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

***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 considerations. Further, we highlight a promising direction, i.e., Machine Learning Operations (MLOps), that can close the gap. We believe this paper spotlights the system-related considerations on implementing & maintaining ML-based solutions and invigorate their full adoption in future networks.**

***Index Terms*—AI/ML for networking, Network Systems, Network engineering**

I. Introduction

To drive digital transformation, modern telecommunication networks are undergoing a disruptive evolution. The ongoing 5G rollout promises to deliver customized network services to billions of subscribers with ultra-high speed, ultra-high reliability, ultra-low latency, and ubiquitous connectivity. The massification of Internet-of-Things is expected to connect trillions of devices and create brand-new business models. Next-generation digital realms, e.g., Metaverse, also call for high-quality, customizable communication mediums to drive human-machine interaction and digital-physical fusion. These technical headways inevitably make modern networks increasingly diverse, decentralized, and complex. Traditionally, networks were mainly managed by predefined or ad-hoc rules. However, these methods either bear oversimplified assumptions about the underlying systems or incur huge computation overhead, which disaccords with the continuing network complexification. As of today, human participation is still essential for in-depth problem diagnoses and decision-making.

Q. Liu is with the Department of Computer Sciences and Networks, Telecom Paris, Palaiseau, France. (E-mail: qiong.liu@telecom-paris.fr)

T. Zhang, C. S. Chen, and A. Aghasaryan are with Nokia Bell Labs, Massy, France. (E-mail: {tianzhu.zhang, chung_shue.chen, armen.aghasaryan}@nokia-bell-labs.com)

M. Hemmatpour is with Simula Research Laboratory, Oslo, Norway. (E-mail: mashemat@simula.no)

H. Qiu is with the Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China. (E-mail: qiuhan@tsinghua.edu.cn)

D. Zhang is with the College of Computer Science and Big Data, Fuzhou University, Fuzhou, China. (E-mail: zhangdong@fzu.edu.cn)

M. Mellia is with the Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy. (E-mail: marco.mellia@polito.it)

With the accruing breakthroughs over the last decade, modern AI/ML has achieved human-level performance in many challenging tasks, e.g., computer vision, gaming, and, more recently, natural language processing. Nowadays, AI/ML-empowered products have permeated various industrial and business sectors, including healthcare, manufacturing, entertainment, and education. According to the findings of Gartner and MIT Sloan Management Review, AI has led to \$3.9T of business value in 2022 and is deemed a strategic priority by 83% of CEOs [1]. Inspired by these successes, network researchers are extensively exploring AI/ML algorithms for network automation and management. This new breed of ML-based networking solutions[1], i.e., applications, functions, and services, has demonstrated more optimistic outcomes than the traditional fixed-policy approaches [2].

Despite the enormous interest, AI/ML is still immature for deployment in real networks. According to a recent report [3], 88% of the telco industry's proof-of-concept AI/ML projects fail to reach live deployment. The major deterrent stems from inadequate "system thinking". Based on our observation, existing AI/ML-based solutions have two fundamental disparities with real-network deployments: (i) One-dimensional design: ML solutions mainly aim to outperform prior solutions on specific metrics, especially accuracy, without vetting other network-/system-critical imperatives. For example, as networks usually become increasingly complex and intertwined, optimization becomes multi-metric and multi-dimensional. (ii) 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, ensuring their performance under sporadic data drifts and system evolutions is non-trivial [4]. Consequently, such a "reality gap" hampers the integration and deployment of AI/ML in real networks.

In information technology (IT) industry, the development and delivery of software products is usually streamlined via Development and Operations (DevOps) practices, which are further customized by the telco industry to enhance service quality and reduce time-to-market [5]. Although network-oriented DevOps practices offset part of the exertions via continuous integration and delivery, they can hardly accommodate the unique characteristics of AI/ML [4]. To smoothly operationalize (i.e., develop, deploy, and manage) AI/ML in

[1]Note that we focus on techniques of AI's data-driven branch, frequently denoted as ML. By abuse of language, we will use the two acronyms interchangeably in this paper.

production, network operators must master skills in ML, data engineering, and systems, which is extremely burdensome.

This paper 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 in Sec. II. In Sec. III, we enumerate the practical considerations to actualize AI/ML in production-ready networks. Afterward, we prospect a promising direction - Machine Learning Operations (MLOps), which closes the gap of operationalizing AI/ML in telco networks in Sec. IV.

II. Background

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

A. Landing AI in Networks

In recent years, AI/ML has sparked tremendous hype in the telco industry thanks to (i) the innovative breakthroughs in theoretical research, (ii) the success in other fields such as computer vision and NLP, and (iii) 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, multi-dimensional data [6].

Standardization bodies (e.g., ETSI, 3GPP) anticipate AI/ML techniques to play a pivotal role in automating future networks and have formed multiple working groups to investigate different use cases [2]. 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 [3]; Huawei's ADN ecosystem features network automation with dedicated support for AI operations [2], which consists of three tiers, i.e., 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 [7], resource scheduling [6], anomaly detection [8], load balancing [9], QoE management [10]. Given the rapid expansion of the AI/ML frontier (e.g., generative AI), their growth in telco networks will continue to enrich.

B. The reality gap

Despite the plethora of solutions, a closer inspection reveals a less rosy picture - These solutions are generally inadequate for live deployment in real network systems [3]. Based on our study, such a reality gap originates from three factors:

Data complexity: Compared to other prevalent AI/ML application domains, e.g., computer vision and language processing, network data has much more diverse formats, e.g., raw packets, flow, 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 [10], not to mention their natural distribution drifts caused by data and system evolvements.

One-dimensional nature: Existing solutions generally focus on optimizing specific performance metrics rather than comprehensively assessing the overall readiness, which is incompatible with the stipulations of real-world ML systems. In particular, some solutions strive for high prediction accuracy using supersized Deep Neural Networks (DNNs) models that can hardly fit into resource-limited network devices. The potentially high inference latency can make them unsuitable for real-time constraints. They may also interfere with the critical data path. But most of all, in production networks, all Key Performance Indicators (KPIs) are naturally intertwined and must be jointly analyzed and attuned to network/system constraints to avoid one-dimensional solutions.

Hidden technical debts: This term was coined by Sculley et al. [11], 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.

III. 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, i.e., data preparation, development, and operations phases, as illustrated in Fig. 1. This section encapsulates these considerations and explores associated studies within the networking domain. The included works were chosen based on two criteria: (i) they address one or more practical aspects, and (ii) the methodologies proposed have undergone implementation and verification within actual network systems.

A. Data preparation

Data quality directly determines the ceilings of any AI/ML-based product, spurring the recent trend towards data-centric AI [1]. Due to the complexities in real networks, good datasets are not always available. Ensuring data quality can averagely cost 60% of time in AI/ML projects [3]. 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.

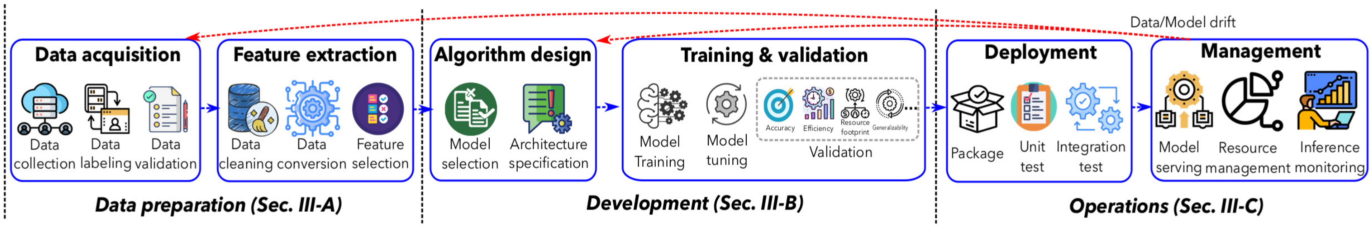

Fig. 1: ML lifecycle in production settings.

Data acquisition. In existing solutions, data can generally originate from three sources: (i) live networks, (ii) controlled environments, or (iii) (curated) public datasets. In case (i), despite the multitudinous data measurement and collection methods, the process can incur huge operational costs, which obligates considerate tradeoffs [9]. For example, sampling is usually prioritized over the per-packet collection in high-speed networks to attenuate the impact on the traffic datapath. Also, network data collection can incur uncontrollable situations, such as packet drops, sampling biases, or schema changes, hence aberrations and outliers [7]. Labeling the collected data still remains a daunting problem as it consumes substantial human effort and does not scale with the data volume and scenarios. In cases (ii) and (iii), as data are from outside the target networks, their statistical properties can be unaligned with deployment assumptions, which leads to unexpected consequences. The lack of labeled data (and datasets) often limits the reproducibility and the solution generalization. Testing in real environments becomes thus necessary to disclose the potential biases/anomalies before model deployment. At last, existing solutions often assume statistically stationary input properties, making them susceptible to uncertainties and novelties commonly found in real networks.

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.

Existing solutions: In contemporary network research, several seminal works approached the practical challenges of data acquisition and feature extraction: Bronzino et al. [10] 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. [9] 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. [12] 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.

The data preparation phase involves direct interaction with real networks, which can be extremely challenging given the high-speed data flows and system-level complexities. Despite the limited test scenarios, these solutions provide valuable first-hand guidelines for efficiently collecting & representing data and initiate a broader trend in network research toward developing more sophisticated, ML-compatible methods for data preparation.

B. Development

Model development consists of two fundamental steps, i.e., 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: (i) making effective use of *existing* knowledge, (ii) gathering a structured understanding of *unknown* phenomena, and (iii) *learning* to achieve a goal, which can be mapped to three branches, i.e., 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 learning 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 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.

RL is suitable for sustained and efficient closed-loop AI automation environments. An example is the automation of

TABLE I: Synoptic of the related works.

Reference	*Data acquisition*	*Feature extraction*	*Algorithm design*	*Hyperparam. tuning*	*Model training*	*Validation*	*Deployment*	*Management*	Target network	Use cases
Bronzino et al. [10]	✓	✓							-	QoE inference
Yao et al. [9]	✓	✓							Datacenter network	Load balancing Traffic classification Resource scheduling
Holland et al. [12]		✓	✓	✓	✓	✓			-	Traffic analysis
Swamy et al. [8]			✓	✓	✓	✓	✓		Datacenter network	Anomaly detection Traffic classification Botnet detection
Lacoboaiea et al. [6]					✓	✓			WLAN	Resource scheduling
Zheng et al. [13]					✓	✓	✓	✓	Datacenter network	Anomaly detection QoE inference
Yang et al. [4]								✓	-	Traffic classification

resource management by using RL, implemented through centralized cloud agents or distributed device agents. In this context, AI agents are dedicated to improving QoS, e.g., 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 network operators with automated and optimized solutions [12].

Model training & 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.

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. [12] 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. [8] 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.

These initial research endeavors provide viable directions for model development in the context of real network systems. Moving forward, the focus of network-oriented AI/ML-based solutions will likely shift towards further simplifying the complexity of these technologies for more diverse network environments and application scenarios.

C. Operations

In this part, we elaborate on AI/ML-based solutions that require attention in real-world networks concerning deployment and management.

Deployment. Operational deployment encompasses packaging, customization, and feasibility tests. As ML-based solutions were mainly intended for the control plane, these tasks can be handled by general-purpose model serving tools. Recently, intrigued by the advantages of in-network ML, researchers began to push the ML frontier into the network data plane to capitalize on the voluminous data there [13]. 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 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 [8].

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 [4]. 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. [13] introduce the Planter framework as a countermeasure to these limitations. This modular architecture facilitates the seamless deployment of diverse in-network ML algorithms across three prominent hardware platforms: Intel Tofino, BMv2, and P4Pi. Notably, Planter is adept at accommodating a slew of mainstream ML algorithms. The post-training phase sees these models automatically transform into P4 code tailored to the specific target, subsequently undergoing compilation and data plane integration for functional validation and deployment. On the other hand, Swamy et al. [8] craft compiler tools designed to autonomously render target-oriented code for three widely adopted deployment platforms: FPGA, Tofino, and Taurus. Additionally, they harness a cycle-

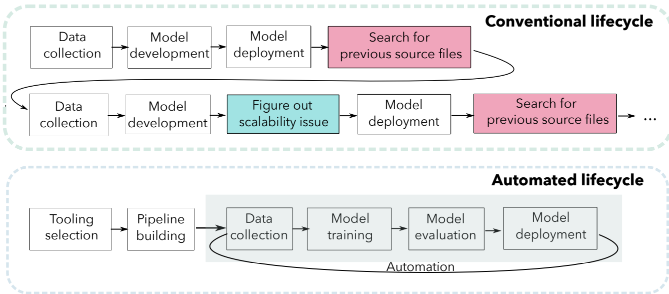

Fig. 2: Conventional vs. Automated ML lifecycle.

accurate simulator to preemptively gauge the model's key performance indicators (KPIs), encompassing throughput, latency, and resource allocation metrics. Yang et al. [4] focus on the pivotal concern of monitoring. Their proposed methodology amalgamates gradient-based techniques with Open Set Recognition and explainable AI paradigms to scrutinize and gauge the quality of inferences. Comparative evaluations have been conducted to validate the proficiency of their approach in delivering meticulous model evaluations, consistently monitoring inference quality, and detecting nuanced data drifts.

These works constitute the vital advancements in deploying and managing the pre-trained AI/ML-based solutions in real network systems. As the network vendors keep introducing more sophisticated features and devices, the supported scope of ML models, deployment targets, and drift detection methods still require further expansion, which the tools and methods of these works can assist.

Based on the discussions in Sec. III-A, III-B, III-C. We summarize all the pioneering works in Table I in terms of the tackled lifecycle stages, supported types of ML algorithms, targeted network environment, and already-validated networking use cases. Essentially, each work covers part of the ML lifecycle stages.

D. 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, all the individual stages must be seamlessly articulated as an end-to-end data processing pipeline to reduce the management & maintenance overhead. 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, communication barriers or silos can also arise due to the disparate expertise & priorities of data scientists and network engineers, hampering productivity and slowing time-to-value.

We specifically illustrate two approaches for managing the ML lifecycle in Fig. 2. The traditional workflow involves a one-off process involving 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 of the model development data, reproducing and enhancing existing models become laborious as the whole process can involve multiple teams, from data science to network engineering. Manual asset transferring is inefficient and burdensome.

Conversely, the second approach adopts a more systematic approach. Initially, the involved teams dedicate significant time to constructing an automated pipeline with established tracking mechanisms. Compared to the conventional manual approach, it confers substantial long-term benefits. Being fully automated, the process ensures consistent reproducibility, enabling continuous model enhancement, and paves the way for concerted efforts across different teams.

IV. MLOps: Towards End-to-End Pipelines

This section discusses MLOps as a potential solution for relieving the operational overhead.

A. Why MLOps?

MLOps is an emerging set of engineering practices in the ML field aiming at applying DevOps principles to unify the development and operation of ML-based systems. 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. The network community has adopted DevOps to fuel technological innovation and revenue growth [4].

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 [1]. Second, traditional software is usually built on full-fledged libraries with clear abstraction boundaries [11]. ML-based solutions, au contraire, often involve a broader range of tools, libraries, and platforms, 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 monitoring and model rebuilding [4]. 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% of telcos lack the pertinent data science talent [3]. Although network practitioners can gradually get acquainted with AI/ML and data science, mastering the theories and technical details requires time.

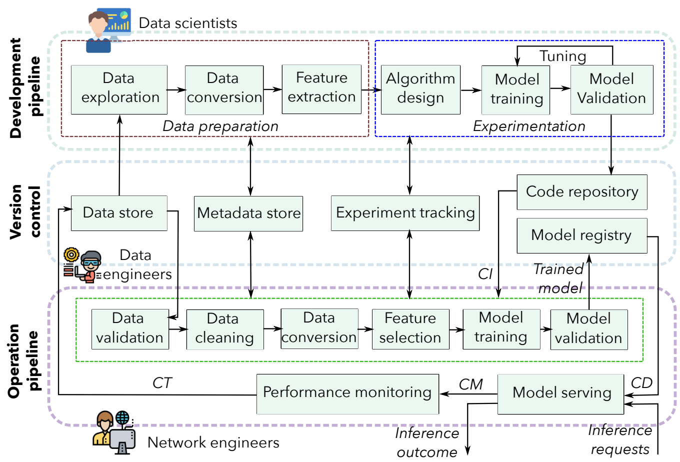

Fig. 3: MLOps for networking: A tenable architecture.

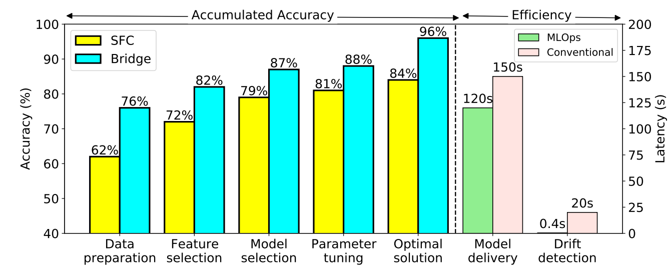

Fig. 4: The MLOps benefits on accuracy and efficiency.

B. What is MLOps?

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

- Continuous Monitoring (CM) / Continuous Training (CT): MLOps addresses the model decay problem by constantly monitoring the data and inference quality and rebuilding the model whenever applicable.
- Automation: To alleviate the operational cost, MLOps aims to streamline all the AI/ML lifecycle stages as a fully automated pipeline without human intervention.
- Versioning: Based on the DevOps version control for code, MLOps advocates the version control of artifacts involved in the process, including data, model, and code. The accompanying data and feature stores also simplify data governance.
- Experiment tracking: All experiments should be systematically tracked to ensure reproducibility and auditability.
- Collaboration: MLOps advocates a common platform and language to build synergy across the involved persons with different priorities and expertise.

With these practices, MLOps proposes a promising way to consolidate innovations across the ML lifecycle and dramatically curtail operational costs, even though this burgeoning discipline is still nascent in the AI/ML network community. We envision a plausible architecture in Fig. 3, where most MLOps practices can be adopted for real networks, though considerable tooling and engineering efforts are still inevitable.

C. MLOps for networking: A case study

This section demonstrates the benefits of MLOps with a case study on a real network benchmark. Specifically, we implement an ML-based solution for KPI forecasting, a crucial task for predictive maintenance in operational networks. To retain production setting, we deploy software network services across a distributed cluster inside a small-scale enterprise datacenter, where each node is a commodity off-the-shelf server equipped with Intel Xeon CPUs and Intel 10/40-Gigabit network cards. The network services attain line-rate traffic processing. We predict the overall network service performance, e.g., throughput, using the features of the underlying hardware subsystems, including Processor, Memory, and PCI buses, to circumvent the intricacy of in-band data collection. These features can be readily obtained using standard profiling tools. As detailed in [14], though these features seem less relevant than the packet-/flow-level statistics, they contain valuable insights on service performance and can be utilized for KPI predictions. To elude the aforementioned manual exertions throughout the data preparation, development, and operations phases, we adopt common MLOps practices to build a prototypical data processing pipeline, whose workflow is detailed as follows:

- Data preparation: We define standard procedures for the entailed steps (i.e., data collection, validation, and labeling) to automatically obtain high-quality data without interfering with the normal operations in the data plane.
- Feature selection: To extract the most relevant features from the ~600 features collected by standard profiling tools (e.g., Linux perf, Intel PCM), we employ Pearson's correlation coefficient to assay their statistical dependencies with the target service KPIs.
- Automated model selection: According to the studies of Manousis et al. [14], we consider three ML models: Linear Regression, Gradient Boosting Regression, and Artificial Neural Network, for automated model selection.
- Automated hyper-parameter tuning: We employ Bayesian optimization to evaluate the performance of different hyperparameter combinations and build a probabilistic model to automatically select the optimal one.
- Performance monitoring and feedback loop: We employ the Jensen-Shannon divergence to inspect the input data distribution to establish a continuous monitoring mechanism. When the new data diverges from the training data, the feedback loop should be activated to retrain and roll out an up-to-date model.
- Versioning and experiment tracking: we employ MLflow [1] to maintain detailed records of all the involved model artifacts, datasets, (hyper-)parameters, and other assets in the model development process, ensuring both reproducibility and auditability.

An empirical evaluation of our data processing pipeline is presented in Fig. 4. We focus on two test scenarios: a singleton virtual network function (L2 bridge) and a Linear Service Chain (SFC) with five network functions. For each scenario, we collect the training data under varied input loads. We first investigate the benefits of incrementally applying the customized data preparation, feature selection, model selection, and hyper-parameter tuning procedures. As shown in Fig. 4, our systematic refinement strategies can incrementally enhance the overall accuracy, i.e., $76\% \rightarrow 96\%$ for bridge and $62\% \rightarrow 84\%$ for SFC. Furthermore, the implemented automation can significantly reduce the incurred model delivery time compared to the conventional, manual approaches. In particular, the time required to detect data drift has been reduced to merely 0.4 seconds, thanks to the seamless identification of distribution shifts. Given that our prototype has not even integrated the most advanced tools and techniques, we believe MLOps can offer huge performance boosts and smooth the adoption of AI/ML in future networks.

V. Conclusion

Due to the lack of system-related considerations, AI/ML is still not an integral part of modern networks. This paper 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. Based on our experience, we recommend MLOps as a promising way to erase operational concerns. We believe this paper can raise awareness about the practical hurdles of developing, deploying, and managing AI/ML-based solutions in production settings and expedite the integration of AI/ML in future network systems.

References

[1] J. Bradley et al., "The Big Book of MLOps," https://www.databricks.com/p/ebook/the-big-book-of-mlops, 2022.

[2] D. Rossi et al., "Landing AI on networks: An equipment vendor viewpoint on autonomous driving networks," IEEE TNSM, 2022.

[3] Nokia Networks, "AVA AI and analytics," https://nokia.com/networks/ai-and-analytics, 2022.

[4] L. Yang et al., "Quality monitoring and assessment of deployed deep learning models for network AIOps," IEEE Network, 2021.

[5] W. John et al., "Service provider devops," IEEE Communications Magazine, 2017.

[6] O. Iacoboaiea et al., "From design to deployment of zero-touch deep reinforcement learning WLANs," IEEE Commun. Mag., 2023.

[7] F. Pacheco et al., "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," IEEE Commun. Surveys Tuts., 2018.

[8] T. Swamy et al., "Homunculus: Auto-generating efficient data-plane ML pipelines for datacenter networks," arXiv:2206.05592, 2022.

[9] Z. Yao et al., "Aquarius-enable fast, scalable, data-driven service management in the cloud," IEEE TNSM, 2022.

[10] F. Bronzino et al., "Traffic Refinery: Cost-aware data representation for machine learning on network traffic," ACM POMACS, 2021.

[11] D. Sculley et al., "Hidden technical debt in machine learning systems," NeurIPS, 2015.

[12] J. Holland et al., "New directions in automated traffic analysis," in ACM CCS, 2021.

[13] C. Zheng et al., "Automating in-network machine learning," arXiv:2205.08824, 2022.

[14] A. Manousis et al., "Contention-aware performance prediction for virtualized network functions," in ACM SIGCOMM, 2020.