increase in error rates. We adopt the Early Drift Detection Method (EDDM) [143] as the performance-based drift detection algorithm that determines the occurrence of model drift by monitoring the degree of model performance degradation to improve the detection in the presence of gradual drift; while keeping a good performance with abrupt drift. EDDM examines the average distance between two prediction errors ($\bar{p}_t$) and its standard deviation ($\bar{s}_t$) instead of the number of errors. We obtain $\bar{p}_{\text{max}}$ and $\bar{s}_{\text{max}}$ when $\bar{p}_t + 2\bar{s}_t$ reaches its maximum. The algorithm considers two parameters $\alpha$ and $\beta$ as the warning and drift threshold, respectively. Particularly, EDDM will signal a Warning if:

$$\frac{\bar{p}_t + 2\bar{s}_t}{\bar{p}_{\text{max}} + 2\bar{s}_{\text{max}}} < \alpha, \tag{6.6}$$

or will signal a Drift if:

$$\frac{\bar{p}_t + 2\bar{s}_t}{\bar{p}_{\text{max}} + 2\bar{s}_{\text{max}}} < \beta. \tag{6.7}$$

This method enables enhanced detection of performance degradation and facilitates timely responses to drift scenarios in dynamic systems.

The drift adaptation procedure is referred to as automated model updates in the network data analytics automation process, as its main purpose is to improve model performance by updating the learning model. After identifying a concept drift, learning models should be able to adapt to new concepts and enhance model performance. Existing drift-adaptive learning techniques fall into two primary

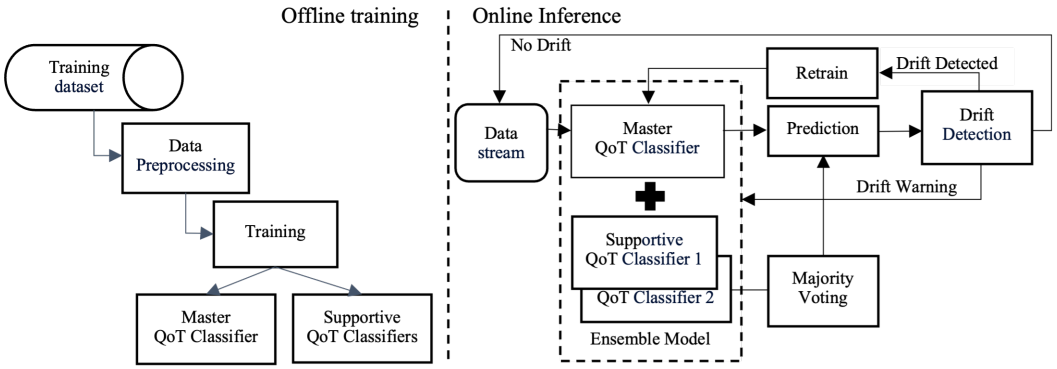



**Fig. 6.9.:** Dynamic Drift-Adaptive Ensemble-based Quality of Transmission Classification Framework

categories: incremental learning and ensemble learning techniques. Incremental learning is the process of learning each incoming data sample in chronological order and partially updating the learner. Hoeffding Tree is a basic incremental learning method that employs the Hoeffding inequality to determine the minimum number of data samples necessary for each split node, thus updating nodes to adapt to new samples. Ensemble online learning models are advanced drift-adaptive learning methods that integrate the output of multiple base learners for performance improvement.

In this work, we introduce the Dynamic Drift-Adaptive Ensemble-based Quality of Transmission Classification Framework (DAEQoT) to overcome this problem. DAE-QoT allows the use of a state-of-the-art batch-learning ML model while combining other online methods to form an ensemble, which enhances the prediction accuracy in the event of potential drift. This approach uses the Early Drift Detection Method to monitor the performance of the models. Evaluations show significant improvement in prediction accuracy, as compared to both online and offline techniques. Moreover, it can avoid severe drift events without the need for a complete retraining of the model.

The framework has two main components: a set of QoT Classifiers and a drift detection mechanism. There are two phases involved in this framework: offline training and online inference. In the training phase, both Offline and Online ML models are considered and trained with the same learning set. We choose CatBoost, which is an ML framework based on gradient-boosted decision trees, as the Master QoT Classifier. In addition, online ML models such as Adaptive Random Forest (ARF), Aggregated Mondrian Forest (AMF), Stream Random Patches (SRP), and Leverage Bagging (LB) are considered Supportive QoT Classifiers for the Master. While CatBoost is an offline model, which can only be trained in batch, those online

models allow incremental learning along the data stream. We select EDDM presented previously as the drift detection mechanism in the online phase since it can provide no drift, warning, and drift detected information. The architecture is presented in Figure 6.9.

Moreover, Algorithm 1 describes the operation of DAEQoT in the online phase. The system begins at the "No drift" phase (Signal = 0), and EDDM is initiated with the warning threshold $\alpha$ and drift threshold $\beta$. DAEQoT uses only the Master QoT Classifier $M$ to predict the QoT $\tilde{y}_i^M$ of each sample $s_i$ from the data stream, based on the feature vector $x_i$. Next, the true label $y_i$ and the predicted one $\tilde{y}_i^M$ are fed to EDDM to compute the average distance between two prediction errors and its standard deviation.

Assuming that the prediction accuracy of $M$ degrades after a while; the system transitions into the "Warning" phase, as signaled by EDDM. Since the performance has decreased, DAEQoT combines $M$ with other Supportive QoT Classifiers $P$. The ensemble-predicted label $\tilde{y}_i^{DAEQoT}$ is derived from the prediction of each classifier in DAEQoT by conducting the Majority Voting mechanism. This output and the true label are then fed to EDDM following the same process. Additionally, all samples in the Warning phase are recorded in batch $B$ as new learning data to retrain $M$ at a later phase.

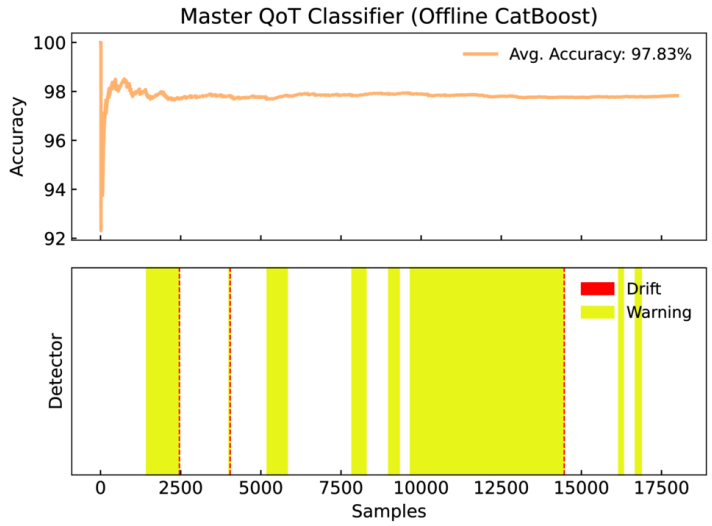
On the other hand, since classifiers in the Supportive set $P$ are online learners, they can fit their models incrementally with each sample. If the performance of DAEQoT continues to decay, a complete drift may occur. In this phase, $M$ is retrained using the recorded samples from $B$. The system is then reset to the initial "No drift" phase.

### 6.4.1 Performance evaluation

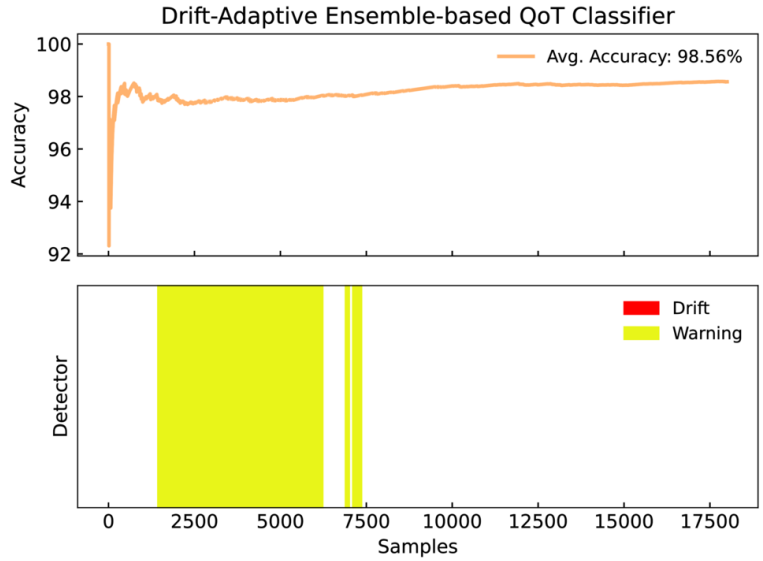
The proposed framework was implemented by extending CatBoost and River Python library. We used 20000 samples from the Lightpath Dataset 03 and split them into the learning (20%) and testing (80%) sets. All ML models are trained offline with the same learning set and can achieve an accuracy of more than 95%. Subsequently, ML models are put in the online inference, where each sample in the testing set is fed sequentially to the models. Since SRP and LB provide the best accuracy during the offline training phase, we choose them as two Supportive models, together with the Master, to form DAEQoT.

**Algorithm 1** Drift-Adaptive Ensemble-based QoT Classification

**Input:**
$S$: data stream
$Signal$: the output of EDDM ($Signal \in \{0, 1, 2\}$)
$M$: Master QoT Classifier
$P$: set of $k$ Supportive QoT Classifiers
$B$: Batch to store samples in Warning zone
**Output:**
$DAEQoT$: Drift-Adaptive Ensemble-based QoT Classifier
**procedure** DAEQoT OPERATION
/* **Initialize** */
$DAEQoT \leftarrow M$ ▷ Let $DAEQoT$ contain only $M$
Initialize $EDDM_{\alpha,\beta}$
$B \leftarrow \emptyset$
$Signal \leftarrow 0$ ▷ No drift
**for** each $s_i \in S$ **do**
**if** $Signal == 0$ **then**
$\tilde{y}_i^M \leftarrow M(x_i)$
$Signal \leftarrow EDDM(y_i, \tilde{y}_i^M)$
**else if** $Signal == 1$ **then** ▷ In Warning zone
$DAEQoT \leftarrow M \cup P$
$\tilde{y}_i^M \leftarrow M(x_i)$
$\{\tilde{y}_i^{P_j}\} \leftarrow \{P_j(x_i)\}$ where $j \in \{1, \ldots, k\}$
$\tilde{y}_i^{DAEQoT} \leftarrow$ MajorityVoting$(\tilde{y}_i^M, \tilde{y}_i^{P_1}, \ldots, \tilde{y}_i^{P_k})$
$Signal \leftarrow EDDM(y_i, \tilde{y}_i^{DAEQoT})$
$B \leftarrow B \cup (x_i, y_i)$
$\{P_j\} \leftarrow \{P_j(x_i, y_i)\}$ where $j \in \{1, \ldots, k\}$
**else** ▷ Drift detected
$M \leftarrow M(B)$ ▷ Retrain $M$ on collected batch $B$
$B \leftarrow \emptyset$
Reset $EDDM_{\alpha,\beta}$
$Signal \leftarrow 0$
**return:** $DAEQoT$




(a) Master QoT Classifier




(b) Drift-Adaptive Ensemble-based QoT Classifier

**Fig. 6.10.:** QoT classifier accuracy and detected warning/drift zones

Figure 6.10a and 6.10b depict the performance of the Master QoT Classifier without Ensemble learning and model retraining (Offline Master) and our DAEQoT approach. DAEQoT outperforms the Master QoT Classifier in accuracy and the capability to avoid severe drift. Without any drift adaptation method, the Offline Master encounters drift three times. On the other hand, DAEQoT never reaches the drift threshold thanks to the Drift-Adaptive Ensemble-based mechanism, which enhances its performance and mitigates drift. Figure 4 shows the performance comparison of the proposed framework with other approaches namely ARF, AMF, SRP, LB, Offline,

and Online Master. The Online Master follows the same principle as the offline one; however, it is retrained in batch mode at every drift point. According to Table 6.2, DAEQoT clearly shows the best performance (98.56%) among approaches, while having a reasonable execution time. The average accuracy of Leverage Bagging (98.47%) is slightly smaller than DAEQoT. However, it takes nearly double the time for Leverage Bagging (3m51), as compared to DAEQoT (1m49), to reach such accuracy. Both Online and Offline Master achieve a reliable prediction of 97.93% and 97.83%, respectively. Since Online Master is updated with new samples during the online operation, it can adapt to the change in the data stream, which leads to higher accuracy. Moreover, both methods are based on an offline ML model, so they can execute within a short amount of time (under 11s). Other online ML models such as ARF, AMF, SRP are not as good as the offline ones in both accuracy and time measurements.

While the proposed drift-adaptive ensemble-based QoT classifier demonstrates improved robustness under non-stationary conditions, several limitations must be considered when evaluating its applicability in real-world optical networks.

A first important aspect is the sensitivity to drift detection thresholds. The performance of the system depends on the ability to accurately detect changes in the underlying data distribution. If the threshold is set too low, the system may trigger frequent and unnecessary model updates, leading to increased computational overhead and potential instability. On the other hand, if the threshold is too high, significant performance degradation may occur before adaptation is triggered. This highlights the need for careful calibration of drift detection mechanisms based on network characteristics.

Another limitation relates to update latency. The adaptation process, which involves retraining or updating ensemble components, introduces a delay between drift detection and model adaptation. In highly dynamic environments, this latency may

**Tab. 6.2.:** Performance of Drift Adaptation methods

| Model | Accuracy | Execution Time |
|---|---|---|
| ARF | 95.22% | 16.7s |
| AMF | 96.26% | 14.6s |
| SRP | 96.95% | 13.9s |
| Master QoT Classifier (Offline) | 97.83% | 7.7s |
| Master QoT Classifier (Online) | 97.93% | 10.8s |
| LB | 98.47% | 3 min 51 s |
| DAEQoT | 98.56% | 1 min 49 s |

result in temporary degradation of QoT prediction accuracy, potentially impacting network decisions. Reducing this latency requires efficient retraining strategies and possibly incremental learning techniques.

Transferability across different network topologies and configurations also remains a challenge. Models trained on a specific network scenario may not generalize well to other topologies with different physical characteristics, traffic patterns, or equipment configurations. This limits the direct portability of the approach and may require retraining or fine-tuning when applied to new environments.

The ensemble nature of the model increases computational and operational complexity compared to single-model approaches. Managing multiple models, monitoring their performance, and coordinating updates introduces additional overhead that must be considered in deployment.

Despite these limitations, the proposed approach provides a flexible framework for handling non-stationarity in optical networks, highlighting the importance of adaptive learning mechanisms in dynamic environments.

### 6.4.2 Conclusion

The introduction of the Dynamic Drift-Adaptive Ensemble-based QoT Classification (DAEQoT) framework represents a significant advancement in the application of machine learning techniques to support QoT assurance in dynamic and heterogeneous optical transport networks. Unlike traditional static classifiers that assume a stationary data distribution, DAEQoT is specifically designed to operate in environments characterized by evolving traffic demands, topology variations, and non-deterministic physical layer impairments. By leveraging a hybrid architecture that integrates a high-performance offline master model (CatBoost) with a pool of adaptive online learners (e.g., SRP, LB), the system maintains a high prediction accuracy while dynamically reacting to performance degradation via a robust drift detection mechanism based on the EDDM.

Nevertheless, the deployment of DAEQoT in production-grade optical environments presents several non-trivial challenges. One of the primary limitations lies in the sensitivity of drift detection thresholds, which must be carefully tuned to avoid false positives or delayed responses under varying network loads. In highly dynamic environments, sudden and overlapping drift events may occur, and existing statistical indicators may be insufficient to capture complex drift patterns involving

both label noise (e.g., faulty receivers) and distributional shifts (e.g., bursty high-bandwidth services). Furthermore, while the ensemble approach mitigates accuracy loss during transition phases, it does not completely eliminate the latency introduced by the retraining of the offline model once a full drift is detected. Maintaining a balance between responsiveness and stability requires careful orchestration of model synchronization, especially in SDN controllers orchestrating services in real-time.

Moreover, the reliance on labeled QoT data for drift detection and retraining poses an additional challenge in real deployments, where continuous supervision is impractical. This limitation highlights the need for future integration with automated telemetry pipelines capable of streaming and labeling optical performance metrics in near real-time. Finally, although the framework demonstrates excellent performance under synthetic datasets, additional research is required to evaluate its robustness and transferability across varying topologies, equipment vendors, and noise regimes.

In conclusion, DAEQoT offers a technically sound and highly promising approach for resilient QoT estimation in the context of dynamic optical networks. It bridges the gap between offline prediction accuracy and online adaptability by incorporating drift-aware intelligence into the ML pipeline. While further enhancements are required to fully address issues of scalability, real-time integration, and generalization under mixed drift scenarios, DAEQoT lays a strong foundation for the next generation of autonomous QoT assurance mechanisms in SDN-controlled, multi-vendor optical infrastructures.

# 7 Conclusion

Part II has explored the integration of ML techniques in the context of optical transport networks, presenting a series of practical, data-driven solutions that address key challenges in control, management, and performance assurance. Through a combination of supervised learning, ensemble modeling, and reinforcement learning—both single-agent and multi-agent—the chapter has demonstrated the effectiveness of ML-based approaches in optimizing critical tasks such as RMSA, QoT estimation, and transponder configuration under the constraints of disaggregated, dynamic, and multi-vendor environments.

By leveraging ML, we have shown how optical networks can transition from static, heuristic-based operations to intelligent, adaptive systems capable of making real-time decisions based on continuously evolving network conditions. Notable contributions include:

- **ML-aided Optical Transponder Configuration Optimizer:** A Gaussian Process Regression-based surrogate model to estimate GOSNR and optimize transponder configuration in disaggregated infrastructures, significantly reducing setup delays and power optimization overhead.
- **DeepSF-PCE:** A fragmentation-aware DRL agent for RMSA that reduces blocking probability and improves spectrum efficiency by incorporating fragmentation metrics into both state and reward functions.
- **MAGC-RSA:** A scalable, multi-agent graph convolutional reinforcement learning model that enables distributed RSA decisions through local observations and cooperative learning across network nodes.
- **DAEQoT:** A drift-adaptive ensemble-based QoT classification framework that maintains high prediction accuracy over time by combining robust offline models with online learners and real-time drift detection mechanisms.

These innovations represent a significant leap forward in the field of intelligent optical networks. The ability to seamlessly integrate ML models into live SDN control loops, execute autonomous optimization tasks, and maintain performance over time despite data drift or evolving network conditions marks a clear step

beyond traditional rule-based automation. Each solution presented contributes not only a specialized function but also demonstrates how learning-based systems can collaborate within a broader AI-native control architecture.

Importantly, this work establishes the first unified demonstration of how heterogeneous ML models—ranging from time-series forecasters to DRL agents and drift-adaptive classifiers—can operate cohesively across the optical control plane. This level of modularity, scalability, and data-driven intelligence has not been achieved in prior work and showcases a novel methodology for managing the growing complexity of dynamic optical transport systems.

From an operational perspective, the proposed solutions elevate the control architecture of optical networks to **Level 4** of the TM Forum Autonomous Networks framework. At this level, networks exhibit self-optimizing behavior, where ML models—trained on live telemetry and historical patterns—continuously analyze, predict, and adapt system behavior in real time without operator intervention. This degree of autonomy enables closed-loop decision-making, resource optimization, and resilience in the face of environmental or traffic variability, significantly reducing manual effort while improving quality of service and operational efficiency.

Nevertheless, the results also reveal the challenges that remain to be addressed in order to transition from cutting-edge research to robust, large-scale deployments. Issues such as data scarcity, model generalization, interpretability, and coordination in multi-agent systems must be systematically addressed. Furthermore, the performance and reliability of these systems depend on well-engineered ML lifecycle management—a dimension still underdeveloped in many optical control environments.

As we move toward highly automated, intelligent transport infrastructures, ML lifecycle management will become indispensable for scaling innovation sustainably. The next chapter addresses this critical requirement by introducing a cloud-native architectural framework for managing ML workflows in optical networks, ensuring consistency, reproducibility, and operational resilience in ML-assisted transport control planes.

In conclusion, the research and implementations presented in this part represent a pioneering and methodologically rigorous contribution to the field. By embedding intelligence directly into the control plane and demonstrating modular, learning-driven solutions across diverse network functions, this work lays the technical foundation for the next generation of autonomous, ML-native optical transport networks.

# Part III

## Compositional Machine Learning Framework for Open and Disaggregated Optical Network Control

The preceding parts of this dissertation have explored the transformative role of Machine Learning (ML) in the evolution of optical transport networks, ranging from the architectural shift towards open and disaggregated systems to the integration of intelligent decision-making mechanisms across various control and management layers. In particular, we have examined the impact of ML on key operational challenges, including routing and spectrum assignment, transponder configuration, impairment-aware path computation, and Quality of Transmission (QoT) estimation. Through the development of targeted ML-based components—such as DeepSF-PCE, MAGC-RSA, and DAEQoT—we have demonstrated that data-driven intelligence can significantly improve network performance, resilience, and adaptability in dynamic and heterogeneous optical environments.

However, as these individual ML solutions mature and proliferate, several practical challenges emerge with respect to their deployment, lifecycle management, and orchestration in real-world systems. Firstly, ML models are not static entities: they require continuous retraining, monitoring, and validation to remain effective in the presence of evolving traffic patterns, network topologies, and operational conditions. Secondly, the integration of multiple ML applications—each developed for a specific task—raises questions about composability, interoperability, and governance of their interaction. Finally, the absence of standardized, scalable mechanisms for automating ML operations (MLOps) poses a barrier to transitioning from prototype-level deployments to production-grade AI-native network control planes.

To address these challenges, this part introduces a cloud-native, automated *Compositional Machine Learning Framework (CMLF)* that unifies the lifecycle management and orchestration of ML models deployed in optical transport networks. CMLF builds upon the architectural principles established in earlier chapters—specifically, the disaggregated control framework and microservice-based deployment model—and extends them with dedicated MLOps pipelines to manage the end-to-end workflow of ML applications. These include stages such as data ingestion, feature extraction, model training and validation, drift detection, versioning, serving, and rollback.

Central to CMLF is the notion of compositionality: the ability to dynamically combine multiple ML models into higher-level composite services based on contextual requirements and evolving network states. Instead of monolithic, siloed ML agents, CMLF enables the flexible chaining of reusable model components—such as classifiers, regressors, decision policies, and optimization modules—into coordinated workflows that adapt to specific control and management tasks. This capability is particularly beneficial in disaggregated optical domains, where diverse vendors,

telemetry formats, and functional scopes demand modular and interoperable intelligent components.

The framework also embraces the MLOps paradigm, integrating continuous integration and deployment (CI/CD) principles with domain-specific control logic to automate the deployment, monitoring, and evolution of ML services. By leveraging containerized services, event-driven orchestration, and configuration-as-code approaches, CMLF ensures that ML applications remain robust, maintainable, and responsive to operational feedback. This not only reduces the overhead for operators but also increases trust in the automation layer by providing transparent auditability and rollback mechanisms.

In this part, we begin by presenting the architectural design of the CMLF and its integration into the existing microservice-based control plane introduced previously. We then describe a set of implemented ML applications—originating from the previous chapters—and how they are registered, composed, and orchestrated within the framework. These applications include drift-aware QoT classifiers, reinforcement learning agents for spectrum assignment, and impairment-aware path computation modules. Finally, we demonstrate the deployment and operation of CMLF in a simulated disaggregated optical network, showcasing its capability to coordinate heterogeneous ML services in real time.

Through this final contribution, we aim to bridge the gap between isolated ML functionalities and scalable Artifitial Inteligence (AI)-native network automation by providing a modular, extensible, and lifecycle-aware framework for intelligent optical transport control. The CMLF not only addresses the technical and operational challenges discussed in earlier chapters but also lays the groundwork for future work in composable AI services for programmable networks.

# 8 Introduction

The emergence of 6G networks marks a transformative shift toward truly AI-native infrastructures, characterized by their capacity for autonomous operation and minimal reliance on human intervention. As these networks become increasingly complex, distributed, and service-driven, the integration of Artificial Intelligence (AI) and Machine Learning (ML) into network control and orchestration becomes not only beneficial but indispensable. These technologies are instrumental in enabling autonomous lifecycle management tasks such as anomaly detection, fault prediction, traffic forecasting, dynamic network optimization, root cause analysis, and alarm storm mitigation. Given this wide spectrum of use cases, the number of ML models deployed within operational network environments is expected to grow exponentially—from dozens to potentially thousands—each supporting distinct inference tasks or decision-making logic.

The incorporation of AI/ML, including Deep Neural Networks (DNNs), Large Language Models (LLMs), and Generative AI (GenAI), into network operations is envisioned to accelerate service delivery and increase system resilience. In particular, intelligent applications embedded within network management planes can substantially reduce provisioning latency, enhance fault response times, and automate repetitive operational tasks—thereby improving Quality of Experience (QoE) for end users and reducing operational expenditures for network operators.

Despite this growing maturity, most existing ML deployments are limited in scope and design, typically involving a single-task model integrated into an MLOps pipeline. As the complexity of ML models and workflows increases, so too do the demands on computational infrastructure, data pipelines, and lifecycle management tools. Figure 8.1 depicts the envisioned AI/ML deployment and service architecture, highlighting the integration of multi-cloud MLOps platforms and edge-cloud continuum to support distributed intelligence across heterogeneous environments.

To manage this complexity at scale, MLOps has emerged as a crucial discipline encompassing end-to-end practices for managing the ML lifecycle—from data acquisition and feature engineering to model training, validation, deployment, and performance monitoring. Depending on the deployment requirements, MLOps platforms can be instantiated over public cloud infrastructures (e.g., Google Cloud

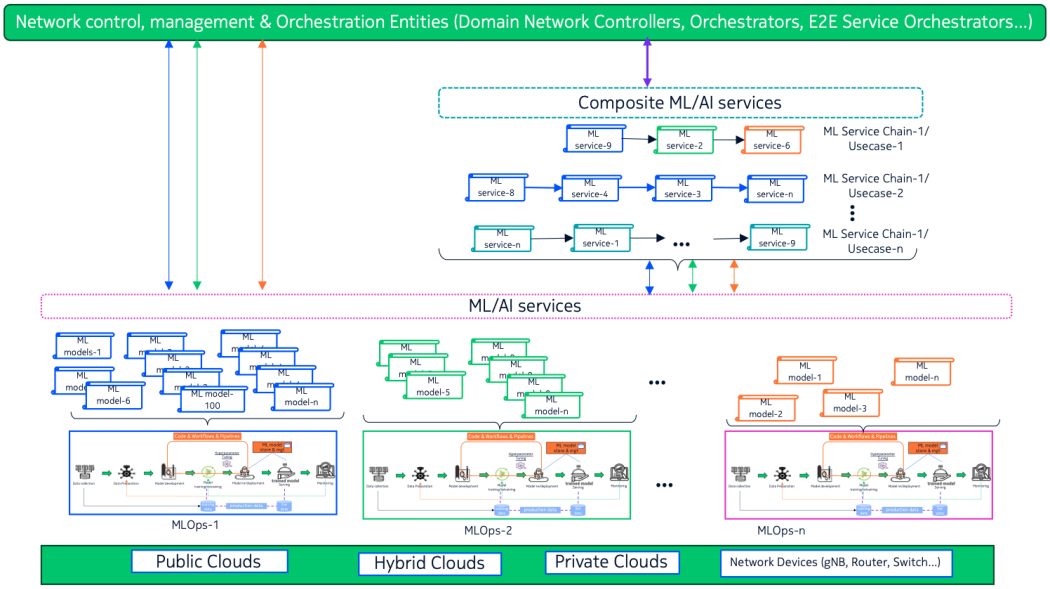



**Fig. 8.1.:** AI/ML System Deployment and Service Architecture

Platform, AWS SageMaker, Microsoft Azure ML), private clouds, or hybrid/multi-cloud environments. These deployments often involve heterogeneous toolchains and lack standardized interfaces, posing significant interoperability challenges for multi-vendor environments.

In this context, the paradigm of Compositional Machine Learning (CML) offers a promising architectural approach. CML allows for the modular assembly of ML services into structured workflows or pipelines—termed ML Service Chains (MLSC)—that can be executed either sequentially or in parallel. These chains enable the decomposition of complex tasks into simpler subtasks, fostering reusability and reducing model development time. For example, a failure prediction model can be seamlessly chained with a traffic forecasting model to anticipate both the occurrence and potential impact of failures, enabling proactive traffic rerouting and enhanced service continuity.

However, enabling such compositional intelligence in operational networks introduces a new set of architectural and orchestration challenges, especially when ML services are distributed across a federated, multi-domain infrastructure. These include: (i) dynamic discovery and versioning of ML services across heterogeneous MLOps environments, (ii) intelligent and automated orchestration of ML service chains, (iii) lifecycle management and runtime adaptation of MLSCs under changing conditions, and (iv) exposure of programmable APIs for integration with higher-level control and orchestration systems.

This contribution addresses these challenges by introducing a modular, programmable, and standards-compliant Compositional Machine Learning Framework (CMLF) for next-generation optical network intelligence. The primary goal of the framework is to facilitate the development, deployment, and runtime management of ML workflows that combine multiple inference models into cohesive services. In particular, the framework addresses the following open research problems:

- *Service Discovery and Registry:* How can ML services be dynamically discovered, updated, and indexed in large-scale, heterogeneous environments spanning distributed MLOps platforms?
- *Autonomous Composition:* How can the composition and chaining of ML services be intelligently automated, avoiding the inefficiencies and risks of manual orchestration?
- *Runtime Execution and Coordination:* How can large-scale MLSCs be executed, monitored, and coordinated across geo-distributed infrastructures to ensure correctness, scalability, and low latency?
- *Programmability and Extensibility:* How can such ML chains be exposed to external consumers via programmable interfaces while maintaining abstraction, security, and portability?

These challenges are compounded by the diversity of MLOps platforms and deployment strategies currently in use. Broadly, ML operating environments may be built in-house (custom operator solutions), procured from third-party vendors and deployed on private infrastructure, or fully managed by public cloud providers. Each of these options typically exposes non-standard APIs for model packaging, deployment, and monitoring, resulting in significant interoperability gaps.

Accordingly, the proposed CMLF not only provides an orchestration layer for ML service chains but also serves as a mediation interface across heterogeneous MLOps environments. By abstracting deployment complexity and enabling dynamic ML pipeline composition, the CMLF lays the foundation for scalable, intelligent, and reusable ML-driven control and automation services in future AI-native network architectures.

# 9 Contribution 3. Conceptual Foundations of the CML Framework

This contribution introduces a novel Compositional Machine Learning Framework (CMLF) designed around a modular microservice-based architecture. The proposed system incorporates several key components, including the Compositional Machine Learning Framework Manager (CMLFM), the Stitch Engine (SE), the Workflow Execution Engine (WFEE), the MLOps Adapter (MLOA), and a dedicated Graphical User Interface (GUI). Each of these modules plays a distinct role in supporting the automation and management of Machine Learning Service Chains (MLSCs) within heterogeneous and widely distributed infrastructure environments.

Building on the challenges identified in previous sections regarding the dynamic composition and orchestration of ML workflows, the proposed framework introduces several architectural innovations. Chief among these is the integration of a workflow-based orchestration mechanism, which enhances scalability and flexibility in coordinating distributed ML services across different runtime environments. Furthermore, the framework incorporates a dynamic stitching mechanism, supported by the Stitch Engine, which enables the on-demand composition of MLSCs using a dedicated stitching algorithm.

The operational model of the framework is structured into three distinct phases: discovery, design, and run-time. This phased approach facilitates continuous discovery of available ML services, the composition of service chains at design time, and the autonomous execution and lifecycle management of these chains at run-time. To ensure interoperability across diverse deployment platforms, the framework adopts a plug-and-play abstraction model, facilitated by the MLOps Adapter. This allows seamless integration with various MLOps systems, regardless of vendor or infrastructure type.

The high-level architecture of the framework is depicted in Figure 9.1, which illustrates the interplay between the individual components.

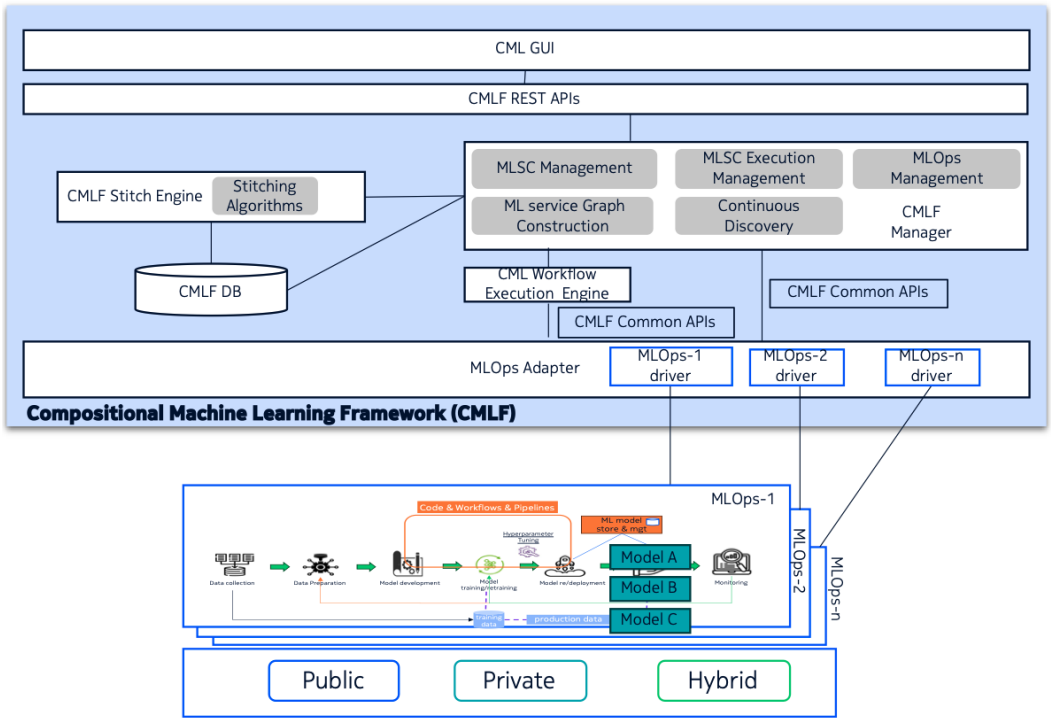



Fig. 9.1.: Compositional Machine Learning Framework Architecture

## 9.1 CMLF Architecture

The architecture of the proposed Compositional Machine Learning Framework (CMLF) is designed to address the operational and integration challenges inherent in compositional machine learning, while also serving as a foundational enabler for intent-based ML operations. As illustrated in Figure 9.1, the framework comprises five key components, each contributing to the end-to-end orchestration, execution, and management of Machine Learning Service Chains (MLSCs).

At the core of the system is the *Compositional Machine Learning Framework Manager (CMLFM)*, which orchestrates the complete lifecycle of MLSCs. An MLSC is defined as an ordered set of interdependent ML services that are logically chained to solve a specific, often multi-faceted, problem. The CMLFM includes a dedicated execution management module that generates workflow execution templates from the designed service chains and oversees their execution during runtime. A crucial feature of the CMLFM is its continuous discovery mechanism, which monitors the availability and state of registered ML services. This process also constructs unidirectional graphs representing service dependencies and compatibility relationships—these graphs serve as the foundational structure for the Stitch Engine. Furthermore, the CMLFM exposes a comprehensive set of REST APIs, allowing external entities such as SDN

controllers, orchestration layers, and higher-level network management applications to interact with and consume the functionalities offered by the CMLF. Additionally, the CMLFM integrates a specialized module for managing the underlying MLOps platforms and their corresponding drivers.

The *Stitch Engine (SE)* plays a pivotal role in the design phase by guiding the composition of ML services into coherent MLSCs. Relying on the service dependency graphs constructed during discovery and using a dedicated stitching algorithm, the SE validates and recommends feasible sequences of ML models to be composed into an operational chain. This ensures that model interoperability, data format compatibility, and functional correctness are preserved throughout the service chain.

During runtime, the *Workflow Execution Engine (WFEE)* is responsible for executing the MLSC workflows that were synthesized during the design phase. This engine manages the orchestration logic of chained ML services, invoking each ML component in the correct order and coordinating data exchange and execution control between them.

To ensure seamless integration with heterogeneous MLOps environments, the architecture includes an *MLOps Adapter (MLOA)*. The MLOA serves as an execution mediation layer and hosts a set of MLOps drivers, each responsible for interfacing with a specific MLOps platform. These drivers abstract the heterogeneity of public, private, or hybrid deployment infrastructures by translating high-level CMLF operations into platform-specific MLOps API calls. The lifecycle of each MLOps driver, including registration, monitoring, and retirement, is managed by the CMLFM to maintain system consistency and modularity.

Finally, the framework incorporates a dedicated *Graphical User Interface (GUI)*, which offers a unified portal for managing the entire compositional ML workflow. The GUI is designed to simplify user interactions by supporting visual model stitching, monitoring pipeline status, and triggering operations across the ML service chain. By abstracting underlying complexity, it enables domain experts to leverage ML capabilities without deep knowledge of MLOps internals.

The CMLF operates on top of existing MLOps platforms to exploit their capabilities in enhancing the reliability, scalability, and efficiency of ML pipelines. These capabilities include but are not limited to version control, CI/CD, automated model testing, and continuous monitoring. In particular, the framework leverages MLOps telemetry to track the inference accuracy of individual models, enabling the detection of performance drift and triggering model adaptation workflows when necessary. Full lifecycle management of ML models—covering training, deployment, revalidation,

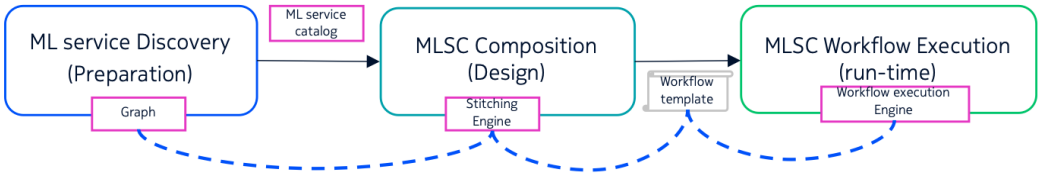



**Fig. 9.2.:** CMLF Operation Phases

and deprecation—is executed within the MLOps layer, which acts as the operational substrate for the entire compositional framework.

## 9.2 CMLF Operational Processes

The Compositional Machine Learning Framework (CMLF) is designed around three principal operational phases—*discovery*, *design*, and *runtime*—as illustrated in Fig. 9.2. In addition to these core processes, an external operational component is required to interface with the underlying MLOps platforms.

### 9.2.1 ML Service Continuous Discovery (Preparation Phase)

The *continuous discovery* process enables CMLF to automatically detect and monitor the availability of ML services deployed across registered MLOps platforms. As depicted in Fig. 9.3, following successful registration of an MLOps platform, its end-point is added to an internal registry managed by CMLF. The continuous discovery module initiates by constructing a unidirectional ML service graph, representing service dependencies based on metadata (input/output parameters). This graph is stored in the system's graph database and regularly updated through polling mechanisms.

Each polling cycle ensures that newly deployed ML models or updates to existing services are reflected in the graph structure. Fig. 9.4 illustrates how this graph is constructed and incrementally updated during each polling iteration.

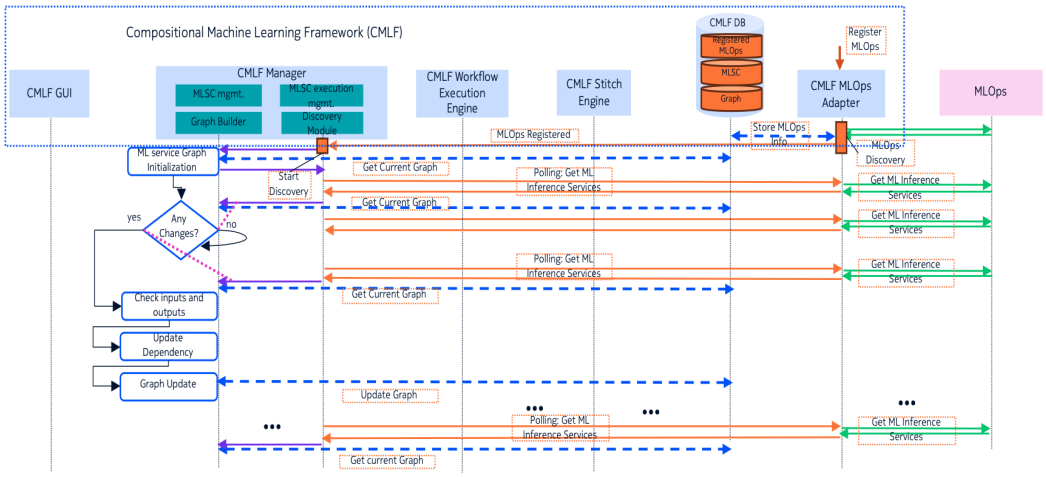



**Fig. 9.3.:** Continuous Discovery Process

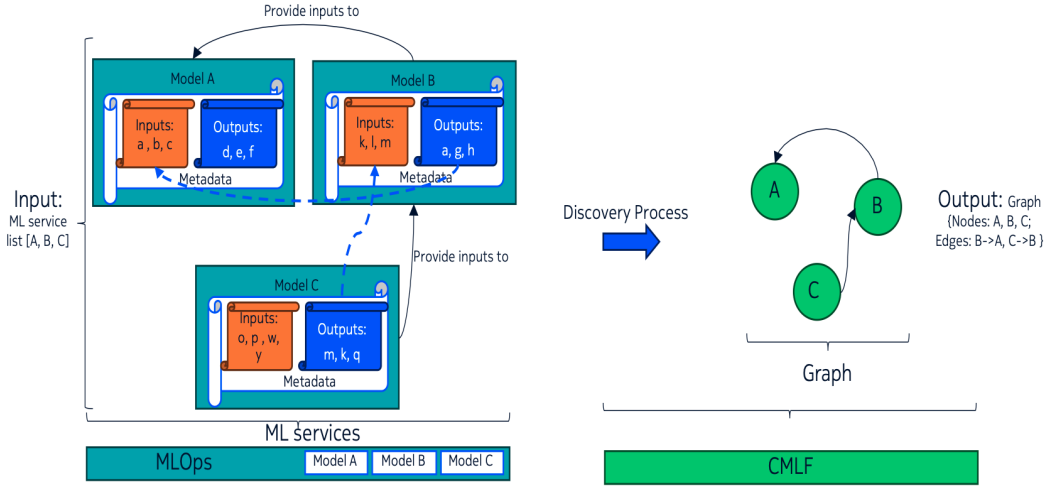



**Fig. 9.4.:** Graph Building Based on Metadata Inputs and Outputs During Discovery

### 9.2.2 Machine Learning Service Chain (MLSC) Composition (Design Phase)

During the design phase, the CMLF enables the construction of *Machine Learning Service Chains* (MLSCs), which represent ordered sequences of ML services collaboratively solving a complex task. The process begins with an MLSC stitching request—comprising an unordered list of ML services—sent to the CMLF Manager. This request is then forwarded to the Stitching Engine, which consults the previously discovered service graph and invokes the internal stitching algorithm to identify all valid compositions (i.e., stitchable MLSC candidates).

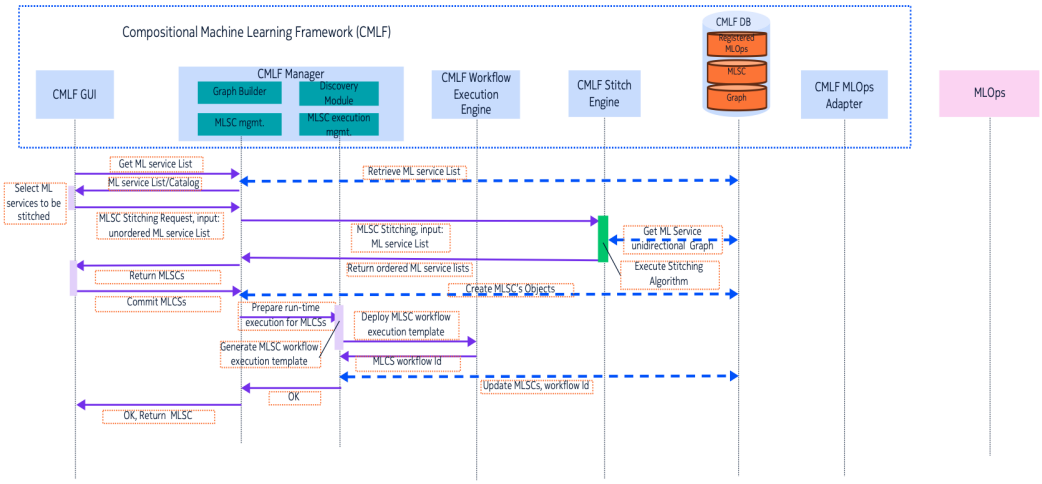



**Fig. 9.5.:** MLSC Composition Process

The resulting ordered service chains are returned to the user for validation and selection. Upon selection, the MLSC is committed to the system and translated into a concrete workflow execution template, following predefined execution blueprints. This workflow is subsequently deployed to the Workflow Execution Engine (WFEE) for run-time invocation. The end-to-end composition process is illustrated in Fig. 9.5.

#### Dynamic ML Service Stitching Algorithm

The dynamic stitching algorithm serves as the core logic behind the composition process (Fig. 9.6). Upon receiving a list of candidate ML services, the system verifies any changes in service metadata and constructs updated graphs reflecting the input-output compatibility among services. The algorithm partitions services into subsets based on the graphs in which they appear, then searches for all directed paths within each graph that consist of at least two valid ML services.

Each path is evaluated as a candidate MLSC, and only chains that meet the composition criteria—e.g., input-output parameter consistency—are retained. The algorithm produces a set of valid MLSCs, which can then be selectively instantiated depending on the use case or execution strategy.

### 9.2.3 MLSC Workflow Execution (Runtime Phase)

At runtime, external systems initiate service execution by submitting an MLSC invocation request to the CMLF Manager through an asynchronous API. In response,

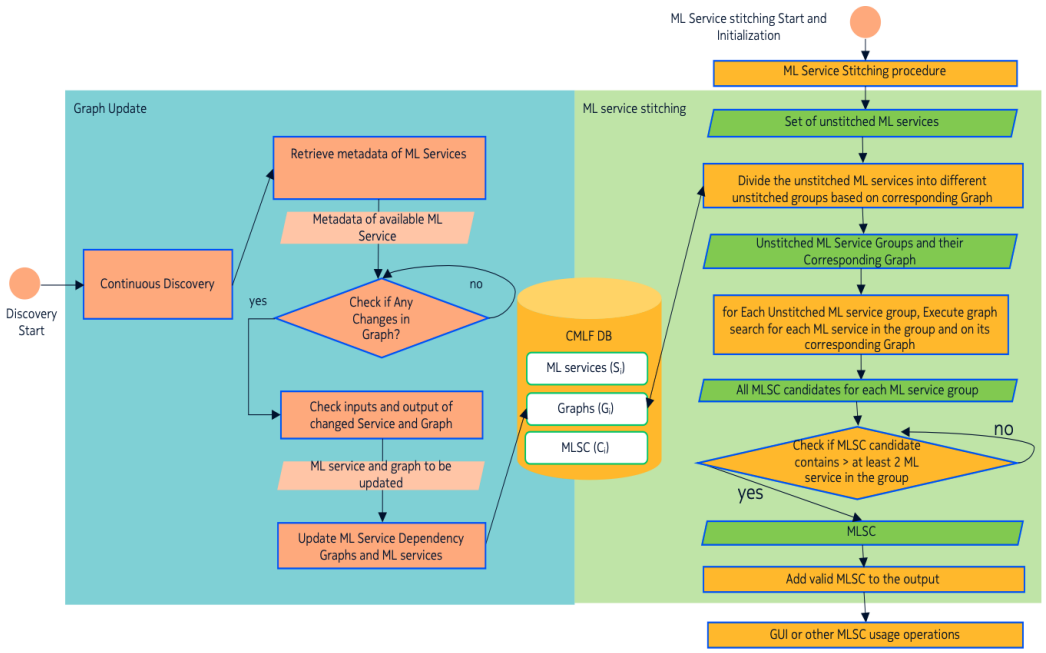



**Fig. 9.6.:** ML Service Stitching Algorithm

the CMLF assigns a unique execution identifier and marks the execution status as "pending". The Workflow Execution Engine (WFEE) then launches the corresponding MLSC workflow as defined in the execution template.

The WFEE sequentially dispatches prediction requests to the MLOps Adapter, which translates them into platform-specific API calls. These are routed to the appropriate MLOps endpoints for inference. Upon completion of all stages in the service chain, the CMLF updates the execution status to "completed" and stores the resulting outputs. External systems can monitor progress by polling the execution endpoint using the assigned identifier. The full runtime orchestration is shown in Fig. 9.7.

### 9.2.4 MLOps Management Processes

MLOps platforms, while external to the CMLF, are essential for hosting and executing the ML models that power each service. To integrate with the CMLF, each MLOps platform must undergo a registration process (see Fig. 9.8). During registration, platform metadata and API endpoints are extracted and associated with a corresponding MLOps driver—a software component that maps CMLF's generic operations to platform-specific implementations.

MLOps drivers are modular and dynamically managed by the CMLF Manager, supporting registration, update, and removal through dedicated APIs. This architectural

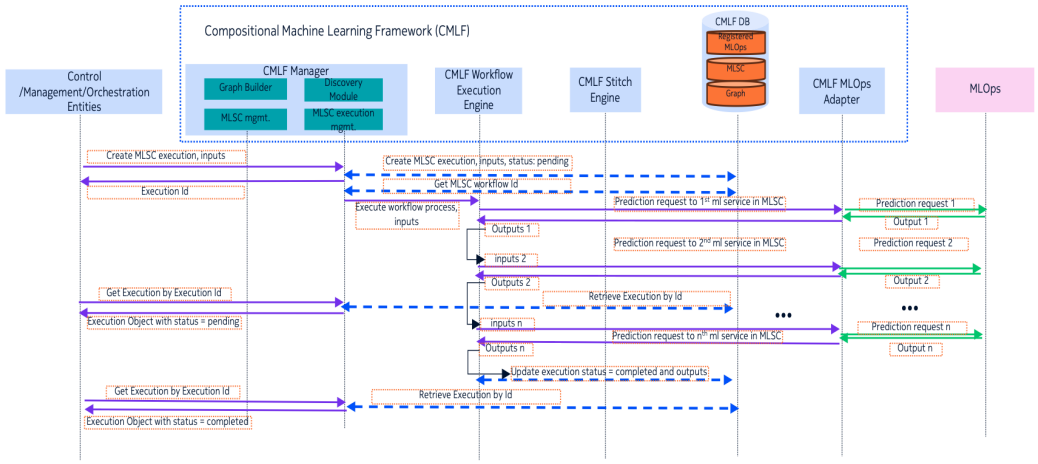



**Fig. 9.7.:** CMLF Runtime Execution Process

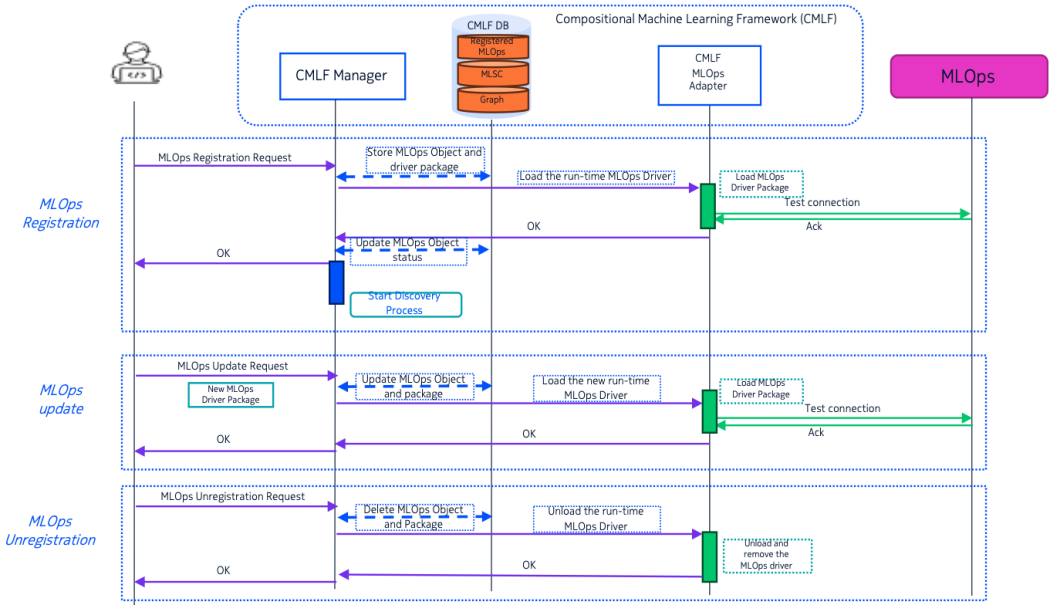



**Fig. 9.8.:** MLOps Management Processes

modularity ensures compatibility with a diverse range of MLOps platforms and allows seamless interaction with public, private, or hybrid cloud-based ML infrastructure.

# 10 Contribution 4. Application of CMLF in Open and Disaggregated Optical Networks

As the demand for ultra-high capacity and low-latency communication continues to intensify in B5G and 6G systems, optical transport networks are undergoing a profound transformation toward disaggregated, software-defined, and intent-driven architectures. This shift enables increased flexibility and vendor-neutral interoperability, but it also introduces substantial operational complexity, especially in dynamic multi-domain environments. AI/ML techniques have thus become essential to facilitate autonomous control, enabling use cases such as QoT estimation, traffic forecasting, fault prediction, anomaly detection, alarm correlation, and dynamic resource allocation, as discussed in [144]. However, most of these AI/ML applications rely on narrowly scoped models tailored for specific tasks, typically deployed in isolation within an MLOps framework [145]. While effective for localized optimization, such siloed deployment models are inherently limited when addressing end-to-end, cross-domain scenarios that require cooperative decision-making across multiple layers and subsystems.

To address this limitation, the paradigm of Compositional Machine Learning (CML) has been proposed as a strategy for modularly assembling pre-trained ML models or services into composite AI/ML workflows capable of solving complex, multi-faceted problems [146]. In the context of optical networks, such a paradigm holds particular promise for enabling adaptive, intent-driven control across tasks that exhibit temporal and spatial interdependencies. However, several challenges currently hinder the practical adoption of CML in real-world network environments. These include the need to automate the decomposition and distribution of ML models across heterogeneous MLOps infrastructures; the absence of dynamic orchestration mechanisms for stitching ML models into executable service chains; the lack of mechanisms to detect and respond to model drift at runtime; and the absence of programmable interfaces to enable fine-grained reconfiguration of ML pipelines in response to changing control objectives.

Motivated by these gaps, we introduce the first implementation of a Compositional Machine Learning Framework (CMLF) that enables dynamic stitching, drift awareness, and runtime programmability of ML service chains in the control of open and disaggregated optical networks. The proposed framework provides a unified environment where ML models, exposed as services through heterogeneous MLOps platforms, can be continuously discovered and organized into metadata-based graphs. These graphs are then used by a dedicated Stitch Engine to construct valid Machine Learning Service Chains (MLSCs) that fulfill user-specified intents. The Workflow Execution Engine (WFEE) manages the orchestration and execution of these MLSCs at runtime, while the MLOps Adapter layer ensures interoperability across different cloud-native ML infrastructures. Importantly, CMLF also monitors the performance of each ML component during runtime, enabling automated handling of drift through retraining, fallback models, or pipeline reconfiguration.

The applicability of CMLF is demonstrated through its integration in a dynamic Routing, Modulation, and Spectrum Allocation (RMSA) use case for Elastic Optical Networks (EONs), where a forecasting model is chained with a DRL-based RMSA agent to enable future-aware and fragmentation-sensitive resource allocation. This end-to-end ML pipeline is composed, deployed, and executed entirely through CMLF's orchestration mechanisms, highlighting its ability to operationalize complex ML-driven control logic with minimal manual intervention. Empirical results show measurable improvements in blocking probability, fragmentation levels, and adaptability under concept drift, thereby validating the feasibility and effectiveness of the proposed architecture. To the best of our knowledge, this is the first realization of a programmable, drift-adaptive, and compositional ML control plane for optical networks, offering a scalable foundation for realizing ML-native network automation in next-generation disaggregated infrastructures.

## 10.1 CMLF-Based RMSA: Architecture, Models, and Workflow

The Compositional Machine Learning-based RMSA Framework (see Figure 10.1) is a flexible system made up of distinct sequential modules: a training module, a time-series predictor, a DRL-based RMSA, and a CML-RMSA manager. The training module handles offline training and retraining of Link Utilization and Fragmentation (LUF) prediction. At its core, the CML-RMSA manager controls, orchestrates, and stitches the ML models. During runtime, when receiving an RMSA request from the

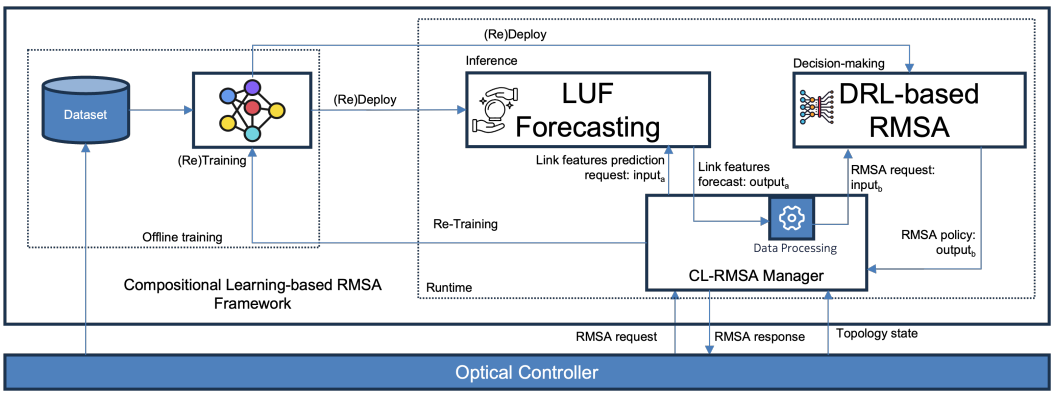



**Fig. 10.1.:** Compositional Machine Learning-based RMSA Framework

optical controller, the CML-RMSA manager sends a prediction request to the LUF forecaster for future link utilization and fragmentation. It assesses the prediction's quality based on model performance metrics. If the model's performance metrics consistently cross thresholds, the CML-RMSA manager initiates retraining for drifted models. Otherwise, the CML-RMSA Manager processes the predicted LUF together with topology information gathered from the optical controller to create the request for the DRL-based RMSA model. The functionality and applicability of Compositional Learning were demonstrated for the first time in [147]. The design of each ML model is presented as follows.

### 10.1.1 Link Features Time-series Forecasting for Enhanced Network State Awareness

#### Problem Definition

Link features forecast is a time-series prediction problem where we intend to foresee the future link features (i.e., link utilization, link defragmentation, etc.) by leveraging historical data. The relation of links can be considered as a graph $G_1 = (V, E)$. $V$ is a set of vertices, where each vertex represents a link of the optical network. Each link is characterized by a set of features (i.e., spectrum utilization, spectrum fragmentation, length, nonlinear impairments...). E is a set of edges, where each edge represents the connection between two links of the optical network (optical network node). In addition, we can construct a fixed adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$.

Given the sequence of observed features of all links $X_{t-m}, X_{t-m+1}, \cdots, X_{t-1}$ ($m$ is the number of historical time steps), we can predict the link features $X_t, X_{t+1}, \cdots, X_{t+p}$

of the next period of $p$ time steps (horizon). The problem of multi-step link features prediction can be described as follows:

$$[X_{t-m+1}, \cdots, X_t, G_1] \xrightarrow{f(\cdot)} [X_{t+1}, \cdots, X_{t+p}] \tag{10.1}$$

where $f(\cdot)$ is the time-series forecasting model to be constructed and $X_t$ is the input link features matrix.

### Spatial-Temporal Link Features Forecasting

Graph Convolutional Networks (GCNs) and Long Short-Term Memory (LSTM) Networks are two different types of neural network architectures that can be applied to time-series forecasting on graph-structured data. GCN is a convolutional neural network with a non-Euclidean structure as the research object, which was proposed by [148].

GCN-LSTM model can capture complex spatial and temporal dependencies and can achieve better forecasting performance than traditional approaches in various scenarios such as traffic prediction [149].

The formal expression of the Graph Convolutional layer is depicted in 10.2:

$$H^{l+1} = ReLU(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^l W^l) \tag{10.2}$$

with $\tilde{A} = A + I$ where $I$ is the identity matrix and $\tilde{D}$ is the diagonal node degree matrix of $\tilde{A}$. $H^l$ and $W^l$ are the graph-level outputs and the weight matrix for the $l$-th neural layer, respectively.

GCN takes as inputs the adjacency matrix, $A$, and the features matrix of links, $X$, in $G_1$. In this work, we consider the link utilization and link fragmentation metrics as features. Feature messages are aggregated between correlated links based on the encoded representation of inputs. We apply the ReLU activation function to output the GCN-encoded feature matrix, which serves as the input of the LSTM block.

Several LSTM layers are stacked to derive the temporal dependencies of link features. LSTMs use three gates that control how the information in a sequence of data comes into, is stored in, and leaves the network: the forget gate, the input gate, and the output gate (denoted as $f_t$, $i_t$, $o_t$, respectively). Each gate operation is described as follows:

$$\langle f, i, o \rangle_t = \sigma(W_{\langle f,i,o \rangle} \cdot [h_{t-1}, x_t] + b_{\langle f,i,o \rangle}) \tag{10.3}$$

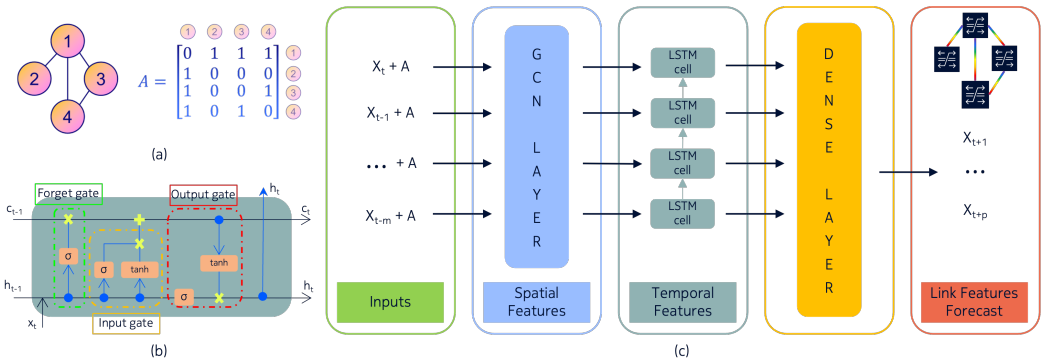



**Fig. 10.2.:** (a) Sample $G_1$ and adjacency matrix; (b) LSTM cell; (c) GCN-LSTM model architecture

where $W_f$, $W_i$, $W_o$ and $b_f$, $b_i$, $b_o$ are the weights and biases of the corresponding gates.

Hidden state $h_t$:

$$h_t = o_t \times tanh(c_t) \tag{10.4}$$

Cell state $c_t$:

$$c_t = f_t \times c_{t-1} + i_t \times tanh(W_c[h_{t-1}, x_t] + b_c) \tag{10.5}$$

A detailed explanation of LSTM can be found in [150].

Finally, a fully connected layer, a.k.a dense layer, outputs the computed tensor; then, it is transposed into appropriate shapes to feed to the DRL agent.

In this work, the input is formed as a series of $m$ timesteps, defined as a window, to forecast the values in $p$ timesteps ahead, called horizon. Depending on the network scenario, the CML-RMSA manager can adjust those parameters to accurately estimate the DRL agent's link features to enrich its network state information. Figure 10.2(c) illustrates the proposed architecture of our GCN-LSTM model.

### 10.1.2 DeepLUF-RMSA

DRL LUF-aware RMSA (DeepLUF-RMSA) is modeled as a Graph $G_2 = (N, L, F)$, where $N$ and $L$ represent the set of nodes and links, and $F = \{F_{l,f} | \forall l \in L, \forall f \in F\}$ contains the state of each Frequency Slot (FS) $f$ on each link $l$. DeepLUF-RMSA learns optimal RMSA policies based on its perception of the current network state (i.e., topology, spectrum state and in-service lightpaths), the forecast link features (i.e., link utilization and fragmentation), and the feedback from the environment (i.e., network operations). At timestep $t$, we model lightpath requests from source,

| **Symbol** | *Description* | **Symbol** | *Description* |
|---|---|---|---|
| $G_1$ | Time-Series forecaster graph | $G_2$ | DRL environment graph |
| $V$ | set of vertices in $G_1$ | $N$ | set of vertices in $G_2$ |
| $E$ | set of edges in $G_1$ | $L$ | set of edges in $G_2$ |
| $A$ | adjacency matrix in $G_1$ | $F$ | set of Frequency Slots of $G_2$ |
| $X$ | feature matrix of links in $G_1$ | $F_l$ | set of Frequency Slots in each link $l$ of $G_2$ |
| $l_{ft}$ | number of link features | $F_{l,i}$ | i-th Frequency Slot of link $l$ in $G_2$ |
| $m$ | historical time steps used for prediction | $F_l^{avail}$ | number of available Frequency Slots in link $l$ of $G_2$ |
| $p$ | horizon | $d_{G_2}^{max}$ | diameter of $G_2$ |
| $LP_t$ | lightpath request at time $t$ | $LP_{t_{hops}}$ | number of hops of the selected path for $LP_t$ |
| $BPSK_C$ | data rate that a Frequency Slot of a BPSK signal can support in Gbps | $LP_{t_{slots}}$ | number of Frequency Slots of the selected candidate Frequency Slot block for $LP_t$ |
| $k$ | number of precomputed path between any pair of nodes | $P_{s,d}$ | path between $s$ and $d$ |
| $J$ | number of computed candidate Frequency Slot blocks per path | $b_{max/min}$ | maximum/minimum capacity demand of any $LP$ throughout simulation |
| $R_{QoA}$ | reward contribution from the quality of the allocation selected by the DRL agent | $R_{NetFrag}$ | reward contribution from network spectrum fragmentation state |
| $H_{frag}$ | Shannon entropy fragmentation metric | $R_{NetAvail}$ | reward contribution from network spectrum availability |

**Tab. 10.1.:** List of notations and symbols

$s$, to destination, $d$, as $LP_t(s, d, b, a, h)$, with capacity demand of $b$ Gb/s and $a, h$ denoting the request arrival time and holding time, respectively. To provision $LP_t$, we need to find a path $P_{s,d}$, determine a proper modulation format $M$, and allocate a number of contiguous and continuous FS's on each link along $P_{s,d}$. DeepLUF-RMSA aims to minimize the long-term blocking probability for a given set of lightpath requests $LP$. The modeling of DeepLUF-RMSA is described as:

1. *Observation:* The observation $\Omega_t$ of DeepLUF-RMSA is an 1-D array of $(2|N| + 2 + (2J + 5) \times k$ elements. It contains the information of $LP_t$ and the spectrum state of $J$ candidate FS's on $k$ paths. The first $2|N| + 2$ elements represent the $s$ and $d$ of $LP_t$ in one-hot encoding, $b$ and $h$. For each path, it computes $J$ candidate free FS blocks, the required number of FS's, the average size of all available FS-blocks, the total number of available FS's, the forecast average link utilization at $t + p$ of path $k$ and the forecast average link fragmentation at $t + p$ of path $k$. Each candidate block is characterized by the FS starting index and its length. From this observation, DeepLUF-RMSA can sense the global current EON state plus an accurate forecast of the future EON state.

2. *Action*: The action space $A$ includes $k \times J + 1$ actions. DeepLUF-RMSA selects for each $LP_t$ to provision one of the $J$ candidate FS blocks on one of the $k$ paths or directly reject it. The number of FS in a block is computed, following the impairment-aware model described in [141], as:

$$LP_{t_{slots}} = \frac{b}{M \cdot BPSK_C} \tag{10.6}$$

   where $M \ \epsilon \ [1, 2, 3, 4, 5, 6]$ corresponds to BPSK, QPSK, 8-QAM, 16-QAM, 32-QAM and 64-QAM modulation shcemes, respectively and $BPSK_C = 37.5Gbps$.

3. *Reward*: To provide detailed information for training our agent efficiently, the DeepLUF-RMSA reward function ($ComplexR_t$) combines three different metrics: network fragmentation metric, network availability metric, and $LP_t$ quality of allocation metric.

$$ComplexR_t = \begin{cases} R_{NetFrag} + R_{NetAvai} + R_{QoA} & \textit{served} \\ -1 & \textit{rejected} \end{cases} \tag{10.7}$$

$$R_{NetFrag} = \frac{\sum_{l=1}^{L} e_l^{-H_{frag}}}{L} \tag{10.8}$$

$$R_{QoA} = \frac{\frac{b_{max}}{BPSK_C} \cdot d_{G_2}^{max} - LP_{t_{hops}} \cdot LP_{t_{slots}}}{\frac{b_{max}}{BPSK_C} \cdot d_G^{max} - \frac{b_{min}}{BPSK_C}} \tag{10.9}$$

$$R_{NetAvai} = \frac{\sum_{l=1}^{L} FS_l^{avai}}{L \cdot FS^{total}} \tag{10.10}$$

The fragmentation metric $R_{NetFrag}$ uses the Path-based Shannon Entropy used in [139] to quantify the level of fragmentation in the network. The network availability metric $R_{NetAvai}$ computes the total number of free FS in $L$. The quality of allocation metric $R_{QoA}$ evaluates the impact that the selected allocation policy has on the path spectrum, by normalizing the total number of slots used by $LP_t$. The combination of these three metrics in $ComplexR$ results in a detailed evaluation of the spectrum status. For evaluation purposes, we also considered a simpler reward function ($SimpleR_t$) that returns $+1$ reward if $LP_t$ is served and -1 otherwise.

4. *Deep Neural Networks (DNNs)*: DeepLUF-RMSA uses a policy DNN $f_{\theta_\pi}(\Omega_t)$ for creating the RMSA policy and a value DNN $f_{\theta_v}(\Omega_t)$ for estimating the discounted cumulative reward of $\Omega_t$. These DNNs share the same fully connected DNN architecture with an input layer of $size(\Omega_t)$ neurons, five hidden layers of 128 neurons but different output layers. The output layer of $f_{\theta_\pi}(\Omega_t)$ consists of $k \times J$ neurons, while $f_{\theta_v}(\Omega_t)$ has a single output neuron.

### 10.1.3 Architectural Design and Workflow of the CMLF for RMSA

This work introduces a novel architectural design that implements Compositional Machine Learning through seamless interaction among three key components: (i) the CML 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, telcos, and cloud operators.

To streamline the development and execution of CML-related applications within the control and management plane, a modular workflow-based Compositional Machine Learning Framework (CMLF) has been created based on the micro-service architecture. This framework is designed to address the challenges associated with CML and serves as a foundation for intent-based CML operations.

As shown in Fig.10.1, the framework consists of a CML and ML Pipeline Manager (CPM), a CML Stitching Engine, a CML Workflow Execution Engine, an MLOps Adapter, and a CML Graphical User Interface (GUI). The CPM is responsible for coordinating the entire lifecycle of ML model chains, exposing CML features to be consumed by external entities such as the SDN controller and its applications. The CML Stitching Engine guides and validates the stitching of multiple ML models at design time. The CML Workflow Execution Engine is the run-time engine responsible for executing the CML workflows generated during design time. The 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 CML GUI is designed to enhance the user experience and simplify CML stitching and management. The CML GUI provides a unified portal to enhance the user experience and simplify the CML stitching and management. The CMLF is running on top of the 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 CMLF 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 CMLF is shown in Fig. 10.3. 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 CML 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 Machine Learning Framework through its implementation for the drift-aware CML-based Dynamic Resource Allocation use-case.

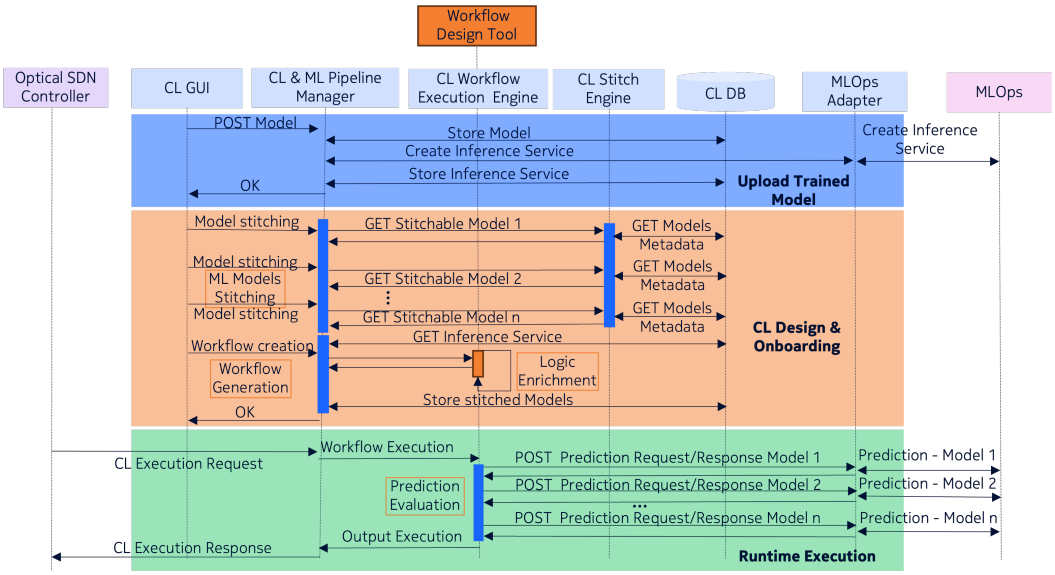



**Fig. 10.3.:** Compositional Machine Learning-based RMSA workflow

## 10.2 CMLF Experimental Setup and Performance Evaluation

The Time-series Forecaster and DeepLUF-RMSA are trained on the CONUS topology (75 nodes and 99 bidirectional links). Each link in both $G_1$ and $G_2$ accommodates 96 frequency slots (FS) of 37.5GHz slot width. The hardware configuration is a server with 20 cores and 64 GB of RAM.

We consider the dataset04 from [151] for the training and evaluation of the Time-series Forecaster. Our GCN-LSTM model is implemented using Tensorflow 2.14 and CUDA 11.1. The Time-series Forecaster consists of one GCN layer, two LSTM layers, and a Dense layer. The model takes a 3-dimensional tensor $(m, |V|, l_{ft})$ as input. The GCN layer has 64 output features. Each LSTM layer has 128 units. The output of the model is a tensor of $(p, |V|, l_{ft})$. The dataset04 was split into 50%, 20%, and 30% of training, validation, and test sets, respectively. We trained the model with 100 epochs and a batch size of 64. After hyperparameter tuning, a window of size $m = 5$ and a forecast horizon of $p = 5$ were selected for the predictor.

In our RMSA environment, we set $k = 9$ and $J = 2$ to prevent our agent from learning a First-Fit (FF) spectrum allocation policy. Hence, the impact of spectrum fragmentation is increased as demonstrated in [152]. To enhance the network's reliability and avoid bottlenecks in higher-degree nodes, we compute $k$-disjoint paths for each $s - d$ pair. We generate dynamic lightpath requests according to the

Poisson process with an arrival rate of 0.5 and a mean lightpath duration of 150 units, which follows the exponential distribution. The $s$ and $d$ randomly selected and $b$ uniformly distributed between [50-300] Gb/s. The DeepLUF-RMSA agent is based on the Proximal Policy Optimization (PPO) framework [101]. Each DNN is updated at the end of an episode using Mean Square Error loss between old and new values of the clipped surrogate loss. The learning rate and discounted factor of PPO were set to $10^{-5}$ and $0.95$, respectively. The simulation was run for 100 episodes. An episode of DeepLUF-RMSA terminates when 5000 requests are handled.

Figure 10.4(a) shows the evolution of the training and validation process of the GCN-LSTM model. The accuracy converges, indicating the good fitting of the model. Figures 10.4(b) and 10.4(c) illustrate the multistep-ahead Mean Squared Error (MSE) of link utilization and link fragmentation, respectively, for each link in $G_1$. The dashed lines correspond to the average MSE of the Baseline and GCN-LSTM models. The Baseline uses the last timestep as the input to forecast the 5-step ahead link features, while the GCN-LSTM uses a window with the 5 last timesteps. Results show that GCN-LSTM reduces the average link utilization and link fragmentation MSE by 25% and 42%, respectively, as compared to the Baseline. Additionally, we observe that the average prediction error of the link fragmentation is higher than the one of link utilization. This is due to the exponential factor of the Shannon entropy formula, which is used to compute the link fragmentation metric.

Once the GCN-LSTM is trained, it is stitched with DeepLUF-RMSA to form GCN-LSTM+DeepLUF-RMSA. Figure 10.4(d) shows the evolution of lightpath blocking probability (BP) during the simulation using kSP-FF, DeepRMSA and GCN-LSTM+DeepLUF-RMSA. GCN-LSTM+DeepLUF-RMSA outperforms kSP-FF and DeepRMSA by reducing the BP by 51.1% and 17.7%, respectively. We evaluate the impact of using the GCN-LSTM predictor. By integrating the time-series prediction, we outperform DeepRMSA by 12.45% in ComplexR and by 13.8% in SimpleR. This proves that the composition of the time-series predictor with the DRL model is more efficient, as the agent is provided with accurate knowledge about the future, producing better RMSA policies. In addition, we examine the effect of using a more sophisticated reward function to guide the agent toward learning LUF-aware RMSA policies. By using the proposed ComplexR, the agent is able to improve the BP by 6.1% in DeepRMSA and by 4.5% in GCN-LSTM+DeepLUF-RMSA, corroborating that our reward function correctly evaluates the status of the spectrum and the quality of allocation. Finally, we evaluate the impact of applying CML as compared to a complex single ML model. GCN-LSTM+DeepLUF-RMSA-SimpleR reduces the BP by 8.3% with respect to DeepRMSA-ComplexR. Also, our approach achieves lower

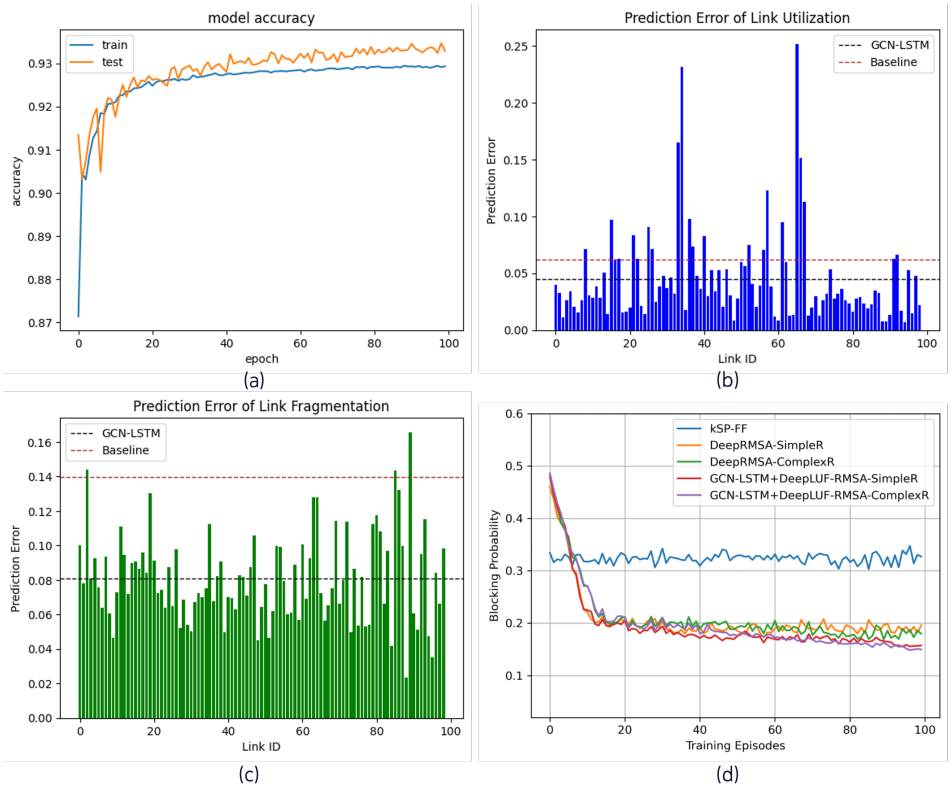



**Fig. 10.4.:** (a) GCN-LSTM model accuracy on train and test data; (b) average link utilization prediction error; (c) average link fragmentation prediction error; (d) lightpath request blocking probability

BP with less number of training episodes. These results demonstrate the benefit of applying CML to decompose complex ML tasks.

Beyond improvements in blocking probability, our compositional machine learning approach also leads to a substantial reduction in training time. By decoupling temporal forecasting from decision-making, the GCN-LSTM+DeepLUF-RMSA architecture allows the reinforcement learning agent to start from a more informed state, avoiding the need to implicitly learn traffic trends through exploration. As a result, the agent reaches the same level of blocking probability performance achieved by the baseline DRL model (DeepRMSA) at episode 100 in just 65 episodes. This represents a 35% reduction in training episodes required to attain equivalent service performance. This gain in sample efficiency not only reduces computational cost and energy consumption during training, but also accelerates the design and deployment cycles of ML-based control agents in dynamic networking environments.

## 10.3 CMLF RMSA Demo Scenario Description

In addition, this work has been demonstrated using real optical hardware, in order to showcase the validity of all the benchmarking and experimental results.

The demonstration was performed live on our remote physical workbench. We use the Cloud-native Optical Transport Control Platform in [153] to manage a network of four Nokia 1830PSS-32 [78] nodes (see Fig. 10.5a). Furthermore, the platform is extended with the CMLF and two MLOps platforms. Each ML model has been trained, served, and monitored in its dedicated ML pipeline prior to the demo. The ML Pipeline Placement Engine in CMLF orchestrates those pipelines on MLOps platforms.

The demo is described as follows:

**Step 1:** *Design and Onboarding phase* - The user navigates through the ML model catalog on the CML GUI. In this demonstration, the user selects the RMSA Deep Reinforcement Learning (DRL) model as the main actor to solve the RMSA problem. The DRL agent performs online learning with the network state provided by the SDN Controller (SDNC) using RLOps principles [154].

Furthermore, to enhance overall performance, the CPM recommends combining the selected model with others already introduced and validated in previous sections or chapters, thus promoting model reuse and modular integration. Specifically, the Link Utilization and Fragmentation (LUF) Forecaster and the DeepLUF-RMSA agent—developed in Section 10.1.1 of this chapter—are proposed to be composed with the Quality of Transmission (QoT) Classifier, which was presented in Chapter 6.4. Together, these models form a composite ML service chain that enhances the RMSA decision-making process by incorporating both forecasted link conditions and QoT estimation into the control loop. The user selects the desired components, defines their composition, and submits the configuration to the CPM. Once received, the CPM triggers the corresponding ML pipelines on each MLOps platform and orchestrates the deployment of the resulting inference services into the Cloud-native Optical Control Platform.

**Step 2:** *Inference phase* - The user creates a T-API 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 output of the LUF forecaster 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

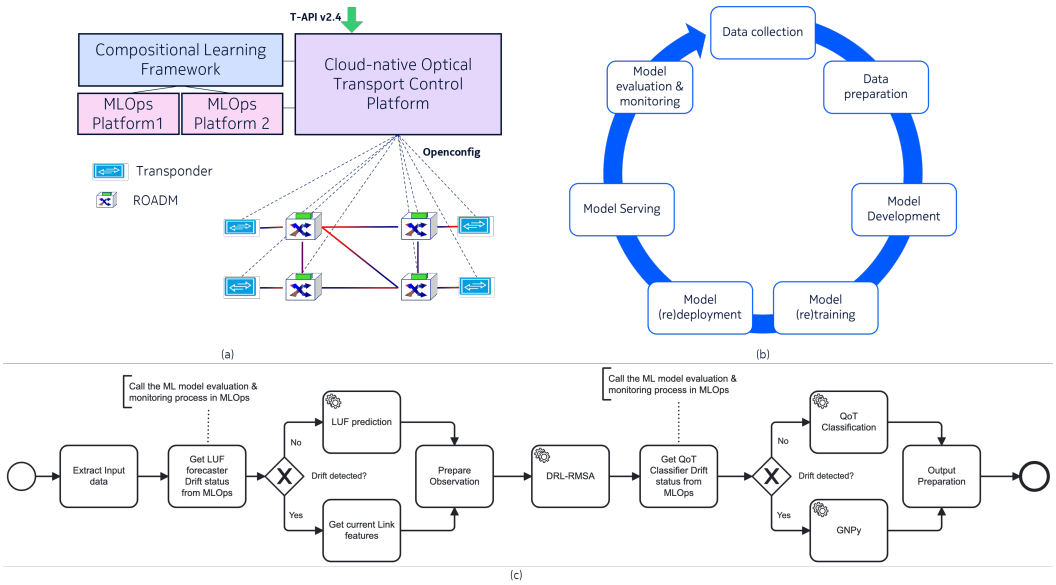



**Fig. 10.5.:** (a) Optical Control Platform and Testbed; (b) MLOps pipeline ; (c) CL-based resource allocation workflow

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 RMSA 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. 10.5c, where a task or sub-task can correspond to a ML model chain inference service.

**Step 3:** *Monitoring phase* - We continue to serve more requests and monitor models' performance. All monitoring data of models is 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 [155]. In the meantime, the CPM updates the composite ML service chain to let the SDNC use GNPy [156] as an alternative solution for QoT validation.

**Step 4:** *Recovery phase* - The ML pipeline, to recover the QoT Classifier performance, executes 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. 10.5b). 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.

The implementation of the proposed Compositional Machine Learning Framework (CMLF) on a real optical testbed using Nokia 1830 PSS-32 equipment validates its practical feasibility in live network conditions. The framework successfully managed end-to-end ML service chains deployed across distributed MLOps platforms, enabling real-time automation of RMSA and QoT estimation tasks within a cloud-native SDN-controlled environment. Throughout the demonstration, all service provisioning and equipment configuration operations were executed reliably, with no observed failures or disruptions. This reinforces the operational robustness of the proposed solution. Nevertheless, further validation at larger network scales is necessary to fully assess its scalability, generalizability, and long-term resilience under production-grade traffic and control workloads.

# 11 Conclusion

Part III has introduced a novel Compositional Machine Learning (CML) framework for elastic optical networks, specifically addressing the challenges of dynamic Routing, Modulation, and Spectrum Assignment (RMSA) through the intelligent orchestration of modular ML models. Building upon prior chapters that established the foundations of disaggregated, cloud-native, and microservice-based optical network control, this work advances the integration of ML by proposing an architecture that supports the design, deployment, and lifecycle management of chained ML models to form composite inference services. By leveraging forecasted network conditions via time-series prediction and coupling them with deep reinforcement learning agents, the proposed system enables more informed and future-aware decision-making, outperforming conventional solutions both in terms of blocking probability and training efficiency.

The key contribution of this chapter lies in the introduction of a fully modular and reusable ML architecture specifically tailored for optical transport networks. For the first time in the literature, complex control tasks are decomposed into semantically meaningful ML components that can be independently trained, monitored, reused, and composed into inference pipelines. The implementation of a GCN-LSTM-based predictor for link utilization and fragmentation, stitched with a DRL-based RMSA agent (DeepLUF-RMSA), illustrates the value of this compositional approach: it internalizes both current and future network states, significantly improving decision quality and reducing learning time. The result is a 51.1% and 17.7% reduction in blocking probability compared to kSP-FF and DeepRMSA, respectively, while also achieving faster convergence in training. This evidences the operational superiority of modular ML architectures over monolithic learning systems.

The architectural framework underpinning this contribution—the Compositional Machine Learning Framework (CMLF)—integrates SDN-based optical controllers, containerized microservices, and MLOps platforms into a unified deployment and management environment. CMLF enables version-controlled composition, lifecycle automation, and seamless integration of heterogeneous models, making it the first practical realization of AI-native orchestration in optical control planes. The validation of this architecture in a real-world testbed using commercial equipment and

cross-platform MLOps integration attests to its technical maturity and deployment feasibility.

Beyond RMSA, the extensibility of CMLF supports future integration of ML models for a broad spectrum of tasks, including QoT estimation, fault localization, transponder optimization, and energy-aware service routing. This modularity paves the way toward an open ecosystem of interoperable ML services that can be dynamically assembled and adapted to evolving network conditions. It positions CML as a scalable design paradigm, not limited to one use case, but foundational to the broader shift toward intelligent, closed-loop control in open optical environments.

Crucially, this research elevates optical control systems toward **Level 4/5 autonomy** as defined by the TM Forum Autonomous Networks framework. By enabling predictive, self-optimizing, and context-aware orchestration of ML modules—with minimal human oversight—CMLF represents a key milestone toward the realization of fully autonomous optical networks. Its support for dynamic composition, intent realization, and service-level adaptation marks a departure from static or offline ML approaches toward truly operational, AI-native infrastructure.

Nevertheless, several challenges must be addressed to further this line of work: model interoperability, error propagation across chained inference tasks, synchronization of real-time data streams, and the need for more advanced orchestration strategies to govern multi-model pipelines. These areas remain open for future investigation, particularly as compositional ML systems are scaled in production environments.

In conclusion, this part presents the first end-to-end architectural and algorithmic realization of compositional learning in optical networks. It bridges the gap between ML research and operational SDN control, offering a concrete, extensible, and intelligent framework capable of adapting to the complexity and dynamism of modern transport infrastructures. As such, it constitutes a foundational contribution to the advancement of AI-native optical control and paves the way for scalable, modular, and autonomous network management in future programmable infrastructures.

# Part IV

Conclusion and Future Works

## 11.1 Synthesis and Final Remarks

This thesis has followed a systematic approach to reconsider the architecture, resource control, service management and operational logic of optical network domains by proposing, integrating and evaluating software-defined networking (SDN), cloud-native orchestration, and machine learning-based decision-making. The core contributions of this breakthrough research have spanned across network function disaggregation and composition, automation pipelines, predictive intelligence, and composable machine learning-native control loops, all underpinned by the operational principles of GitOps and Network Management as Code (NMaC).

At the heart of this work is the concept of disaggregation—not yet fully adopted in operational environments, but increasingly seen as a future-proof design principle. Today's optical network systems are built around vendor-specific platforms where control logic, telemetry, configuration, and service lifecycle management remain vertically integrated. This monolithic model limits the potential for interoperability, service agility, and AI-driven optimization. By leveraging open, standard data models such as OpenConfig and T-API, and interoperable control interfaces like NETCONF, RESTCONF, and gRPC, my research demonstrates how network functions can be exposed as programmable components. This exposes new opportunities for abstracting the physical topology and defining a unified operational state that supports any optical system orchestration and model-driven automation across domains.

To operationalize such programmable environments, this thesis introduces a GitOps-based NMaC pipeline that treats infrastructure as declarative code. Configuration artifacts, service policies, and control applications are maintained in version-controlled repositories and automatically synchronized with the live network state through continuous deployment mechanisms, using tools such as ArgoCD and Kubernetes-native orchestrators. This approach enables reproducibility, traceability, and error-resilient rollouts of optical control services. Microservice decomposition, containerized packaging, and composable Helm charts form the building blocks of a scalable deployment strategy that supports the dynamic instantiation and management of optical network functions.

Building on this programmable infrastructure, the research explores the integration of machine learning (ML) to augment traditional control loops with adaptive and predictive capabilities. While optical transport networks are relatively static, they operate under highly dynamic service demands, varying signal impairments, and evolving resource utilization. These conditions require control mechanisms that can

learn from historical patterns, forecast future states, and respond to anomalies in near real-time. My work integrates ML models—including time-series forecasting, supervised classification, and deep reinforcement learning (DRL) class agents in the control architecture.

These models serve to enhance the control plane's ability to anticipate and act. Forecasting models are trained to predict traffic trends, link fragmentation, and capacity bottlenecks. Supervised learning models enable real-time quality-of-transmission (QoT) estimation, eliminating the need for exhaustive simulations or field trials. DRL agents are trained to learn policies for resource allocation (e.g., routing, modulation, and spectrum assignment) that adapt to both current and forecasted network conditions. This thesis presents a coherent framework for integrating these models into composable pipelines that interact with telemetry feeds, topology views, and service orchestration interfaces.

A key innovation of this thesis lies in the lifecycle management and composability of ML models. Rather than designing ML applications as monolithic and tightly coupled, the proposed architecture modularizes them into reusable components such as predictors, estimators, and policy learners. This modularity enables independent training, replacement, and reuse across different use cases and topologies. Standard interfaces and data schemas ensure compatibility and facilitate the integration of models in production workflows. The ML lifecycle—from training and validation to deployment and inference monitoring—is managed using MLOps tools such as MLflow and Kubeflow, integrated into the GitOps workflow for consistent automation.

Despite these advances, several technical challenges persist. One of the most pressing limitations is the scarcity of large-scale, labeled, and high-quality datasets in optical networks. While synthetic data supports prototyping, it often lacks the variability and physical-layer realism of real-world deployments. Additionally, critical failure events or degradation patterns are rare, making supervised learning difficult to scale. Another concern is model generalization and transferability. Models trained on data from a particular network topology or hardware configuration often struggle to generalize and may exhibit reduced performance when applied to other environments with different structural or operational characteristics.

The trustworthiness and explainability of ML models are also critical. Decisions involving lightpath establishment, transponder configurations, or failure recovery require high levels of reliability. However, black-box models, particularly deep neural networks, lack interpretability, making them difficult to validate or troubleshoot in production. There is a growing need for eXplainable AI (XAI) methods adapted to the

optical network, capable of attributing model outputs to physical-layer conditions, historical patterns, or service-level objectives.

From an architectural standpoint, inference latency and availability are fundamental barriers. Some control decisions, especially those related to fault mitigation or dynamic provisioning, require fast responses. Hosting ML inference engines as centralized services may introduce delays and single points of failure. Exploring edge inference strategies, model quantization, or even in-network ML processing may help meet these constraints.

Finally, this work highlights the complexity of coordinating ML-driven decision-making across multiple network layers and domains. While some abstraction mechanisms exist—such as those offered by hierarchical SDN controllers—fully coordinated policy learning and enforcement across IP and optical domains remains an open problem. Conflicts in policy objectives, mismatch in telemetry granularity, and lack of unified reward functions all contribute to the fragmentation of ML deployment. Addressing these issues will be essential for building holistic, autonomous control frameworks.

Through a set of proofs-of-concept, this thesis has demonstrated both the feasibility and the trade-offs of embedding ML-driven intelligence within an automated, programmable optical network stack. By restructuring network management into modular, composable, and declarative components and by enabling predictive learning agents to participate in decision-making loops, this research contributes toward the long-term vision of fully autonomous, AI-native optical transport systems. However, realizing this vision at scale will require continued research in the areas of data quality, model robustness, system interoperability, and end-to-end orchestration across diverse technological domains.

## 11.2 Paving the Way to AI-Native and Self-Driven Optical Networks

While this thesis establishes a foundational framework for machine learning-native, composable, and automated control of open and disaggregated optical networks, several research directions remain open to fully realize the vision of autonomous, self-managed, and energy-efficient optical infrastructures. These open challenges span the domains of distributed systems, machine learning, and network engineering, and call for continued advances in architectural design, algorithmic robustness, interoperability across layers, and the development of trustworthy AI-driven control mechanisms.

A major research challenge lies in the integration and coordination of traditionally independent control planes—most notably, the IP and optical layers. While SDN-based controllers and telemetry infrastructures have been widely adopted in both domains, their deployment often remains siloed, with limited cross-layer visibility or policy synchronization. As a result, IP and optical controllers operate based on separate and incomplete views of the network state, which leads to globally suboptimal routing decisions, delayed convergence during topology changes, and inefficient resource utilization. Enabling unified, cross-layer control frameworks that abstract and align state information between the IP and optical layers is seen as a necessary evolution. However, this integration also raises significant concerns regarding scalability and abstraction fidelity.

As multi-layer networks grow in size and heterogeneity—with increasing numbers of devices, disaggregated functional roles, diverse technologies, and complex service-level objectives—the volume and rate of control-plane information that must be exchanged across layers increases significantly. This growth introduces serious challenges for real-time decision-making, synchronization, and control-loop stability. Furthermore, the effort to coordinate decisions across layers is not always aligned with the architectural goals of disaggregation. While disaggregation promotes modularity, openness, and the separation of concerns, multi-layer integration tends to reintroduce tight coupling across protocol stacks. This creates a trade off between operational abstraction and architectural decoupling. Although standardization bodies have proposed models for cross-layer coordination, these abstractions are still incomplete and are often insufficient to support granular, real-time orchestration in production-grade IP-over-optical systems. Future research should aim to reconcile this trade-off by developing scalable, modular, and intent-aware control

architectures that enable cross-layer coordination without violating the principles of disaggregation.

In this context, predictive intelligence offers a promising way to improve coordination without relying on constant control-plane signaling. Future transport network architectures must support real-time, bidirectional feedback loops and shared intent abstractions between routing and switching layers. IP routers, which continuously monitor traffic entering the client interfaces of optical transponders, can act as upstream sensors for traffic shifts. Historical analysis and time-series forecasting applied at the IP layer can be used to infer forthcoming trends in traffic demand, enabling proactive resource optimization at the optical layer. This thesis has explored one such mechanism for energy-aware control of Traffic Engineering Links (TE-Links), where low-utilization optical Trails can be temporarily deactivated based on predicted traffic patterns. While the implementation details remain confidential, early results demonstrate significant potential for reducing power consumption by minimizing the number of active line cards during off-peak periods.

In addition, the use of transfer learning across network layers shows promise in addressing data sparsity and domain heterogeneity. Forecasting models trained on the IP layer—which often provides more abundant and reliable data—can be adapted to support decisions in the optical layer, where data is sparser. Altogether, these advances contribute toward a longer-term vision of predictive, energy-aware, and self-optimizing IP-over-WDM networks—capable of operating with minimal human intervention while respecting both architectural modularity and operational scalability.

In parallel, the ability to ensure resilience and recoverability is foundational to any AI-native transport network infrastructure. A promising research direction in this domain involves leveraging the State of Polarization (SoP) as a passive and continuous signal for physical-layer integrity monitoring. Coherent optical receivers can track SoP fluctuations at high frequency, which are often indicative of environmental disturbances, fiber stress, or impending failures. When processed using advanced signal analytic and deep anomaly detection models, SoP trajectories can reveal early-warning signatures of events such as connector misalignment, micro-bending, or fiber cuts—without the need for active probing or invasive instrumentation. Future work should explore scalable representations for SoP dynamics (e.g., spectral fingerprints or quaternion-based encodings), as well as methods for uncertainty quantification and root cause localization. Importantly, to realize network-wide failure prediction capabilities, SoP telemetry must be standardized across different coherent transceiver technologies and made accessible through inter-operable APIs.

These models could then be integrated into the compositional Machine-Learning framework, dynamically stitched into failure mitigation pipelines that include actions such as rerouting or proactive transponder reconfiguration.

Building upon telemetry and prediction mechanisms, the next step is to enable self-healing network architectures in which faults are not only detected, but also automatically diagnosed and mitigated without human intervention. Achieving this vision requires the integration of anomaly detection, root cause analysis, impact forecasting, and policy-driven mitigation models into closed-loop control workflows, where each component can be dynamically discovered, orchestrated, and executed as part of an end-to-end autonomous response. These capabilities can be embedded in a Machine-Learning service marketplace, enabling context-specific compositions based on the severity, location, and impact of the detected anomaly. For instance, a composite pipeline could be invoked in response to a suspected amplifier degradation to forecast its Quality-of-Transmission (QoT) impact, validate alternative paths, and trigger an automated reconfiguration. However, designing such closed loops at scale raises open challenges in multi-agent coordination, policy conflict resolution, and stability under uncertainty—particularly in heterogeneous and multi-technology environments.

The effective operation of these intelligent workflows relies on Machine-Learning models that are both adaptive and trustworthy over time. Optical networks are subject to various forms of concept drift due to aging hardware, environmental changes, and evolving traffic dynamics. Future research must focus on robust drift detection and adaptation mechanisms, such as online learning, active learning, and real-time retraining. While this thesis introduced an MLOps-based approach to manage the machine learning lifecycle—including model versioning, deployment, and monitoring—further research is needed to support context-aware retraining, continuous performance evaluation, and fine-grained telemetry-driven triggering mechanisms. These capabilities are essential to ensure that deployed models remain accurate, interpretable, and robust over time, even when network conditions evolve.

Moreover, as AI-native control expands into multi-domain and multi-operator environments, collaborative learning mechanisms become increasingly important. Privacy requirements, regulatory constraints, and the need to preserve data locality often prevent centralized aggregation of telemetry data. In this context, federated learning emerges as a promising approach, allowing the training of shared models across distributed datasets without transferring raw data. When combined with domain adaptation techniques, such models can better accommodate variations in

network topology, device characteristics, and operational contexts. This enables broader generalization while maintaining data ownership, thus supporting the development of distributed and coordinated intelligence across diverse network domains.

In this context, large language models (LLMs) and generative AI (Gen-AI) offer a complementary pathway to enhance automation at the semantic level. Unlike domain-specific models that operate on structured numerical inputs, LLMs can learn, understand and generate unstructured textual and configuration data. Their potential applications in optical networking control include the automatic generation of network configurations, translation of high-level intents into structured YANG descriptors, and real-time assistance in failure diagnosis or root cause analysis. Furthermore, fine-tuned LLMs can act as explainability agents, generating natural-language justifications for ML-driven control decisions or summarizing system behavior for human operators. Future research may explore the safe delegation of control tasks to generative agents, governed by execution policies and constraint-aware planning, while ensuring auditability and compliance in mission-critical settings.

Finally, realizing a truly AI-native transport infrastructure requires extending intelligent control beyond the optical layer to span the entire network stack—from access to metro and core domains, and from service provisioning to assurance. This necessitates the evolution of compositional learning frameworks into cross-layer and multi-domain control systems that manage not only machine learning inference, but also associated data pipelines, policy enforcement mechanisms, and observability functions. Designing such architectures presents several open challenges, including distributed orchestration, multi-objective optimization, and system-wide fault tolerance. At the same time, it offers the opportunity to fundamentally rethink how networks are operated and optimized. As the complexity and scale of modern networks continue to grow, integrating intelligence, programmability, and automation into a unified control unit becomes not only technically feasible but essential for sustaining future service demands and operational efficiency.

# Bibliography


[1] Govind P. Agrawal. *Fiber-Optic Communication Systems*. 4th ed. Foundational reference on WDM/DWDM technologies and applications. Hoboken, NJ: Wiley, 2012 (cit. on p. 15).

[2] Bell. *Building a More Resilient Optical Network: How OTN-switching and Mesh Restoration are Adding Value to Wavelength Services*. White Paper. 2016 (cit. on p. 15).

[3] Luis Velasco, Ramon Casellas, Katerina Gómez, et al. "An Architecture to Support Autonomic Sliceable Transceiver White Boxes in Disaggregated Optical Networks". In: *Journal of Optical Communications and Networking* 10.10 (2018), pp. D84–D99 (cit. on p. 16).

[4] Open Networking Foundation. *TR-521: SDN Architecture, Issue 1.1*. Tech. rep. ONF, 2016 (cit. on p. 16).

[5] Open ROADM MSA. *Open ROADM MSA: Optical Network and Device Models, Release 7.0*. `https://www.openroadm.org`. Model-driven multi-vendor ROADM/device specifications. 2020 (cit. on pp. 16, 22, 43).

[6] ZTE Corporation. *Intelligent EOTN Solution: Building the Intelligent Elastic Optical Transport Network*. `https://www.zte.com.cn`. White paper. 2020 (cit. on p. 17).

[7] Francesca Paolucci, Filippo Cugini, Nicola Sambo, and Piero Castoldi. "Toward SDN Telemetry in Coherent Optical Networks". In: *Journal of Lightwave Technology* 36.7 (2018), pp. 1571–1579 (cit. on p. 18).

[8] João Pedro, M. Svaluto Moreolo, Filippo Cugini, and Francesca Paolucci. "Quality of Transmission Estimation for Beyond-100G Elastic Optical Networks". In: *IEEE/OSA Journal of Optical Communications and Networking* 12.10 (2020), pp. C1–C14 (cit. on p. 18).

[9] Vittorio Curri, Filippo Cugini, Nicola Sambo, Francesca Paolucci, and Piero Castoldi. "Software-Defined Optical Networks (SDONs): A Comprehensive Survey". In: *IEEE Communications Surveys & Tutorials* 22.4 (2020), pp. 3309–3349 (cit. on p. 18).

[10] Achim Autenrieth, Jörg Peter Elbers, Thomas Szyrkowiec, Pawel Kaczmarek, and Wolfgang Kellerer. "Optical Network Programmability—Requirements and Applications". In: *arXiv preprint arXiv:1801.09563* (2018). Survey of requirements for open and programmable optical control architectures (cit. on p. 18).

[11] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman. *Network Configuration Protocol (NETCONF)*. RFC 6241. 2011 (cit. on pp. 18, 21).

[12] A. Bierer, M. Bjorklund, K. Watsen, and P. Shafer. *RESTCONF Protocol*. RFC 8040. 2017 (cit. on pp. 18, 21).

[13] ZTE Corporation. *ZTE xHaul Transport: Disaggregated and Programmable Optical Line Systems*. https://www.zte.com.cn. White paper. 2022 (cit. on p. 18).

[14] TM Forum. *Autonomous Networks: Levels and Framework (Version 2.0)*. https://www.tmforum.org. Defines Levels 0–5 for autonomous networks. 2022 (cit. on p. 19).

[15] Martin Rowe. *Optical Networks Gear Up for 5G*. https://www.5gtechnologyworld.com/optical-networks-gear-up-for-5g/. Discusses the increased flexibility demands placed on optical networks by 5G. 2020 (cit. on p. 19).

[16] Z. Zakrzewski. "Optical Technologies Supporting 5G/6G Mobile Networks". In: *Photonics* 11.9 (2024). Highlights reliance on proprietary optical systems and the need for dynamic, programmable solutions, p. 833 (cit. on p. 19).

[17] Cisco Systems. *Cisco Global Cloud Index (GCI): Forecast and Methodology*. https://www.cisco.com. Global IP traffic projections. 2022 (cit. on p. 20).

[18] ZTE Corporation. *Intelligent Elastic Optical Transport Networks: Concepts and Practices*. https://www.zte.com.cn. White paper on intelligent EOTN. 2020 (cit. on p. 20).

[19] Peter J. Winzer. "Scaling Optical Fiber Capacity: A Decathlon of Multiplexing Options". In: *Journal of Lightwave Technology* 35.5 (2017), pp. 1099–1115 (cit. on p. 20).

[20] Vittorio Curri et al. "A Hybrid Flexible Grid Architecture for Next-Generation Optical Networks". In: *IEEE/OSA Journal of Optical Communications and Networking* 8.8 (2016), B1–B10 (cit. on p. 20).

[21] G. Kamalov and et al. "Evolution to Open Line Systems in Optical Transport Networks". In: *Proc. Optical Fiber Communication Conference (OFC)*. 2017, W1I.1 (cit. on p. 20).

[22] Bruno Parreira, Víctor López, et al. "Disaggregated Optical Networks: Models, Interfaces, and Control". In: *IEEE Communications Magazine* 61.8 (2023), pp. 88–95 (cit. on pp. 20, 21).

[23] Abhishek Kumar, Jayant Kolhe, Sanjay Ghemawat, and Louis Ryan. *gRPC Protocol*. Internet-Draft draft-kumar-rtgwg-grpc-protocol-00. Work in Progress. Internet Engineering Task Force (IETF), 2016 (cit. on p. 21).

[24] Ingo Lütkebohle. *BWorld Robot Control Software*. https://www.openconfig.net/. [Online; accessed 19-July-2008]. 2020 (cit. on p. 22).

[25] Open Networking Foundation. *TR-547: Transport API (T-API) – Northbound Interface Specification, Version 2.4.0*. Tech. rep. ONF, 2020 (cit. on pp. 22, 39).

[26] Francesca Paolucci, Filippo Cugini, Nicola Sambo, and Piero Castoldi. "A YANG Model and Streaming Telemetry Framework for Optical Networks". In: *IEEE/OSA Journal of Optical Communications and Networking* 10.10 (2018), pp. C100–C112 (cit. on p. 22).

[27] Latif Khan, Boris Bellalta, et al. "Autonomous Networks: Empowering Digital Twins for Smart Societies and Industries". In: *IEEE Communications Magazine* (2023) (cit. on p. 25).

[28] Morteza Mahloo and et al. "Energy-aware optical networks: Challenges and research opportunities". In: *IEEE Network* (2020) (cit. on p. 25).

[29] Y. Liu, S. Guo, and D. Zhang. “A survey of optical network virtualization”. In: *IEEE Communications Surveys and Tutorials* (2020) (cit. on p. 25).

[30] X. Chen and et al. “Next-generation optical access: Technology, standardization, and deployment”. In: *IEEE Communications Magazine* (2019) (cit. on p. 25).

[31] Luis Velasco and et al. “Control and management of flexgrid optical networks”. In: *Proceedings of the IEEE* (2017) (cit. on p. 25).

[32] R. Sharma et al. “Advances in software-defined and virtualized optical networks”. In: *IEEE Journal on Selected Areas in Communications* (2018) (cit. on p. 25).

[33] Y. Luo and F. Zhang. “Disaggregated optical networks: Challenges and opportunities”. In: *Journal of Lightwave Technology* (2019) (cit. on p. 25).

[34] G. Bianchi et al. “Optical transport networks in 5G: Architectural enablers and deployment scenarios”. In: *IEEE Communications Standards Magazine* (2020) (cit. on p. 25).

[35] Jordi Fàbrega and et al. “Multidomain orchestration of optical networks with federated control”. In: *Computer Networks* (2020) (cit. on p. 26).

[36] K. Poularakis and L. Tassiulas. “Programmable optical networks: Are we ready for SDN?” In: *Computer Networks* (2021) (cit. on p. 26).

[37] A. Mohan and et al. “GitOps-based declarative operations for Kubernetes-based infrastructures”. In: *Software: Practice and Experience* (2021) (cit. on p. 26).

[38] Y. Colmant and et al. “Observability pipelines for cloud-native environments”. In: *Proceedings of the ACM Symposium on Cloud Computing*. 2022 (cit. on p. 26).

[39] Len Bass, Ingo Weber, and Liming Zhu. *DevOps: A Software Architect’s Perspective*. Boston, MA: Addison-Wesley Professional, 2015 (cit. on pp. 27, 29).

[40] Michael Hüttermann. *DevOps for Developers*. Berkeley, CA: Apress, 2012 (cit. on p. 27).

[41] Leah Riungu-Kalliosaari, Simo Mäkinen, Lucy Ellen Lwakatare, Juha Tiihonen, and Tomi Männistö. “DevOps Adoption Benefits and Challenges in Practice: A Case Study”. In: *Product-Focused Software Process Improvement (PROFES 2016)*. Vol. 10027. LNCS. Cham: Springer, 2016, pp. 590–597 (cit. on p. 27).

[42] Gene Kim, Jez Humble, Patrick Debois, and John Willis. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. Portland, OR: IT Revolution Press, 2016 (cit. on p. 27).

[43] Floris M. A. Erich, Chintan Amrit, and Maia Daneva. “A Qualitative Study of DevOps Usage in Practice”. In: *Journal of Software: Evolution and Process* 29.6 (2017), e1885 (cit. on p. 27).

[44] Kief Morris. *Infrastructure as Code: Managing Servers in the Cloud*. Sebastopol, CA: O’Reilly Media, 2016 (cit. on p. 27).

[45] Alexander Clemm, Laurent Ciavaglia, Lisandro Granville, and Jeff Tantsura. “Intent-based networking—Concepts and definitions”. In: *IEEE Communications Surveys and Tutorials* 21.1 (2019), pp. 377–396 (cit. on pp. 27, 29).

[46] Felix Biehl, Alexander Clemm, Dirk Trossen, and Dirk Kutscher. “Intent-based networking: Concepts and implementation in software-defined networks”. In: *Journal of Network and Systems Management* 29.1 (2021), pp. 1–25 (cit. on pp. 27, 29).

[47] Brian Fitzgerald and Klaas-Jan Stol. “Continuous Software Engineering: A Roadmap and Agenda”. In: *Journal of Systems and Software* 123 (2017), pp. 176–189 (cit. on p. 28).

[48] Benjamin H. Sigelman, Luiz André Barroso, Mike Burrows, et al. “Dapper, a Large-Scale Distributed Systems Tracing Infrastructure”. In: *Technical Report* (2010) (cit. on p. 28).

[49] Nicole Forsgren, Jez Humble, and Gene Kim. *Accelerate: The Science of Lean Software and DevOps*. Portland, OR: IT Revolution Press, 2018 (cit. on p. 28).

[50] Manfred Broy, Ingolf Krüger, and Christian Salzmann. “Engineering Automotive Software”. In: *Proceedings of the IEEE*. Vol. 95. 2. 2007, pp. 356–373 (cit. on p. 28).

[51] “ETSI GS NFV-MAN 001 V1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration”. In: (Dec. 2014). First MANO specification; widely cited in 2014–2016 NFV MANO literature (cit. on p. 28).

[52] Tal Mizrahi, Yuval Moses, and Ayal Shimonov. “Network as code: Toward programmable network infrastructures”. In: *IEEE Communications Magazine* 59.1 (2021), pp. 74–80 (cit. on p. 29).

[53] Dieter Hock, Richard Cziva, Manuel Peuster, and Stefan Schneider. “Network management as code: Towards agile and reliable service delivery”. In: *IEEE Transactions on Network and Service Management* 18.2 (2021), pp. 2047–2060 (cit. on p. 29).

[54] Andy Bierman, Martin Bjorklund, Kent Watsen, and Juergen Schoenwaelder. “YANG–A data modeling language for the Network Configuration Protocol (NETCONF)”. In: *RFC 6020, IETF* (2010) (cit. on p. 29).

[55] Diego Villa, Daniele Sanvito, and Gianpaolo Cugola. “Towards intent-driven network configuration with network as code”. In: *Proceedings of the ACM SIGCOMM Conference Posters and Demos*. 2020, pp. 77–79 (cit. on p. 30).

[56] Vamsi Jalaparti, Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Antony Rowstron. “Config2Spec: Mining network specifications from network configurations”. In: *USENIX NSDI*. 2018, pp. 289–303 (cit. on p. 30).

[57] Raouf Boutaba, Mohammad A Salahuddin, Nazim Limam, et al. “A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities”. In: *Journal of Internet Services and Applications* 9.1 (2018), pp. 1–99 (cit. on p. 30).

[58] Weaveworks. *What is GitOps?* `https://www.weave.works/technologies/gitops/`. Accessed: 2025-09-02. 2021 (cit. on p. 31).

[59] Argo Project. *Argo CD - Declarative GitOps CD for Kubernetes*. `https://argo-cd.readthedocs.io/en/stable/`. Accessed: 2025-09-02. 2021 (cit. on p. 31).

[60] Flux Project. *Flux - The GitOps Family of Tools*. `https://fluxcd.io`. Accessed: 2025-09-02. 2022 (cit. on p. 31).

[61] Brendan Rosen and Jason Weinstock. "A Study of GitOps in Practice". In: *Journal of Systems and Software* 189 (2022), p. 111304 (cit. on p. 31).

[62] Yifan Liang, Bo Zhang, and Tao Xie. "Securing GitOps: Challenges and Approaches". In: *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*. 2021, pp. 1–11 (cit. on p. 31).

[63] Cloud Native Computing Foundation. *GitOps and Edge Computing: Benefits and Use Cases*. `https://www.cncf.io/blog/2021/06/15/gitops-at-the-edge/`. Accessed: 2025-09-02. 2021 (cit. on p. 32).

[64] Imran Sheikh and Muhammad Javed. "GitOps-driven Continuous Delivery for 5G Core Network Slices". In: *IEEE NFV-SDN*. 2021, pp. 45–50 (cit. on p. 32).

[65] Martin Fowler and James Lewis. "Microservices". In: *martinfowler.com* (2014). `https://martinfowler.com/articles/microservices.html` (cit. on pp. 33, 34).

[66] Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2015 (cit. on pp. 33, 34).

[67] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. "Borg, Omega, and Kubernetes". In: *Communications of the ACM* 59.5 (2016), pp. 50–57 (cit. on pp. 33, 34).

[68] Ryan Whitlock, Takashi Ishimatsu, and Lei Geng. "DevOps for Optical Networks: Challenges and Opportunities". In: *Proceedings of the Optical Fiber Communication Conference (OFC)*. 2021 (cit. on p. 36).

[69] Stefano Salsano, Mattia Polverini, and Domenico Siracusa. "DevOps for Networks: State of the Art and Research Challenges". In: *Proceedings of the 2021 ACM SIGCOMM*. 2021, pp. 240–246 (cit. on p. 36).

[70] Orhan Dagdeviren, Thomas Neumann, and Piotr Gaj. "Bridging DevOps and Network Automation: Experience with Netopeer and YANG-Based Provisioning". In: *2023 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE. 2023, pp. 122–130 (cit. on p. 36).

[71] Ze Liu and Ioannis Tomkos. "Intent-Based Management of Programmable Optical Networks: Architecture, Use Cases and Research Directions". In: *Journal of Optical Communications and Networking* 15.1 (2023), A1–A12 (cit. on p. 36).

[72] Oscar Gonzalez de Dios, Carlos Vaquero, Alberto Castro, and Ruben Viera. "Towards Intent-Based Networking in Disaggregated Transport Networks". In: *IEEE Communications Standards Magazine* 6.1 (2022), pp. 44–50 (cit. on p. 37).

[73] Quan Pham-Van, Huy Tran-Quang, Dominique Verchère, et al. "Demonstration of Container-Based Microservices SDN Control Platform for Open Optical Networks". In: *OFC 2019*. Also known as IEEE document 8696900. IEEE, 2019, pp. 1–3 (cit. on p. 39).

[74] Alexander Clemm, Guido Maier, Carmen Mas Machuca, K. K. Ramakrishnan, et al. “Proceedings of the 8th IEEE Conference on Network Softwarization (NetSoft 2022)”. In: *IEEE NetSoft 2022*. Conference on advances in network softwarization technologies. IEEE, 2022 (cit. on p. 39).

[75] Red Hat. “What is GitOps?” In: *Red Hat DevOps Topics* (2025). Defines GitOps as managing infrastructure and application configurations via Git as a single source of truth (cit. on p. 39).

[76] Scott Chacon and Ben Straub. *Pro Git*. Git reference manual treating Git as the single source of truth. Apress, 2014 (cit. on p. 40).

[77] Argo CD Documentation. *Argo CD: Declarative GitOps Continuous Delivery for Kubernetes*. `https://argo-cd.readthedocs.io/en/stable/`. Describes Argo CD as a declarative GitOps continuous delivery tool for Kubernetes. 2025 (cit. on p. 40).

[78] Nokia. *1830 Photonic Service Switch (PSS) — DWDM Multiservice, Multilayer P-OTN Transport Platform*. Nokia Optical Networks product page. Family of DWDM multiservice, multilayer P-OTN platforms for metro, regional, long-haul and data center interconnect applications. 2025 (cit. on pp. 42, 151).

[79] Ahmed Triki, Christophe Betoule, Gilles Thouenon, et al. “OpenROADM Compliant SDN Controller for Full Interoperability of the Optical Transport Network”. In: *Journal of Optical Communications and Networking* 12.6 (2020). Demonstrates OpenDaylight Transport-PCE relying on OpenDaylight and OpenROADM models for optical SDN control, pp. C24–C31 (cit. on p. 43).

[80] Linux Foundation Networking Developer Forum. *ODL: T-API contribution to T-PCE*. Presentation at LFN Developer and Testing Forum, Jan 12, 2022. Presented by errea and tqhuy812 on contribution of T-API to Transport-PCE. Jan. 2022 (cit. on p. 44).

[81] Linux Foundation Networking Developer Forum. *ODL: Planned extension of T-API module and integration of OpenConfig device models in T-PCE*. Presentation at LFN Developer and Testing Forum, Jan 12, 2022. Presented by errea and tqhuy812 on T-API and OpenConfig integration in Transport-PCE. Jan. 2022 (cit. on p. 44).

[82] Javier Perell, Riccardo Mazzini, Nicola Sambo, Filippo Cugini, and Piero Castoldi. “GNPy: An Open Source Application for Physical Layer Aware Optical Network Planning and Design”. In: *Proceedings of the Optical Fiber Communication Conference (OFC)*. Optica Publishing Group. 2020, p. M3I.6 (cit. on p. 48).

[83] Nicola Sambo, Javier Perell, Filippo Cugini, and Piero Castoldi. “Assessment of Quality of Transmission (QoT) Estimators for Open Optical Network Planning”. In: *Journal of Optical Communications and Networking* 10.10 (2018), pp. D52–D62 (cit. on p. 48).

[84] Telecom Infra Project. *GNPy Open Source Repository*. Available at: `https://github.com/Telecominfraproject/gnpy`. 2023 (cit. on p. 48).

[85] Osvaldo Simeone. “A Very Brief Introduction to Machine Learning With Applications to Communication Systems”. In: *arXiv preprint arXiv:1808.02342* (2018) (cit. on p. 64).

[86] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016 (cit. on p. 64).

[87] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis. “Explainable Artificial Intelligence: A Review of Machine Learning Interpretability Methods”. In: *Entropy* 23.1 (2020), p. 18 (cit. on p. 64).

[88] I. Salehin et al. “AutoML: A systematic review on automated machine learning”. In: *Artificial Intelligence Review* (2024) (cit. on p. 64).

[89] H2O.ai. *What are the Advantages of Automated Machine Learning?* `https://h2o.ai/wiki/automated-machine-learning/`. Accessed online. 2025 (cit. on p. 64).

[90] Oussama Ayoub et al. “Explainable Artificial Intelligence in Communication Networks: State-of-the-Art and Future Directions”. In: *Computer Communications* 193 (2022), pp. 1–17 (cit. on p. 64).

[91] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. Springer, 2009 (cit. on p. 65).

[92] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605 (cit. on p. 66).

[93] Leland McInnes, John Healy, and James Melville. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. In: *arXiv preprint arXiv:1802.03426* (2018) (cit. on p. 66).

[94] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations (ICLR)* (2014) (cit. on p. 66).

[95] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. “Extracting and Composing Robust Features with Denoising Autoencoders”. In: *International Conference on Machine Learning (ICML)* (2008), pp. 1096–1103 (cit. on p. 66).

[96] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *International Conference on Machine Learning (ICML)* (2020) (cit. on p. 67).

[97] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. “Momentum Contrast for Unsupervised Visual Representation Learning”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020) (cit. on p. 67).

[98] Jean-Bastien Grill, Florian Strub, Florent Altché, et al. “Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning”. In: *Advances in Neural Information Processing Systems* 33 (2020) (cit. on p. 67).

[99] Christian Hennig. “What Are the True Clusters? Pattern Recognition and Statistical Clustering”. In: *Pattern Recognition Letters* 64 (2015), pp. 53–62 (cit. on p. 68).

[100] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), pp. 529–533 (cit. on p. 70).

[101] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal Policy Optimization Algorithms”. In: *CoRR* abs/1707.06347 (2017) (cit. on pp. 70, 84, 97, 149).

[102] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 1861–1870 (cit. on p. 70).

[103] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, et al. “DRN: A Deep Reinforcement Learning Framework for News Recommendation”. In: *Proceedings of the 2018 World Wide Web Conference* (2018), pp. 167–176 (cit. on p. 70).

[104] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. “Multi-Agent Deep Reinforcement Learning for Large-scale Traffic Signal Control”. In: *CoRR* abs/1903.04527 (2019) (cit. on p. 70).

[105] Ryan Lowe, Yi Wu, Aviv Tamar, et al. “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments”. In: *Advances in Neural Information Processing Systems*. 2017 (cit. on p. 70).

[106] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, et al. “QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 4295–4304 (cit. on p. 70).

[107] Cristina Rottondi, Lorenzo Barletta, Alessio Giusti, and Massimo Tornatore. “Machine-learning method for quality of transmission prediction of unestablished lightpaths”. In: *IEEE/OSA Journal of Optical Communications and Networking*. Vol. 10. 2. Demonstrates SVM application for QoT classification. 2018, A286–A297 (cit. on p. 75).

[108] Stanisław Kozdrowski, Paweł Cichosz, Piotr Paździewski, and Sławomir Sujecki. “Machine Learning Algorithms for Prediction of the Quality of Transmission in Optical Networks”. In: *Entropy* 23.1 (2020). Random Forests benchmarked for QoT generalization and robustness, p. 7 (cit. on p. 75).

[109] Jesper Wass, Jakob Thrane, Molly Piels, and Rasmus Jones. “Gaussian Process Regression for WDM System Performance Prediction”. In: *Proc. Optical Fiber Communication Conference (OFC) 2017*. Uses Gaussian process regression to predict BER in a 24×28 GBd QPSK WDM system. 2017 (cit. on p. 75).

[110] Francesco Musumeci, Cristina Rottondi, Avishek Nag, et al. “An Overview on Application of Machine Learning Techniques in Optical Networks”. In: *IEEE Communications Surveys & Tutorials* 21.2 (2019). Survey covering DNN usage and hierarchical ML approaches in optical networks, pp. 1383–1408 (cit. on pp. 75, 78).

[111] Uiara C. de Moura, Miquel Garrich Alabarce, Heitor Carvalho, et al. “Cognitive Methodology for Optical Amplifier Gain Adjustment in Dynamic DWDM Networks”. In: *Journal of Lightwave Technology* 34.8 (2016). CBR-based cognitive methodology for EDFA gain control, showing OSNR improvements over time, pp. 1971–1979 (cit. on p. 76).

[112] Francesco Da Ros, Uiara Celine de Moura, and Metodi P. Yankov. “Machine learning-based EDFA Gain Model Generalizable to Multiple Physical Devices”. In: *arXiv preprint* arXiv:2009.05326 (2020). Neural network-based EDFA gain modeling with generalization across physical units (cit. on p. 76).

[113] Shengxiang Zhu, Craig Gutterman, Alan Diaz Montiel, et al. “Hybrid Machine Learning EDFA Model”. In: *OFC 2020*. Hybrid analytical + ML EDFA gain model combining domain knowledge and data-driven learning. 2020 (cit. on p. 77).

[114] Zhiquan Wan, Zhenming Yu, Liang Shu, et al. “Intelligent Optical Performance Monitor Using Multi-Task Learning Based Artificial Neural Network”. In: *arXiv preprint arXiv:1812.03792* (2018). Uses amplitude histograms as input features for MTL-ANN to estimate OSNR and modulation format (cit. on p. 78).

[115] G. O. Ferreira, C. Ravazzi, F. Dabbene, G. Calafiore, and M. Fiore. “Forecasting Network Traffic: A Survey and Tutorial with Open-Source Comparative Evaluation”. In: *IEEE Access* (2023) (cit. on p. 79).

[116] Krzysztof Walkowiak, Daniel Szostak, Adam Włodarczyk, and Andrzej Kasprzak. “Long-Term Traffic Forecasting in Optical Networks Using Machine Learning”. In: *International Journal of Engineering and Technology* (2023) (cit. on p. 79).

[117] Boyi Liu, Xiangyan Tang, Jieren Cheng, and Pengchao Shi. “Traffic Flow Combination Forecasting Method Based on Improved LSTM and ARIMA”. In: Proposes SDLSTM-ARIMA hybrid model with improved forecasting accuracy. 2019 (cit. on p. 79).

[118] Tania Panayiotou, Maria Michalopoulou, and Georgios Ellinas. “Survey on Machine Learning for Traffic-Driven Service Provisioning in Optical Networks”. In: *arXiv preprint arXiv:2209.05080* (2022). Comprehensive review of ML techniques for optical network traffic-driven service provisioning (cit. on p. 79).

[119] Weiwei Jiang and Jiayun Luo. “Graph Neural Network for Traffic Forecasting: A Survey”. In: *arXiv preprint arXiv:2101.11174* (2021). Survey on GNNs for spatial-temporal traffic forecasting (cit. on p. 79).

[120] M. Aygül et al. “Machine Learning-Based Spectrum Occupancy Prediction”. In: *Frontiers in Communications and Networks* (2025) (cit. on p. 80).

[121] Aleksandra Knapinska, Robert Kanimba, Yusuf Yesilyurt, Piotr Lechowicz, and Krzysztof Walkowiak. “Application of Ensemble Regression Methods in Elastic Optical Network Optimization”. In: *Chalmers Research (preprint)*. Ensemble ML for predicting spectrum and resource allocation metrics. 2024 (cit. on p. 80).

[122] S. Kozdrowski. “Machine Learning Algorithms for Prediction of the Quality ...” In: *Entropy* (2020). Random forest and extreme gradient boosting deliver high predictive performance using histogram-based features (cit. on p. 80).

[123] Ehsan Etezadi, Carlos Natalino, Renzo Diaz, et al. “Deep reinforcement learning for proactive spectrum defragmentation in elastic optical networks”. In: *Journal of Optical Communications and Networking* (2023). Presents DeepDefrag, a DRL framework for proactive spectrum defragmentation (cit. on p. 80).

[124] X. Li. “Collective sparse symmetric non-negative matrix factorization”. In: *Brain Structure and Function* (2018). Introduces CSS-NMF method for joint decomposition of related datasets (cit. on p. 80).

[125] E. Dávalos. “Triggering strategy for defragmentation process in Elastic Optical Networks”. In: *Applied Sciences* (2023). ML-based method to estimate blocking rate and trigger defragmentation (cit. on p. 80).

[126] Zhilong Wang, Min Zhang, Danshi Wang, and et al. “Failure Prediction Using Machine Learning and Time Series in Optical Networks”. In: *Optics Express* 25.16 (2017), pp. 18553–18565 (cit. on p. 81).

[127] Danshi Wang, Chunyu Zhang, Wenbin Chen, et al. “A Review of Machine Learning-Based Failure Management in Optical Networks”. In: *Science China Information Sciences* 65.11 (2022), pp. 1–20 (cit. on pp. 81, 82).

[128] Francesco Musumeci, Cristina Rottondi, Avishek Nag, et al. “An Overview on Application of Machine Learning Techniques in Optical Networks”. In: *IEEE/OSA Journal of Optical Communications and Networking* (2018). Includes descriptions of FEELING and TISSUE frameworks using Decision Trees and SVM (cit. on p. 81).

[129] Wenjie Du, David Côté, Chris Barber, and Yan Liu. “Forecasting Loss of Signal in Optical Networks With Machine Learning”. In: *arXiv preprint arXiv:2201.07089* (2022). Regression-based ML model using PM data to predict LOS events (cit. on p. 81).

[130] Ankit Sharma, Vishal Lohani, and Yatindra Nath Singh. “A vectored fragmentation metric for elastic optical networks”. In: *Photonics Network Communications* 45 (2023), pp. 12–24 (cit. on p. 83).

[131] Yulong Zhang, Jianxin Xin, Xuepeng Li, and Shijian Huang. “Overview on Routing and Resource Allocation Based on Machine Learning in Optical Networks”. In: *Optical Fiber Technology* 60 (2020). Surveys ML applications (decision trees, SVM, DNN) for RMSA (cit. on p. 84).

[132] Jie Liu, Feiping Nie, and Heng Huang. “Collective Nonnegative Matrix Factorization for Multi-View Clustering”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*. Introduces Collective NMF, useful for extracting traffic and spectrum usage patterns. 2018, pp. 2577–2583 (cit. on p. 84).

[133] J. Zhang, H. Yu, Y. Zhao, and W. Guo. “Traffic Pattern Clustering and Spectrum Allocation in Elastic Optical Networks”. In: *Journal of Optical Communications and Networking* 9.11 (2017). Applies clustering to traffic matrices for proactive spectrum allocation decisions, pp. D56–D66 (cit. on p. 84).

[134] Xiaoliang Chen, Baojia Li, Roberto Proietti, et al. “DeepRMSA: A Deep Reinforcement Learning Framework for Routing, Modulation and Spectrum Assignment in Elastic Optical Networks”. In: *Journal of Lightwave Technology* 37.16 (2019). DRL (DQN, A2C) applied to RMSA in EONs, pp. 4155–4163 (cit. on pp. 84, 98).

[135] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)* 48 (2016). Introduces A3C, the basis of Advantage Actor-Critic (A2C), pp. 1928–1937 (cit. on p. 84).

[136] M. A. Hady. “Multi-Agent Reinforcement Learning for Resource Allocation Optimization: A Survey”. In: *Artificial Intelligence Review* (2025). Surveys MARL applications in telecommunications, energy systems, and more (cit. on p. 84).

[137] A. Sgambelluri et al. “OpenConfig and OpenROADM Automation of Operational Modes in Disaggregated Optical Networks”. In: *IEEE Access* 8 (2020), pp. 190094–190107 (cit. on p. 90).

[138] I. Khan et al. “Assessment of cross-train machine learning techniques for QoT-estimation in agnostic optical networks”. In: *OSA Continuum* 3.10 (2020), pp. 2690–2706 (cit. on p. 90).

[139] P. Wright, M. C. Parker, and A. Lord. “Simulation results of Shannon entropy based flexgrid routing and spectrum assignment on a real network topology”. In: *39th European Conference and Exhibition on Optical Communication (ECOC)*. Paper presented at ECOC 2013. 2013, pp. 1–3 (cit. on pp. 97, 146).

[140] C. Natalino and P. Monti. “The Optical RL-Gym: An open-source toolkit for applying reinforcement learning in optical networks”. In: *22nd International Conference on Transparent Optical Networks (ICTON)*. 2020, pp. 1–4 (cit. on pp. 98, 107).

[141] B. Kozicki, H. Takara, Y. Sone, A. Watanabe, and M. Jinno. “Distance-adaptive spectrum allocation in elastic optical path network (SLICE) with bit per symbol adjustment”. In: *Optical Fiber Communication Conference (OFC)*. 2010, pp. 1–3 (cit. on pp. 98, 145).

[142] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention Is All You Need”. In: *arXiv preprint arXiv:1706.03762* (2017). Introduced the Transformer architecture for sequence modeling (cit. on pp. 102, 104).

[143] Manuel Baena-Garc, José Avila, Albert Bifet, Ricard Gavald, and Rafael Morales-Bueno. “Early Drift Detection Method”. In: (2005) (cit. on p. 112).

[144] Francesco Musumeci, Cristina Rottondi, Avishek Nag, et al. “An Overview on Application of Machine Learning Techniques in Optical Networks”. In: *IEEE Communications Surveys & Tutorials* 21.2 (2019), pp. 1383–1408 (cit. on p. 139).

[145] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. “Machine Learning Operations (MLOps): Overview, Definition, and Architecture”. In: *arXiv preprint arXiv:2205.02302* (2022). Provides an aggregated overview of MLOps principles, components, and architecture, including single-model deployment practices (cit. on p. 139).

[146] Debmalya Biswas. *Compositional AI: Fusion of AI/ML Services*. Tech. rep. Mar. 2021 (cit. on p. 139).

[147] Javier Errea, Huy Tran Quang, Dominique Verchere, et al. “Open Disaggregated Optical Network Control with Network Management as Code”. In: *2023 Optical Fiber Communications Conference and Exhibition (OFC)*. 2023, pp. 1–3 (cit. on p. 141).

[148] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *International Conference on Learning Representations (ICLR)*. Introduces Graph Convolutional Networks (GCNs) for non-Euclidean graph-structured data. 2017 (cit. on p. 142).

[149] Bing Yu, Haoteng Yin, and Zhanxing Zhu. “Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting”. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)* (2018). Proposes ST-GCN, a GCN-LSTM hybrid for spatio-temporal traffic prediction, pp. 3634–3640 (cit. on p. 142).

[150] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997). Original paper introducing the Long Short-Term Memory (LSTM) neural network architecture, pp. 1735–1780 (cit. on p. 143).

[151] Fraunhofer HHI. *QoT dataset collection*. 2021 (cit. on p. 148).

[152] Javier Errea, Deborah Djon, Huy Quang Tran, Dominique Verchere, and Adlen Ksentini. “Deep Reinforcement Learning-aided Fragmentation-aware RMSA Path Computation Engine for Open Disaggregated Transport Networks”. In: *2023 International Conference on Optical Network Design and Modeling (ONDM)*. 2023, pp. 1–3 (cit. on p. 148).

[153] Javier Errea et al. “Open Disaggregated Optical Network Control with Network Management as Code”. In: *2023 Optical Fiber Communications Conference and Exhibition (OFC)*. 2023, pp. 1–3 (cit. on p. 151).

[154] Peizheng Li et al. “RLOps: Development Life-Cycle of Reinforcement Learning Aided Open RAN”. In: *IEEE Access* (2022) (cit. on p. 151).

[155] 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)*. 2023, pp. 1–4 (cit. on p. 152).

[156] 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)*. 2020, pp. 1–6 (cit. on p. 152).