OpenData: A Framework to Train and Deploy ML Solutions in Wide-Area Networks

Sina Keshvadi, Shuihai Hu, Geng Li, Yi Lian

***Abstract*—Data-driven solutions hold significant promise for improving network protocols and services in wide area networks. However, their practical adoption in production networks has been limited. This paper investigates the potential of leveraging network data itself to enhance the effectiveness of data-driven solutions. We evaluate a Quality of Service (QoS) forecasting model trained on directly collected network data to demonstrate the advantages of harnessing networking data for machine learning purposes. Our results reveal that training the model with network data effectively addresses the challenges of Data Drift. We also acknowledge the limitations of designing a generic framework to support all problem domains. To overcome this challenge, we propose a comprehensive set of potential solutions that leverage network data for machine learning (ML) applications.**

***Index Terms*—Data-Driven Solutions, Wide Area Networks, Network Data**

I. Introduction

Machine Learning (ML) has emerged as the method of choice for developing practical software for computer vision, speech recognition, natural language processing, robot control, etc. The success of ML applications in addressing these problems has motivated researchers to adapt ML as a tool for solving complex problems of modern computer networks. Though innovative in recently proposed data-driven solutions, these proposals suffer from a consistent challenge: the risk and difficulty of deploying ML models in a production network [1].

The main focus in data-driven solutions is around the data, where the behavior of a ML system is dependent on the qualities of its input features. Building, training, and testing ML models to address complex networking problems, such as congestion control and traffic engineering, often requires a large amount of data. In current practices, ML developers mainly acquire training data from datasets or collect synthetic data from network traffic generators, simulators, or emulators, which none of them are able to accurately represent the behavior of a target environment that a model must operate in.

As networks are highly connected and filled with dependencies, small changes in one part of the network can result in large performance effects in others. Due to the heterogeneity of network functionalities and services, it is not easy to migrate a data-driven solution from one network to another network [2]. Data-driven solutions need to be trained in the target network to learn complex properties about that network and make optimized decisions [3]. The end result is that, often, the only way to ascertain the true performance of a data-driven solution at a production network is to build and train its model with proper data collected from the same network that the model is supposed to operate in [2].

Unfortunately, setting up a data collection and measurement framework to capture data and the real effects of a specific network on a model is prohibitively challenging and expensive. Existing network infrastructures lack efficient mechanisms to collect and process sufficient high-quality data to feed and train a ML model.

With limited existing data services, developing standard protocols and mechanisms to facilitate training and maintaining a data-driven solution in a network has become crucial. In contrast to today's data-driven deployments that train models on synthetic data, we propose OpenData, a generic framework to collect and process network data on the fly, which enables a data-driven solution to train and tune its model in the target network. To demonstrate the benefits of using the OpenData framework, we study the effect of using online networking data to train a QoS forecasting model in our worldwide overlay network. Our primary results show that online training improves a model's accuracy, and it partially addresses data drift and data scarcity problems.

However, by conducting a literature review and carefully investigating different ML paradigms, networking scenarios, and current data-driven solutions, we realize that designing a practical general framework that is capable of providing online data for all types of data-driven applications is likely impossible. Data collection and processing for data-driven applications is highly domain-specific and a case-by-case data collection approach is required. In addition, we observe that most current practices collect internal data and do not need an external framework to acquire the required data. We propose possible directions to partially address the main challenges while providing the main benefits to train ML models in a network.

The rest of this paper is organized as follows. Section II provides background and motivation on networking data-driven solutions. Section III conceptualizes the OpenData framework and our primary observations of using online data to train a QoS forecasting model. Section IV summarizes our exploration of a general framework for online training. Section V provides possible directions to address the main designing challenges. Finally, Section VII concludes the paper.

II. Background and Motivation

A. Current Practice of Building ML Models for Networking

ML starts with data. In general, networking data collection can be achieved in two phases: offline and online. Offline data is gathered from a large amount of historical data and online data is the real-time network data. A ML system trains

a model to gradually learn the relationships between previously observed data and recognize certain types of patterns to predict unseen data. Data-driven algorithms demand large amounts of high-quality data to be effectively trained. There are three main streams to collect massive amounts of networking data: using traffic generators on available datasets, network simulators, or network emulators. The main motivation to generate synthetic training data comes from the difficulties and challenges in collecting training data from a real production network, such as:

- Obtaining real-world data traces is difficult due to the critical and private nature of network traffic.
- The process of traffic capturing and data collection is often expensive, complicated, and time-consuming.
- Acquiring network data comes at the cost of increased monitoring overhead.
- Data collection for learning may disrupt applications' performance and users' experience.

After collecting proper training data, there are also internal processes to deal with the preparation of the data such as normalization so that a model can learn as effectively as possible. During the training process, the model's hyperparameters are iteratively changed until the hyperparameters with the lowest possible loss are discovered. Historically, the process of hyperparameter optimization may have been performed through trial and error, but now, optimization algorithms such as Bayesian optimization are used to rapidly assess hyperparameter configuration to identify the most effective settings. The automated optimization process enables a network controller such as an Software-Defined Networking (SDN) controller to offer optimization services to ML applications [4]. After training a model with a proper amount of data, it will be deployed in a production environment. A deployed model will bring much more value if it's trained with data that fully capture the production environment. Poor training data, on the other hand, can lead to ineffective models [5].

B. Benefits of Using Online Networking Data

Computer network research has long depended on a number of techniques from simulation, emulation, and small-scale laboratory experiments to large-scale testbeds such as Emulab and PlanetLab. While these tools all have a role, experience has shown that networks are inevitably more diverse and variable than we anticipate. However, even with data collected from real networks, the differences in the network structures and architectures can lead to inaccuracy in training a model with data from other networks. For instance, the enterprise network of one organization is diverse and disparate from another. Therefore, the patterns learned from one network may not be applicable to another network. Therefore, only building and training a model with proper data collected from the target network can reveal the true performance of a data-driven solution. The main benefits of using online networking data are:

- It represents the real nature of a network. The randomness in simulated data may bring difficulties to train ML models, in particular Reinforcement Learning (RL), in which the randomness in data brings variance and noise to rewards.
- It captures the full dimensionality of each network. Each network has several dependencies and features which is not possible to capture in other networks, simulators, and emulators.
- It provides the dynamic pattern of a particular network and its continuous growth in the number of applications and the kinds of devices connected to it.
- Most emulated or prototype networks use a very simple network topology. However, a model must learn using the traffic generated over the target network topology including a collection of links, LANs, routers, middleboxes, etc.

C. The Main Challenges of Online Training

After deploying a trained model in a production network, a practical system must be able to dynamically and periodically retrain on new data. Although technological advances in networking, such as Software-Defined Networking (SDN) or programmable switches, have improved the applicability of deploying a ML model to collect online data in a network, due to the lack of a standard ML-friendly framework or protocols, applying real-time data collection and online training involves solving a number of new and nontrivial challenges:

- Labeling. Labeling presents two significant challenges: the difficulty of labeling data in real-time and the time disparity between features and labels. To illustrate this, let us consider a scenario involving a CDN cache management system, where an object is initially requested but remains unaccessed for a span of five hours before being accessed again. The acquisition of training data for this particular problem becomes inherently complex due to the substantial temporal gap between the occurrence of features and the corresponding labeling. During this extended five-hour period, the object's features persist and undergo continuous variations. However, retaining temporal information for every unlabeled object throughout this entire duration would impose excessive memory overhead and make it impractical in practice.
- Data scarcity. Data-driven systems must consider a large amounts of dynamic information to make decisions. However, testing on individual users requires each user to generate a tremendous number of connections but a user may generate little data (e.g. only visit a Website a few times).
- Computational, memory, and bandwidth overhead. While more data features can improve training quality, the allocation of resources for data collection can hinder the core functionality of the system. Often times the data-driven applications have to mitigate the processing from client devices to a server-side system which increases the client's bandwidth and power usage. In addition, generating data for learning requires testing configurations and may disrupt the user's performance.
- Scalability. Collecting massive amounts of data from distributed clients and processing that data on a large

parallel system is a complex process. This extensive and tedious task can cause difficulty in deploying a scalable data-driven application in a dynamic production network.

III. OpenData Framework

In this section, we present OpenData, a comprehensive framework designed to facilitate data collection and processing for online training and tuning of data-driven solutions in a network. OpenData provides protocols and APIs that enable data-driven solutions to seamlessly collect and process online data features from a production network.

A. Overview

Figure 1 provides a conceptual visualization of the OpenData framework. It comprises three layers: the Application Layer, the Data Layer, and the Infrastructure Layer. At the heart of the framework, the OpenData controller resides in the Data Layer, acting as a strategic control point. It leverages data collection interfaces in the Infrastructure Layer to gather data from the network and offers various services, including data collection, data processing, and data aggregation, through its APIs. The Application Layer encompasses diverse application services that cater to users' requirements. A machine learning (ML) solution deployed at the Application Layer can effortlessly collect real-time data from the network to train, re-train, and tune its model hyperparameters, ensuring adaptability to network changes. By abstracting away the complexities of data collection and processing in the network, the OpenData framework empowers model developers to focus on model development, eliminating the need for them to directly deal with the intricacies of data collection and processing within the network.

To address the challenges outlined in previous section and showcase the potential of OpenData, we highlight its key features:

- **Data Drift and Data Scarcity:** OpenData overcomes the limitations of relying on synthetic datasets or embedded data collection mechanisms by providing direct access to live network data. By collecting data features from the production network in real-time, data-driven solutions can continually train and update their models to adapt to changing network conditions. This specially addresses the challenge of data drift and enables models to stay relevant and accurate over time. Moreover, OpenData enhances data availability by offering a centralized framework for data collection and aggregation which it mitigates the issue of data scarcity and facilitates the utilization of diverse and comprehensive datasets for training and tuning.
- **Computational Efficiency:** The OpenData framework streamlines data collection and processing, optimizing computational efficiency in data-driven applications. By leveraging the centralized OpenData controller and its data collection interfaces, data-driven solutions can efficiently retrieve and process network data without incurring unnecessary overhead. This reduces the computational burden on individual applications and ensures that computational resources are efficiently utilized for model training and inference.
- **Scalability and Responsiveness:** OpenData addresses the architectural challenges of scalability and responsiveness by providing efficient protocols and APIs. These enable seamless communication and coordination between the Application Layer, Data Layer, and Infrastructure Layer, ensuring high scalability and responsiveness of the overall system. By leveraging OpenData's scalable design, data-driven solutions can effectively handle large-scale data collection and processing tasks in real-time network environments.

Here, we first demonstrate the benefits of using online data to train a model in a production network before investigating the data demand by networking applications in Section IV.

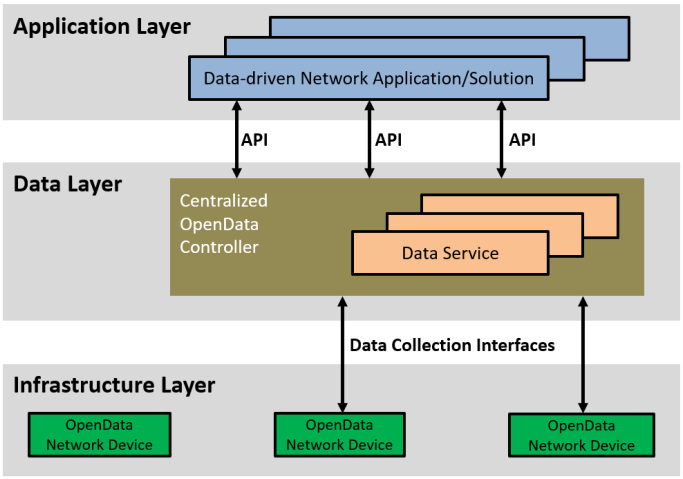


Fig. 1: OpenData Framework

IV. Our exploration of a general framework for online training

Data-driven solutions start with data and the main focus is around the data. A data-driven application must continuously test, verify, and monitor the input data since the behavior of a model is dependent on input data features [6]. As we observed in the previous section, the quality of networking data used to train a model can significantly affect its performance in a production network. To build a general framework that is able to collect proper data for various networking data-driven solutions, the data demands by these applications must be identified. The networking data can be represented as local or network-wide data. The local representation of networking data is necessary to drive local actions such as congestion control or adaptive bitrate algorithms, while the network-wide data is required for network-wide solutions such as routing and traffic classification. In this section, we conduct a literature review to identify the common data demands for training and testing data-driven solutions in wide-area networks.

A. Data features required by different domains is diverse

Table I represents the data demand for congestion control, video streaming, resource management, and traffic engineer-

TABLE I: Networking Data Demand

Local				Wide-area			
Domain	**Solution**	**Model**	**Networking Data Demand**	**Domain**	**Solution**	**Model**	**Networking Data Demand**
Congestion Control	Remy	SL	ACK packets RTT	**Resource Management (SDN Network)**	NFVdeep	DRL	Network topology Links bandwidth Links latency
	PCC Vivace	SL	Throughput RTT Loss rate	**Resource Management (Cloud Network)**	Decima	RL	Network topology
	Aurora	RL	Throughput Latency Loss rate	**Resource Management (General)**	DeepRM [7]	RL	Estimated bandwidth
	Orca	RL	ACK packets Congestion Window Throughput Loss rate Delay	**Flow Scheduler**	AuTO	DRL	Flows 5-tuples Flows throughput
	DRL-CC	DRL	Congestion Window Throughput Goodput RTT Jitter		DeepPacket	DL (CNN)	Flows duration Flows throughout Inter-flows statistics
Video Streaming	Pensieve	RL	Throughput	**Routing**	Learning to Route	RL	Flows 5-tuple Flows throughout Inter-flows statistics
	Swift	DRL	Throughput		RouteNet	SL (GNN)	Network topology Links latency Flow Delay Flow Jitter
	ABRL	RL	Estimated bandwidth		DQRC	RL	Links latency Packet delivery time
	LiveNAS [8]	DNN	Estimated bandwidth	**Packet/Flow Classification**	NeuroCuts [9]	DRL	Flows 5-tuples
Resource Management (CDN Network)	LRB	SL	-		NuevoMatch	DL (NN)	Flows 5-tuples
	LFO	RL	-		ODCNN	DL (CNN)	Flows 5-tuples Flows statistics Inter-flows statistics

ing. In the following paragraphs, we briefly highlight the common demands for each domain.

Congestion Control. Several heuristic techniques have been proposed to address congestion control, including Vegas, NewReno, FAST TCP, CUBIC, and BBR. These mechanisms usually apply end-to-end techniques to tune the congestion window (cwnd) as a mean to achieve better performance. The main limitation of traditional approaches is the unrealistic assumptions regarding network conditions. This is where applying machine learning techniques becomes useful. As we see in Table I, the common networking data demand for congestion control includes throughput, loss rate, delay (or RTT), ACK messages, and cwnd.

Video Streaming. Considering the importance of users' quality of experience, the dynamic nature of networks, and the complex nature of optimization objectives (including re-buffering ratio, average bitrate, number of quality changes, startup delay, etc.), many new data-driven solutions have been explored to improve adaptive video streaming. As seen in Table I, the only network-related feature that a data-driven ABR solution needs to perceive from the underlying network is throughput or an estimation of available bandwidth. The client-side ABR agent estimates the bandwidth using the download time of the last video chunk data. Unlike traditional ABR algorithms, data-driven solutions use server-side techniques which mitigate the computational and memory overheads from client devices to the server-side.

Resource Management. Resource management problems are ubiquitous across computer systems and networks. The majority of these problems are solved today using heuristic solutions that could become insufficient and unreliable if an aspect of the problem changes [7]. Data-driven resource management solutions have been proposed to provide higher scalability, availability, and cost-efficiency of resources. These solutions mainly map resource demands to resources, in order to meet resource management optimization objectives such as improving resource utilization, reducing average response time, and enhancing fairness. From Table I, it can be seen that the data demand depends on the scope of a resource management problem, which could be within a CDN, SDN, cloud network, or across geo-distributed cloud data centers. While a data-driven CDN manager normally does not perceive any networking data to improve its hit rate, the most common data demand within other domains is network-wide data which includes underlying network topology, links latency, and throughput between every two nodes in the network.

Traffic Engineering. Traffic Engineering (TE) enables a network operator to steer traffic through a path that may result in more efficient use of network resources [10]. The main challenge of TE involves the ever-increasing dynamics of traffic loads, network characteristics, and multi-faceted optimization goals which are difficult to be interpreted as a simple formula for handcrafted heuristics to optimize [6]. As summarized in Table I, data-driven TE solutions need aggregate information for each network flow, such as flow throughput, average packet loss, average delay, number of packets, etc. A flow is typically

defined as all the packets that share the 5 tuples of the same flow (source and destination IP addresses, port numbers, and protocol) and have an inter-arrival time within a specific time slot. TE algorithms also demand network-wide information such as network topology, nodes' throughput, and links' load and latency. Table I shows more uses of deep learning and neural networks in TE algorithms.

B. The overhead for an external tool to collect required data

In the previous section, we aimed to understand the networking data demands by data-driven applications. Our main observations are as follows:

1) There are only a few common data features across different domains of solutions. This indicates that a case-by-case domain-specific data collection approach is required for each problem domain.
2) Most training data is collected by a solution internally and can be more difficult to be obtained by external tools. For example, video streaming collects most data internally from the video player, and a congestion control mechanism collects data from the transport protocol. This indicates that most ML applications are able to collect internal data and there is no need to collect data from an external framework.
3) RL is widely adopted. This highlights the fact that due to the lack of labeling and the time disparity between observed features and the label, RL approaches are more desirable in networking scenarios [11]. In addition, by using a RL learning approach, the quality of training data for building a model is not as critical compared to a supervised learning model. An RL solution can learn in the environment and rectify its decision-making algorithm after being deployed. It also does not require the generation of a label for features since the reward function provides feedback on each given action. However, studies show that pre-training a (D)RL learning model on poor quality data can result in a confounded model in a real environment. As well, it increases the time a model consumes to learn to perform better in the earlier stages of deployment.

Taken together, these observations indicate that while individual problems need a better data collection and processing mechanism, designing a generic framework to support all data demands for all data-driven applications is likely impossible.

V. Possible directions

Inspired by the main motivations behind designing a generalized framework, we focus on possible instantiations of OpenData frameworks. We classify them into three possible directions: domain-specific, model-specific, and data-assist frameworks. Here we provide more details on each category.

A. Domain-specific solution

A domain-specific solution leverages client-side applications or a centralized network controller within the framework to automate the development of data-driven applications for specific networking domains, such as video streaming or traffic classification. These applications collect training data either locally or by importing data from the framework, which facilitates data collection and processing. Domain-specific insights and algorithms can be incorporated into a domain-specific solution to help automate the labeling process, reducing the difficulty of labeling data on the fly, and efficiently managing the time disparity between features and labels. In congestion control, for instance, specific heuristics and techniques like leveraging ACK packets and round-trip time (RTT) measurements can be used to effectively label data and manage the time disparity by setting labels within a designated time-scale window. Similarly, in video streaming applications, a domain-specific solution can integrate video player analytics to collect local data on video quality, buffering events, and network conditions.

Furthermore, the domain-specific solution optimizes computational, memory, and bandwidth overhead by incorporating resource-efficient data collection and processing mechanisms. This ensures a balance between data-driven functionalities and system performance. The framework can employ distributed data collection strategies and leverage large-scale parallel processing systems to handle the substantial amounts of data generated by distributed clients to enhance scalability and overall efficiency of data-driven applications. Ultimately, this domain-specific approach improves the performance of data-driven models in specific networking domains.

B. Model-specific solution

There are three main machine learning paradigms (SL, UL, and RL), and each has been branched into several specific types. These learning paradigms have distinct building and training processes, necessitating a framework that provides model-specific services to facilitate online data processing, training, and model maintenance. For instance, a framework can offer graph data structures and libraries specifically designed for deploying graph neural network (GNN) models.

This model-specific approach optimizes the performance of data-driven models within networking domains by leveraging techniques and algorithms tailored to the characteristics of each learning paradigm. In the context of reinforcement learning (RL) applications, a model-specific solution can provide services that facilitate efficient data processing and RL model training. These services may include specialized algorithms for reward shaping, exploration-exploitation strategies, and model update mechanisms. Similarly, in supervised learning (SL) or unsupervised learning (UL), the model-specific solution can offer feature selection techniques, data augmentation methods, and algorithms designed to accommodate the unique features of networking data. By providing these services, the solution streamlines the training process, reduces computational overhead, and optimizes the performance of models in networking scenarios.

C. Data-assist solution

In a nutshell, A data-assist solution represents an approach where a dedicated framework enables a data-driven application

to aggregate its local data with data obtained from other network elements. This framework provides mechanisms for efficient data collection, data synchronization, and data fusion to create a more comprehensive and diverse dataset. For example, consider the scenario of optimizing CDN and bitrate selection for individual video sessions. By providing a global view of video quality, the data-assist solution allows the model to make informed decisions regarding the selection of CDN and bitrate. This holistic approach takes into account diverse factors such as network congestion, user preferences, and quality metrics from various sources. By aggregating data from multiple sessions and network elements, the model can benefit from a broader perspective and make more accurate decisions.

The data-assist solution offers several advantages, particularly in situations where decision quality exhibits significant spatial diversity or temporal variability, and where individual decisions may lack sufficient data for reliable outcomes. By aggregating data from multiple sources, the solution enhances the availability and diversity of training data, overcoming limitations posed by data scarcity for individual decisions. This approach improves the model's accuracy by incorporating a wider range of network conditions, user behaviors, and system dynamics.

Through the centralized aggregation of data, the data-assist solution empowers the model to make more informed decisions, optimize resource allocation, and deliver enhanced performance in complex and dynamic network environments. By leveraging the collective intelligence of multiple network elements, the solution facilitates better decision-making, reduces computational costs, and produces more interpretable models. It offers a powerful means to harness the full potential of data-driven applications in network domains and optimize their performance.

VI. Case Study: QoS Forecasting Model

To demonstrate the effect of using a framework to collect and provide online networking data, we train and test a Quality of Service (QoS) forecasting model in our production overlay network. Our overlay network hosts various large-scale services and is built on an infrastructure of connected centers through high bandwidth wide area networks. The current interest towards adaptable and re-configurable systems in dynamic environments such as wide-area overlay networks has resulted in a need for an automatic QoS detection system. QoS forecasting models [12] have been presented as a promising technique to predict future QoS values in supporting and improving proactive network management systems [13]. Our model utilizes past networking data and generates future predictions for the features of latency, packet loss rate, and jitter. This allows the study of online data on the performance of a data-driven solution in a production network.

A. Methodology

We deployed 12 nodes in 12 geographically different locations in our worldwide overlay network. To generate data, each node used the Ping tool to send 10 ICMP ping messages to another node in 10-second time intervals. Then, data was recorded by logging the results of the ping which contained source ID, destination ID, timestamp, packet loss rate, average RTT, and average jitter. As shown in Table II, we conducted two sets of experiments and collected two datasets. *Dataset A*, which includes about two million connection logs, was collected for a period of four days from May 13, 2021, to May 16, 2021. Two months later, *Dataset B*, which includes one million connection logs, was collected for a period of two days from June 20, 2021, to June 21, 2021. We trained the QoS forecasting model in three different scenarios: *Model 1* was trained only over Dataset A, *Model 2* was trained only over Dataset B, and *Model 3* was first trained over Dataset A and then re-trained over Dataset B. We tested all three models over unseen data from Dataset B, where each test case was over a pair of source and destination nodes.

TABLE II: Datasets collected from an overlay network

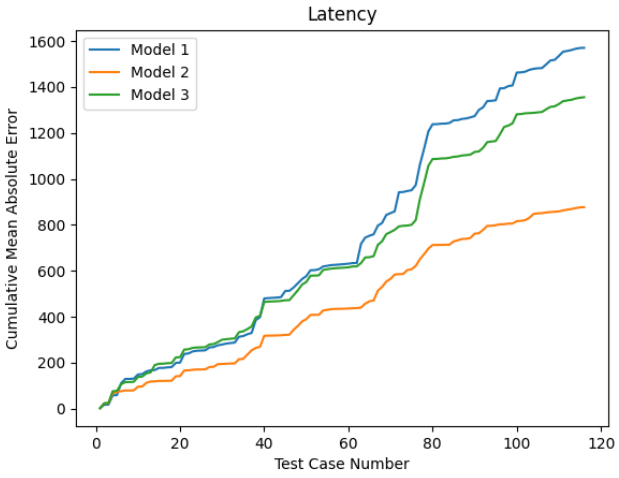
Dataset	# Nodes	# Logs	Duration	Interval
Dataset A	12	1955756	Four days	10 sec
Dataset B	12	1087349	Two days	10 sec

We deployed a network of 12 nodes located in geographically distinct areas within our worldwide overlay network. The data generation process involved each node utilizing the Ping tool to transmit 10 ICMP ping messages to another node at intervals of 10 seconds. Subsequently, we recorded the ping results, which encompassed essential information such as the source ID, destination ID, timestamp, packet loss rate, average round-trip time (RTT), and average jitter. The data collection process yielded two distinct datasets, namely *Dataset A* and *Dataset B*, as outlined in Table II. *Dataset A* was amassed over a four-day period, spanning from May 13, 2021, to May 16, 2021, and consists of approximately two million connection logs. In contrast, *Dataset B* was gathered two months later over a two-day period, specifically from June 20, 2021, to June 21, 2021, and comprises one million connection logs. Our QoS forecasting model was trained in three distinct scenarios: *Model 1*, which exclusively employed *Dataset A* for training; *Model 2*, trained solely on *Dataset B*; and *Model 3*, initially trained on *Dataset A* and subsequently retrained using *Dataset B*. To assess the performance of these models, we conducted testing on unseen data from *Dataset B*, with each test case involving a pair of source and destination nodes.

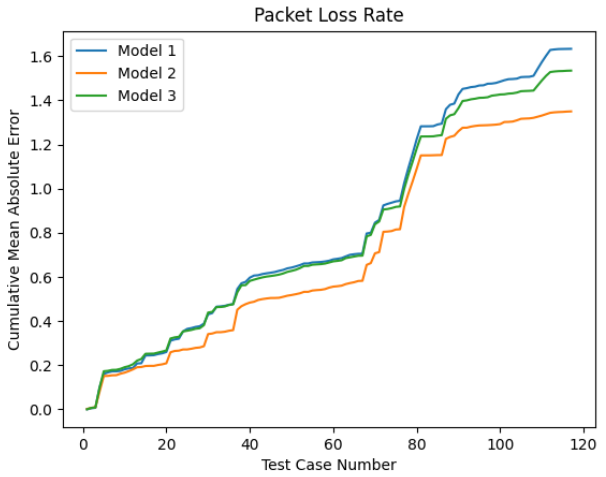
To facilitate the machine learning (ML) approach, we employed a Long Short-Term Memory (LSTM) model as the underlying model architecture. The features utilized for training the model consisted of the source ID, destination ID, timestamp, packet loss rate, average RTT, and average jitter. Notably, we leveraged a history log length of 100 and a timestep of 20 seconds to capture the relevant temporal dependencies within the data.

B. Preliminary Results

Figure 2 represents the results of 116 different test cases on predicting latency (Figure 2a) and packet-loss rate (Figure 2b). Each test case was conducted on a random pair of source and destination nodes. As we see, Model 2, which was trained only on Dataset B, outperforms the other two models in



(a) Latency



(b) Packet Loss Rate

Fig. 2: Performance of three QoS forecasting models

predicting latency and packet-loss rate in most test cases. Model 1 performed the worst among these models, which indicates the vulnerability of data-driven systems to the data drift problem [14]. The data drift problem is a result of a mismatch between training data in the past and test data in the future, and it can lead to significant performance degradation and system inefficiencies [14]. Interestingly, Model 3 only performed slightly better than Model 1, which indicates that re-training a model with new data does not necessarily adapt the model to the environment changes.

Another observation from Figure 2 is that Model 2 performs more consistently than the other two models. We observed that Model 3 has a similar standard deviation of mean absolute error (MAE) to Model 1, whereas the standard deviation for Model 2 is less than half of the other two models. The final observation is the prediction performance of Model 2 for packet-loss rate. Since 87% of the packet-loss data in our dataset is 0, it is very difficult for a model to infer the future packet-loss rate. However, as we see in Figure 2b, Model 2 can significantly outperform the other two models in most test cases. This observation is aligned with [8] that proper data can address the data scarcity issue.

VII. Conclusion/Future Directions

In recent years, the effect of ML has been felt across computer networks. In current practice, synthetic datasets or data collected from other networks are used in the training and validation of ML models, and their applicability in practical settings remains questionable. To this end, we presented OpenData, a generalized framework to facilitate online training in a production network. We argue that OpenData has the potential to significantly improve a model's performance through harnessing online data. By training a QoS forecasting model with old and fresh networking data, we observed that fresh data can partially address the data drift and data scarcity problems. However, we identified several key challenges to design a generic framework to support all data demands for all type of data-driven applications. Finally, we proposed possible directions to provide a broad research guideline on networking with machine learning to motivate researchers in developing innovative algorithms, standards, and frameworks. There are also several architectural challenges that a data-driven framework like OpenData must address. For example, this framework must be scalable (i.e., scale to tens of millions of concurrent sessions per second) and responsive (i.e., respond to every client request within at most tens of milliseconds) [15]. We believe by developing frameworks and protocols that provide services to develop ML applications, we can unleash the unharnessed potential of data-driven solutions in computer networks.

References

[1] T. Eliyahu, Y. Kazak, G. Katz, and M. Schapira, "Verifying learning-augmented systems," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 305–318.

[2] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, 2018.

[3] B. Arzani, K. Hsieh, and H. Chen, "Interpretable feedback for automl and a proposal for domain-customized automl for networking," in *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*, 2021, pp. 53–60.

[4] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A survey of networking applications applying the software defined networking concept based on machine learning," *IEEE Access*, vol. 7, pp. 95 397–95 417, 2019.

[5] N. Wu and Y. Xie, "A survey of machine learning for computer architecture and systems," *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–39, 2022.

[6] M. E. Kanakis, R. Khalili, and L. Wang, "Machine learning for computer systems and networking: A survey," *ACM Computing Surveys (CSUR)*, 2022.

[7] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM workshop on hot topics in networks*, 2016, pp. 50–56.

[8] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han, "Neural-enhanced live streaming: Improving live video ingest via online learning," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 107–125.

[9] E. Liang, H. Zhu, X. Jin, and I. Stoica, "Neural packet classification," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 256–269.

[10] T. D. Nadeau, *MPLS network management: MIBs, tools, and techniques*. Elsevier, 2003.

[11] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 495–511.

[12] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting qos attributes of web services based on arima and garch models," in *2012 IEEE 19th International Conference on Web Services*. IEEE, 2012, pp. 74–81.

[13] G. White, A. Palade, and S. Clarke, "Forecasting qos attributes using lstm networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[14] A. Mallick, K. Hsieh, B. Arzani, and G. Joshi, "Matchmaker: Data drift mitigation in machine learning for large-scale systems," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 77–94, 2022.

[15] J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "Unleashing the potential of data-driven networking," in *International Conference on Communication Systems and Networks*. Springer, 2017, pp. 110–126.