Operationalizing AI/ML in Future Networks: A Bird's Eye View from the System Perspective

Qiong Liu, Tianzhu Zhang, Masoud Hemmatpour, Han Qiu, Dong Zhang, Chung Shue Chen, Marco Mellia, and Armen Aghasaryan

The authors concentrate on the practical issues of developing and operating machine learning-based solutions in real networks.

Abstract

Modern artificial intelligence (AI) technologies, led by machine learning (ML), have gained unprecedented momentum over the past decade. Following this wave of "AI summer," the network research community has also embraced AI/ML algorithms to address many problems related to network operations and management. However, compared to their counterparts in other domains, most ML-based solutions have yet to receive large-scale deployment due to insufficient maturity for production settings. This article concentrates on the practical issues of developing and operating ML-based solutions in real networks. Specifically, we enumerate the key factors hindering the integration of AI/ML in real networks, and review existing solutions to uncover the missing components. Further, we highlight a promising direction, that is, machine learning operations (MLOps), that can close the gap. We believe this article spotlights the system-related considerations on implementing and maintaining ML-based solutions, and invigorates their full adoption in future networks.

Introduction

The last decade has witnessed a thorough evolution of the modern telco industry with the advent of network softwarization techniques, such as Software-Defined Networking (SDN) and Network Function Virtualization (NFV). By transforming traditional hardware-centric networking components into software-based processes, SDN/NFV authorizes unprecedented flexibility, scalability, and efficiency [1–3]. Despite these benefits, with the rapid expansion of telco infrastructure, the scale and dynamism of modern networks keep growing, and network management remains a daunting task [4].

Meanwhile, AI/ML makes remarkable advancements and has attracted strategic attention across various business sectors. According to Gartner and MIT Sloan Management, AI has led to $3.9T of annual business value and is deemed a strategic priority by 83 percent of CEOs [5]. Inspired by these successes, network researchers are extensively exploring AI/ML for diverse tasks [2, 6]. These *ML-based solutions*, that is, applications, functions, and services, have demonstrated more promising outcomes than traditional fixed-policy approaches [4].

Despite the enormous interest, the modern network's fast-paced evolution has made it impossible to construct and manage large corpses of networking data, which were crucial for AI's successful deployment in real systems. According to a recent report [7], 88 percent of the telco industry's proof-of-concept AI/ML projects fail to reach live deployment. The major deterrent stems from inadequate "system thinking" [8]. Based on our observation, existing AI/ML-based solutions have two fundamental disparities with real-network deployments:

- *One-dimensional design:* ML solutions mainly aim to outperform prior solutions on specific performance metrics, especially accuracy, without vetting other network-/system-critical imperatives. For example, as network operations get increasingly complex and intertwined, optimization becomes multi-metric and multi-dimensional [9].
- *System discrepancy:* These solutions were mostly demonstrated in controlled environments and became costly to fit into real network systems with much higher scale, complexity, and dynamism. For instance, given the data-driven nature of ML-based solutions, fulfilling performance guarantees under sporadic data and environment drifts is non-trivial [10].

This "reality gap" greatly hampers the integration and deployment of AI/ML in real networks.

To make AI/ML an integral part of modern networks, there is a need for lightweight techniques capable of timely prioritizing and triggering model updates, which guarantee the deployed models remain fit for their task regardless of environment evolvement. Given these premises, this article aspires to elucidate the practical challenges of integrating AI/ML into the future network landscape. Specifically, we present network-oriented AI/ML research and its gap with real networks. Then, we enumerate the practical considerations to actualize AI/ML in production-ready networks. Afterward, we prospect a promising direction — MLOps, which applies agile methodologies to combine software development (Dev) and IT operations (Ops), aimed at shortening the systems development lifecycle and providing continuous

Qiong Liu is with Telecom Paris, France; Tianzhu Zhang (co-first author, corresponding author) is with Nokia Bell Labs, France; Masoud Hemmatpour was with Simula Research Laboratory, and is now with Arctic University of Norway, Norway. Han Qiu is with Tsinghua University, China; Dong Zhang is with Fuzhou University, China; Chung Shue Chen is with Nokia Bell Labs, France; Marco Mellia is with Politecnico di Torino, Italy.

Digital Object Identifier: 10.1109/MCOM.001.2400033

delivery with high software quality [11]. We finally introduce two example use cases in network softwarization about continual performance prediction and abnormal detection, where we apply several of the abovementioned techniques.

Landing AI in Networks

In this section, we briefly review the current status of AI/ML and elaborate on the practical barriers obstructing their general adoption in operational networks.

Current States

In recent years, AI/ML has sparked tremendous hype in the operational networks thanks to:

- The innovative breakthroughs in theoretical research
- The success in other fields such as computer vision and NLP
- The presence of optimized development toolkits with hardware acceleration.

Compared to fixed-policy approaches, AI/ML algorithms exhibit exceptional pattern matching, incremental learning, and automation capabilities on large-scale, multidimensional data [6].

Standardization bodies (e.g., ETSI, 3GPP) anticipate AI/ML techniques to be crucial in automating future networks. In February 2024, ETSI released a standard (ETSI TR104032 [12]), which highlighted the necessity of logging key details throughout an AI model's lifecycle by using model trace records like the MLOps framework. Moreover, a 3GPP standard (Rel-17[13]) underscored the necessity for management tools and services to facilitate the incorporation of AI/ML technologies in 5G networks.

In industry, carrier-grade platforms are under active development to bolster AI/ML-augmented network services: Nokia's AVA Ecosystem offers telco operators cloud-native AI/ML and analytic services to automate network operations, enhance service assurance and subscriber experience and reduce cost [7]; Huawei's ADN ecosystem features network automation with dedicated support for AI operations [4], which consists of three tiers, that is, on-device AI, online fog/cloud AI, and offline cloud AI, to support network and AI operations with assorted temporal-spatial properties. In academia, ML algorithms are widely developed to tackle a large spectrum of "networking" problems, such as traffic classification [10], resource scheduling [6], anomaly detection [1], load balancing [2], QoE management [14]. Given the rapid expansion of the AI/ML frontier (e.g., generative AI), their growth in telco networks will continue to enrich. However, there is still a certain distance between proof-of-concept and successful real-time deployment of AI/ML projects. We will discuss the specific difficulties in the following sections.

Challenges and Barriers

The term "ML system" is frequently associated with the algorithms it employs, like logistic regression or various neural networks. However, these algorithms only represent a fraction of a full ML system in a production environment. As mentioned in Fig. 1, ML systems in the real world encompass the initial business objectives, the interfaces, the entire data stack, and the methodologies for model development, monitoring, and updating. ML in production does not connote ML in research, as the latter seldom bothered with the deployment and maintenance issues once the optimization goal was achieved on the test dataset [4]. Based on our study, the key challenges of landing AI in networks can be summarized as follows.

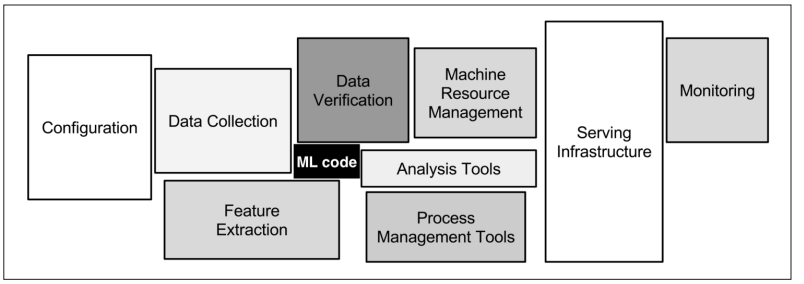

FIGURE 1. Basic components for real-world ML systems (Picture originated from Sculley *et al.* [8]).

Data Complexity: Network data has much more diverse formats, for example, raw packets, flow-level statistics, configuration files, system logs, and event alarms. They may contain categorical, temporal, spatial, or even graph semantics. Such multi-modal data with high variety, velocity, and volume can be exceedingly onerous to model and process [14], not to mention their natural distribution drifts caused by data and system evolvements.

Multi-Dimensional Requirements Nature: Researchers often align on one single objective. The most common objective is model performance: developing a model that achieves state-of-the-art results on benchmark datasets. In production networks, KPI optimization cannot be done in isolation. For example, some DNN models with high prediction accuracy can hardly fit into resource-limited network devices [3]. Besides, the potential high inference latency can make the model unsuitable for real-time requirements, particularly in high-speed networks where the service latency is measured in microseconds [14]. In essence, the learning and run-time complexity in ML systems should be equivalently considered: the former pertains to the computational & resource costs associated with developing an ML model, and the latter refers to the costs of deploying and managing a trained model.

Hidden Technical Debts: This term was coined by Sculley *et al.* [8], which refers to the massive operational costs of operationalizing ML-based systems by non-experts. Similar debts also apply in network systems. As existing solutions were mostly developed in simulated or controlled environments, the practical deployment and maintenance issues were usually sidelined. In real systems, instead, ML models should be deployed as part of a data-processing pipeline. Owing to disparate development toolkits and deployment targets, integrating them into real networks can be laborious and error-prone. As network devices can come from sundry vendors with bespoke configuration, optimization, and execution routines, deploying AI/ML on them can result in complicated manual tuning, customization, and feasibility tests. In addition, rather than a one-off process, ML-based solutions must be continuously upgraded to meet business requirements and sustain long-term value over the rapid evolution of the telco industry.

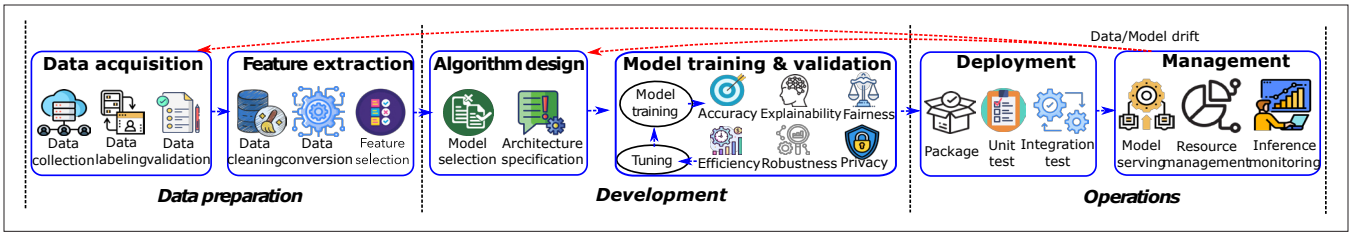

FIGURE 2. ML lifecycle in production settings.

Operationalizing AI/ML in Production Networks: The Status Quo

To close the gap and seamlessly operationalize AI/ML in production, many critical system-related considerations exist throughout the ML lifecycle, that is, data preparation, development, and operations phases, as illustrated in Fig. 2. This section encapsulates these considerations and explores associated studies within the networking domain. The included works were chosen based on two criteria: they address one or more practical aspects, and the methodologies proposed have undergone implementation and verification within actual network systems.

Data Preparation

Data quality directly determines the ceilings of any AI/ML-based product, spurring the recent trend toward data-centric AI [5]. Due to the complexities in real networks, good datasets are not always available. Ensuring data quality can averagely cost 60 percent of time in AI/ML projects [7]. Special considerations should be enforced upon data preparation to supply the ML algorithms with high-quality data: the constituent *data acquisition* and *feature extraction* processes.

Data Acquisition: As supervised learning is the most applied algorithm, obtaining labels is integral to creating training data [9]. In existing solutions, data can generally originate from three sources: live networks, controlled environments, or (curated) public data/datasets. In the first case, despite the various data collection methods, the process can incur huge operational costs, which obligates considerate trade-offs [2]. For example, sampling is usually prioritized over per-packet collection in high-speed networks to attenuate the impact on the datapath. Also, data collection can incur uncontrollable situations, such as packet drops, sampling biases, or schema changes, hence aberrations and outliers. Data labeling remains laborious as it consumes substantial human effort and does not scale with data volume [10]. Despite the advanced techniques (e.g., weak supervision, semi-supervision, transfer learning, and active learning) to mitigate data scarcity, these methods still depend on pre-labeled datasets or human input, limiting their scalability and efficacy in handling large, complex datasets. In the second and third cases, as data are from outside the target networks, its statistical properties can be unaligned with deployment assumptions, leading to unexpected consequences, such as data drifts. Thus, testing becomes necessary to disclose potential biases/anomalies before model deployment.

Feature Extraction: Raw network data must be converted to features conformant with the ensuing AI/ML algorithms. Feature extraction is challenging - different feature sets imply varied system costs (and model performance), thus merit closer scrutiny: many existing ML-based solutions empirically define custom features, which may become hard to obtain and scale in deployment. Furthermore, feature selection schemes, when applied, might face revamping upon network evolution. As detailed in [15], traffic patterns and network conditions in real systems always shift, rendering existing features obsolete and necessitating engineering new features.

Existing Solutions: In contemporary network research, several seminal works approached the practical challenges of data acquisition and feature extraction: Bronzino *et al*. [14] introduced Traffic Refinery, an efficient automation pipeline for flow-level data collection and feature extraction. It aligns network operator goals by consolidating multiple design choices to alleviate packet losses. Additionally, a dedicated profiler quantifies system-level costs, offering operators a trade-off between feature selection and model accuracy. In a distinct exploration, Yao *et al*. [2] proposed the Aquarius framework to enable flexible data collection and feature extraction for data center networks. This system embeds a transport-layer collector for effective TCP traffic feature extraction, storing them in shared memory to facilitate seamless ML algorithm interactions on the control plane, devoid of data plane disruption. Lastly, Holland *et al*. [15] proposed the nPrint framework, which transforms packets into a consistent binary format without sacrificing contextual meaning. This mechanism empowers ML algorithms to automatically identify key features, avoiding the efforts of manual feature extraction.

Development

Model development is an iterative process. With each cycle, it's important to assess the current model's performance compared to its past versions and determine its readiness for live deployment [9]. Model development consists of two fundamental steps, that is, *algorithm design, model training & validation*, each crucial to determine a solution's overall readiness for the target network.

Algorithm Design: The purpose of ML can be threefold: making effective use of *existing* knowledge, gathering a structured understanding of *unknown* phenomena, and *learning* to achieve a goal, which can be mapped to three branches, that is, Supervised, Unsupervised, and Reinforcement Learning (RL) — with potential intersections among them (e.g., semi-supervised or self-supervised learning).

Supervised ML techniques, such as regression and classification, excel at tracking well-specified problems in open-loop settings to increase visibility about network traffic or distill insight from raw data. In particular, regression techniques are fit for forecasting (e.g., traffic demand or user behavior) or learn-

ing complex relationships, such as relating network Quality of Service (QoS) indicators to user Quality of Experience (QoE). Classification techniques are another related example where AI techniques are useful: traffic prioritization requires coarse-grained traffic class labels for policing and may additionally require fine-grained application labels.

Unsupervised ML operates by identifying patterns and structures within data without labeling, relying instead on the algorithm's ability to discern intrinsic features and relationships within the dataset. For example, unsupervised AI employs algorithms in anomaly detection to discern data deviations by autonomously learning underlying distributions. These algorithms identify outliers representing significant departures from established patterns without reliance on pre-labeled normal data instances. Lastly, RL is suitable for sustained and efficient closed-loop AI automation environments. An example is the automation of resource management by using RL, implemented through centralized cloud agents or distributed device agents [4]. In this context, AI agents are dedicated to improving QoS, for example, enhancing transmission efficiency and reducing latency. To attain such a goal, agents are rewarded for their actions, effectively balancing exploration and exploitation within a vast state space, thus providing automated and optimized solutions [15].

Model Training and Validation: In the system context of model training & validation, factors such as inference efficiency, generalizability, and safety hold similar significance as the traditional focus on accuracy. For instance, generalizability ensures timely adaptation in dynamic environments like disaster-resilient networks, safety is crucial for ML algorithms that require frequent interaction with real systems, and inference efficiency is crucial for quick decision-making. The process of training and validating models can be enhanced using tools such as MLflow, Weights & Biases, and DVC. These tools facilitate the selection of ML algorithms and the adjustment of hyperparameters, driving toward automated and efficient optimization of models.

Existing Solutions: Two prior works explore AutoML to automatically carry out model selection and hyper-parameter tuning to hide the AI/ML-specific complexities from network operators. Holland *et al.* [15] leverage the AutoGluon-Tabular framework to locate and ensemble models with high predictive accuracy and low inference latency, given the features and labels. Similarly, Swamy *et al.* [1] employ an optimization framework that automatically performs algorithm selection and model generation as a Bayesian optimization problem based on user intents and network constraints. Lacoboaiea *et al.* [6] address the challenges of building a Deep RL-based channel manager, specifically focusing on training safety, efficiency, environment realism, and generalization. They leverage digital twins for secure training, adjust learning rates for efficiency, enhance simulator fidelity with real-world data, and bolster generalization via synthetic noise and actual data integration.

Operations

This part elaborates on AI/ML-based solutions requiring attention in real networks on *deployment* and *management*.

Deployment: Operational deployment encompasses packaging, customization, and feasibility tests. As traditional ML-based solutions were mainly intended for the control plane, which standard model serving tools can handle. Recently, with the rise of in-network ML, researchers began to push the ML frontier into the network data plane to capitalize on the voluminous data there [3]. Model deployment becomes a Sisyphean task due to the distinctions between the local implementation environment and network infrastructure, and the divergent tooling can sorely impede customization. Moreover, as networks are replete with a plethora of specialized hardware devices (e.g., SmartNICs, P4 switches, embedded devices) with disparate architectures, configuration routines, and resource footprints, the deployment process entails refactoring a solution into a generic data-processing pipeline with minimal interference on the network service [1].

Management: Furthermore, managing the deployed ML-based solutions involves model serving, resource & operation management, and drifting monitoring tasks. In particular, as network systems can evolve expeditiously, the intrinsic concept/data drifts can result in model decay and service degradation. The inference quality should thus be constantly inspected to detect performance diminishments and trigger the model-rebuilding process whenever applicable. In real networks, the correct quality metrics and triggers should be carefully scoped, and the monitoring overhead should also be balanced with the quality assessment accuracy [10]. Depending on the problem context, the rebuilding process can start from the data preparation and labeling or model development stage, which must be specified beforehand.

Existing Solutions: To cope with these challenges, Zheng *et al.* [3] introduce the Planter, a modular architecture that facilitates the seamless deployment of diverse in-network ML algorithms across three prominent hardware platforms. Planter accommodates a slew of mainstream ML algorithms. Its post-training automatically converts the models into tailored P4 code for specific targets, subsequently undergoing compilation and integration for deployment. Similarly, Swamy *et al.* [1] craft compiler tools designed to render target-oriented code for popular data planes autonomously. They harness a cycle-accurate simulator to preemptively gauge the model kPIs, encompassing throughput, latency, and resource utilization. Yang *et al.* [10] tackle inference monitoring and combine gradient-based techniques with Open Set Recognition & explainable AI to scrutinize inference qualities. Comparative evaluations have been conducted to validate the proficiency of their approach in inference monitoring and data drift detection.

We summarize all these pioneering works in Table 1 regarding the tackled lifecycle stages, supported types of ML algorithms, targeted network environment, and use cases. Essentially, each work covers part of the ML lifecycle stages.

Missing Pieces to the Puzzle

Based on the proceeding review, we identify three missing pieces to the fully operationalized AI/ML puzzle. First, despite the optimistic individual advancements, they have not been cumulatively translated into global benefits. In real systems,

Unsupervised ML operates by identifying patterns and structures within data without labeling, relying instead on the algorithm's ability to discern intrinsic features and relationships within the dataset.

Reference	Data acquisition	Feature extraction	Algorithm design	Hyperparam. tuning	Model training	Validation	Deployment	Management	Target network	Use cases
Bronzino et al. [14]	✓	✓							—	QoE inference
Yao et al. [2]	✓	✓							Datacenter network	Load balancing Traffic classification Resource scheduling
Holland et al. [15]		✓	✓	✓	✓	✓			—	Traffic analysis
Swamy et al. [1]			✓	✓	✓	✓	✓		Datacenter network	Anomaly detection Traffic classification Botnet detection
Lacoboaiea et al. [6]					✓	✓			WLAN	Resource scheduling
Zheng et al. [3]					✓	✓	✓	✓	Datacenter network	Anomaly detection QoE inference
Yang et al. [10]								✓	—	Traffic classification

TABLE 1. Synoptic of the related works.

individual stages must be seamlessly articulated as an end-to-end data processing pipeline. With the current reliance on manual interventions, ML-based solutions will become heavy to manage in future networks. Second, reproducibility is not enforced due to the absence of systematic logging and tracking. Traditional version control tools cannot sufficiently capture the nuances of ML workflows' datasets, parameters, and configuration dependencies, which must be consistently reproducible for scientific rigor and regulatory compliance. Third, silos can also arise due to the disparate expertise & priorities of data scientists and network engineers, hampering productivity and stalling time-to-value.

Figure 3 illustrates two approaches for ML lifecycle management. The traditional workflow is a one-off process of data collection, model development, and deployment. This approach prioritizes rapid delivery for the initial time. Nonetheless, as the temporal dimension extends, this method becomes less efficient. In particular, the data/system shifts necessitate continuous model retraining. Without proper management, reproducing and enhancing existing models become laborious as the whole process can involve multiple teams, from data scientists to network engineers. Manual asset transferring is inefficient and burdensome.

Conversely, the second approach adopts a more systematic strategy. Initially, the involved teams dedicate significant time to constructing an automated pipeline with established tracking mechanisms. Compared to the manual approach, it confers substantial long-term benefits, including reproducibility, continuous model enhancement, and seamless communication.

Continual Learning

Continual learning enables AI/ML practitioners to update and deploy models efficiently. It addresses data distribution drifts, adjusts models based on rare events, and solves the cold start problem arising from unseen data [9]. With respect to network systems, we enumerate the progression toward continual learning below.

Stage 1 — Manual, Stateless Retraining: Initially, researchers manually retrain models without leveraging historical data state, which is common in settings without dedicated teams to manage ML platforms.

Stage 2 — Automated Retraining: Researchers begin automating model retraining. The retraining frequency often relies on intuition, such as daily updates, to optimize performance without a solid empirical basis.

Stage 3 — Automated, Stateful Training: To improve efficiency, researchers begin exploring the recently saved model states and checkpoints, which is especially beneficial for use cases requiring frequent model updates.

Stage 4 — Continual Learning for Network Management: The most advanced phase involves transitioning from fixed-schedule updates to dynamic, trigger-based model updates based on time intervals, performance metrics, network volume, or traffic patterns, enabling more responsive and adaptive network management.

Applying continual learning in modern networks faces significant hurdles. Fortunately, MLOps provides means to offset them, as detailed in the next section.

MLOps: Toward End-to-End Pipelines

MLOps is an emerging set of practices that apply DevOps principles to unify the development and operation of ML-based systems [5, 9].

Why MLOps?

Traditionally, the operational costs of delivering software products can be countered with DevOps, which encompasses an assemblage of principles to break the silo between software developers and IT operations engineers, promoting Automation and Continuous Integration (CI)/Continuous Deployment (CD) throughout the product lifecycle. These principles help drive IT and business outcomes for many businesses and organizations [10]. The network community has adopted DevOps to fuel technological innovation and revenue growth.

However, though DevOps can curb the operational overhead of productionalizing traditional software projects, they lack supplemental support for the unique characteristics of ML. There are five fundamental discrepancies between conventional software and ML: First, code quality predominantly decides the performance in traditional software; In AI/ML, the model and data all impact the outcome

[5]. Second, traditional software is usually built on full-fledged libraries with clear abstraction boundaries [8]. ML-based solutions often involve a broader range of tools and libraries, subject to extra integration and maintenance costs. Third, unlike traditional software that conveys deterministic outputs, ML models are intrinsically stochastic and entail disparate processes to validate their behaviors. Fourth, ML models are susceptible to data/concept drifts, which are common in real networks and thus necessitate drift detection and model rebuilding [10]. Finally, building and operating ML-based solutions call for data science skillsets, which are missing in traditional software/network routines. According to a recent survey, 55 percent of telcos lack the pertinent data science talent [7].

Layered on the DevOps tenets, MLOps accommodates the unique traits of AI/ML with the following practices:

- Continual Monitoring (CM)/Continual Training (CT): MLOps addresses the model decay problem by constantly monitoring the data and inference quality and rebuilding the model whenever applicable.
- Automation: MLOps streamlines AI/ML lifecycle into a fully automated pipeline to alleviate operational costs.
- Versioning: Based on DevOps, MLOps extends the version control of artifacts involved in the process, including data, model, and code.
- Experiment Tracking: Experiments are systematically tracked to ensure reproducibility and auditability.
- Collaboration: MLOps advocates a common platform to build synergy across the involved participants.

With these practices, MLOps consolidates innovations across the AI/ML lifecycle and dramatically curtails operational costs, even though this burgeoning discipline is still nascent for the network research community. We envision a plausible architecture in Fig. 4, which adopts most MLOps practices for real networks.

MLOps for Networking: A Case Study

We demonstrate the advantages of MLOps through a case study on real-time KPI prediction, a critical aspect of network management. We deploy a network service chain in a smallscale data center and explore a lightweight ANN model for "non-intrusive" KPI prediction using infrastructural-level hardware features. We employ the Pearson Correlation Coefficient for feature selection, Bayesian optimization for automatic hyperparameter tuning, and Jensen-Shannon divergence to quantify data drift. We restructured the processing pipeline using Kubeflow, an open-source MLOps platform based on Kubernetes. Figure 5 demonstrates how MLOps enables real-time KPI prediction with sustainable performance. Initially, our model achieved an average prediction accuracy of 91 percent

Conclusion

Due to the lack of system-related considerations, AI/ML is still not an integral part of modern networks. This article analyzed the inconsistencies between existing AI/ML-based solutions and real network systems and discussed all the practical considerations throughout their product lifecycle. We also reviewed the related works and identified the missing pieces. Then, we conducted a case study to validate the advantages of MLOps in a real network system. This article can raise awareness about the practical hurdles of operationalizing AI/ML in production settings and expedite its integration into future networks.

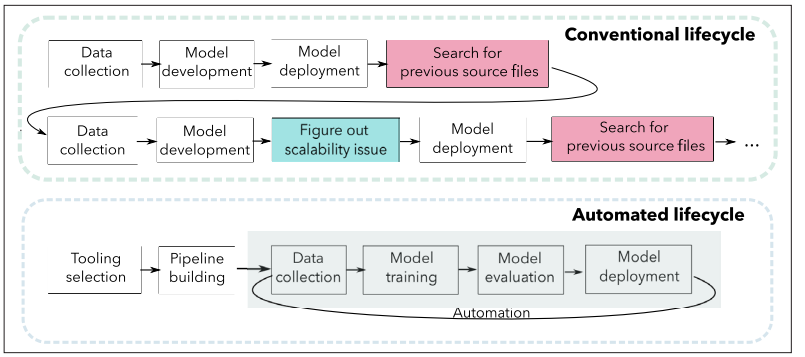

FIGURE 3. Conventional vs. Automated ML lifecycle.

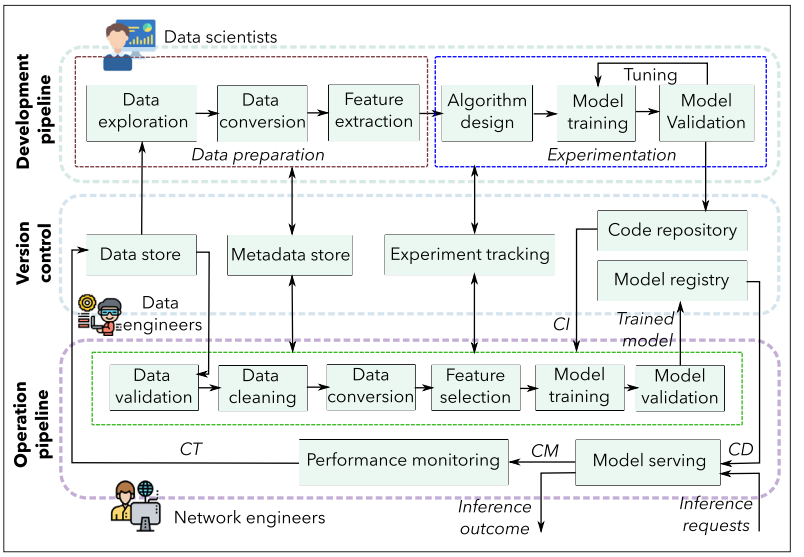

FIGURE 4. Operationalizing AI/ML in Future Networks: A Bird's Eye View from the System Perspective.

References

[1] T. Swamy *et al.*, "Homunculus: Auto-Generating Efficient Data-Plane Ml Pipelines for Datacenter Networks," *ACM ASPLOS*, vol. 3, p. 329–42.

[2] Z. Yao *et al.*, "Aquarius—Enable Fast, Scalable, Data-Driven Service Management in the Cloud," *IEEE TNSM*, vol. 19, no. 4, 2022, pp. 4028–44.

[3] C. Zheng *et al.*, "Planter: Rapid Prototyping of In-Network Machine Learning Inference," *ACM SIGCOMM CCR*, 2024.

[4] D. Rossi *et al.*, "Landing AI on Networks: An Equipment Vendor Viewpoint on Autonomous Driving Networks," *IEEE TNSM*, vol. 19, no. 3, 2022, pp. 3670–84.

[5] J. Bradley *et al.*, "The Big Book of MLOps," https://www.databricks. com/p/ebook/the-big-book-of-mlops; accessed: 2023-12-13.

[6] O. Iacoboaiea *et al.*, "From Design to Deployment of Zero-Touch Deep Reinforcement Learning WLANs," *IEEE Commun. Mag.*, vol. 61, no. 2, 2023, pp. 104–09.

[7] Nokia Networks, "AVA AI and Analytics," https://nokia.com/networks/ ai-and-analytics, 2022; accessed: 2023-12-13.

[8] D. Sculley *et al.*, "Hidden Technical Debt in Machine Learning Systems," *NeurIPS*, vol. 28, 2015.

[9] C. Huyen, *Designing Machine Learning Systemss: An Iterative Process for Production-ready Applications*, O'Reilly Media, 2022.

[10] L. Yang *et al.*, "Quality Monitoring and Assessment of Deployed Deep Learning Models for Network AIOps," *IEEE Network*, vol. 35, no. 6, 2021, pp. 84–90.

[11] D. Rossi *et al.*, "Network Artificial Intelligence, Fast and Slow," *ACM NativeNi*, 2022, pp. 14–20.

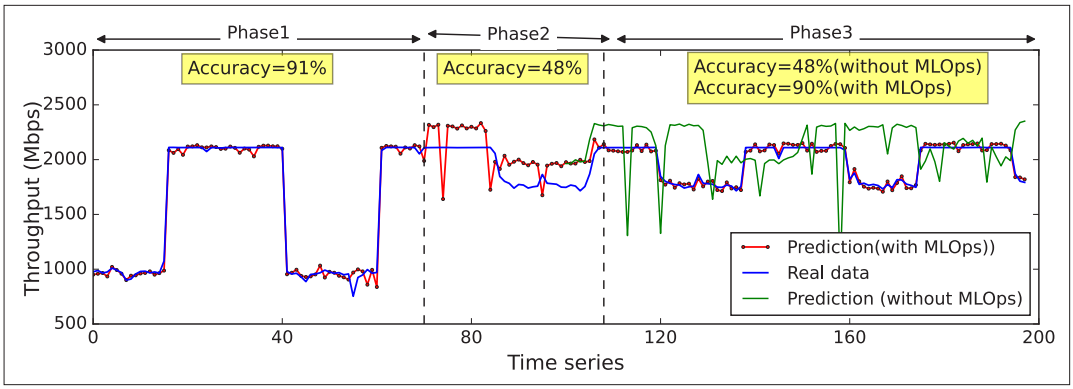

FIGURE 5. The benefits of MLOps for networking.

[12] European Telecommunications Standards Institute, "SIA: Traceability of AI Models," https://www.etsi.org/deliver/etsi tr/104000 104099/104032/01.01.01 60/tr104032v010101p.pdf, Mar. 2021; accessed 2-Apr-2024.
[13] 3GPP, "Study on AI/ML Management," https://www.3gpp.org/technologies/ai-ml-management, 2024; accessed 2-Apr-2024.
[14] F. Bronzino *et al.*, "Traffic Refinery: Cost-Aware Data Representation for Machine Learning on Network Traffic," *ACM POMACS*, vol. 5, no. 3, 2021, pp. 1–24.
[15] J. Holland *et al.*, "New Directions in Automated Traffic Analysis," *ACM CCS*, 2021, pp. 3366–83.

Biographies

Qiong Liu [M] is a postdoc researcher at Telecom Paris. She received her B.S. degree from Shandong University, China 2015. She received the M.S. degree from Xidian University in 2018, and the Ph.D degree from INSA Rennes, France in 2022. Currently, she focuses on applied AI for network systems and stochastic geometry-based performance evaluation in large-scale networks.

Tianzhu Zhang [M] is a research scientist at Nokia Bell Labs. He received his B.S. degree from Huazhong University of Science and Technology, China, in 2012. He received his M.S. and Ph.D. degrees in 2014 and 2017 from Politecnico di Torino, Italy. From 2017 to 2019, he was a PostDoc researcher at Telecom ParisTech. His research interests center around applied AI/ML for network systems.

Masoud Hemmaptour received the MS degree in computer and communication network engineering and the PhD degree in control and computer engineering from Politecnico di Torino, Italy, in 2015, and 2019, respectively. His research interests include high-performance interconnect and programmable network devices, Currently, he is a researcher at the Arctic University of Norway. His research interest centers around the performance and energy efficiency of in-network processing applications.

Han Qiu received the B.E. degree from the Beijing University of Posts and Telecommunications, China, in 2011, the M.S. degree from Institute Eurecom, France, in 2013, and the Ph.D. degree from the Department of Networks and Computer Science, Telecom-ParisTech, France, in 2017. He worked as a postdoc at Telecom Paris from 2017 to 2020. He is an assistant professor at the Institute for Network Sciences and Cyberspace, Tsinghua University, China. His research interests include AI & Data security and cloud computing.

Dong Zhang [M] received the B.S. and Ph.D. degrees from Zhejiang University, China, in 2005 and 2010, respectively. He visited Alabama University, USA, as a Visiting Scholar from 2018 to 2019. He is currently a Professor at the College of Computer Science and Big Data, Fuzhou University, China. His research interests include software-defined networking, network virtualization, and Internet QoS.

Chung Shue Chen [SM] received the B.Eng., M.Phil., and Ph.D. degrees in information engineering from the Chinese University of Hong Kong (CUHK), Hong Kong, in 1999, 2001, and 2005. He is a DMTS at Nokia Bell Labs. His research interests include wireless networks, communications, optimization, machine learning, 5G/6G, IoT, and intelligent systems.

Marco Mellia [M'97, SM'08, F'20] is a full professor at the Control and Computer Engineering Department of Politecnico di Torino and the coordinator of the SmartData@PoliTO Center on Data Science and Machine Learning. His research interests include traffic monitoring and big data analysis, with applications to traffic classification, management, and security.

Armen Aghasaryan holds a PhD in signal processing and telecommunications from INRIA / University of Rennes, France. He joined Alcatel in 2000 and is heading the Machine Learning & Systems group in the AI Research Lab, Nokia Bell Labs. His research interests include AI/ML, Data Analytics, Cloud Computing, and Network Automation.