IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 16, 2021, accepted October 1, 2021, date of publication October 5, 2021, date of current version October 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3117763

Next-Generation Data Center Network Enabled by Machine Learning: Review, Challenges, and Opportunities

HAIWEI DONG[1], (Senior Member, IEEE), ALI MUNIR[1], HANINE TOUT[1], AND YASHAR GANJALI[1,2]

[1]Data Center Network Laboratory, Huawei Technologies Canada, Waterloo, ON N2L 0A4, Canada
[2]Department of Computer Science, University of Toronto, Toronto, ON M5S 2E4, Canada

Corresponding author: Haiwei Dong (haiwei.dong@huawei.com)

ABSTRACT Data center network (DCN) is the backbone of many emerging applications from smart connected homes to smart traffic control and is continuously evolving to meet the diverse and ever-increasing computing requirements of these applications. The data centers often have tens of thousands of components such as servers and switches/routers that work together to achieve a common objective and serve these applications. Managing such large data centers is a tedious process and demands automation, intelligent control and decision making within the data center. Recently both the industry and academia have focused on bringing intelligence to the control, automation, and management of DCNs. Despite the variety of works that surveyed ML for networking, to the best of our knowledge, none has focused on DCN, which makes this survey original. Readers in the academic and industrial communities will all benefit from a comprehensive discussion of the ML solutions applied in DCN to address critical essential problems, including workload forecasting, traffic flow control, traffic classification and scheduling, topology management, network state prediction, root cause analysis, and network security. Furthermore, this article outlines the challenges and concludes with the future research venues in adopting ML for automatic, intelligent and autonomous DCNs.

INDEX TERMS Data center network, machine learning applications, survey.

I. INTRODUCTION

Data center network (DCN) hosts multi-tenant and multi-objective applications with ever-growing compute and communication requirements. The data centers often have tens of thousands of components such as servers and switches/routers that work together to achieve a common objective and serve these applications. In other words, the network is evolving continuously and the dynamics and large scale of the network impede the application of fixed and manual methods for DCN management and control. Managing such large data centers is a tedious process and demands for automation, intelligent control and decision making within the data center.

Machine learning (ML) analysis is the process of examining data in a system and deducing knowledge out of it. Recent developments in ML have made these techniques applicable, adaptable and robust in countless real-life scenarios, which range from mundane to exceptional ones. ML has become an essential critical facet in many systems including but not limited to smart homes, healthcare, robotics, edge computing, cybersecurity, wireless communication, autonomous vehicles and Internet of Things [1]–[3]. Besides the advancements in ML techniques, the massive amount of data available for analysis in today's world is another factor that contributes to the resurgence in ML solutions, which not only are capable of identifying hidden patterns in data, but also able to learn and understand the systems where data is generated. In addition, the breakthrough in computing, like graphic and tensor processing units, is offering adequate storage and processing capabilities for ML models' training and inference, whereas ML models can even run on resource-constrained hardware through lightweight ML deployment versions.

While ML has been applied in various areas and proved its capability, it has not been deployed ubiquitously to optimize and manage DCN since the latter is challenged by the data that can be collected and the control actions that can be applied on legacy devices in the network. However,

The associate editor coordinating the review of this manuscript and approving it for publication was Rentao Gu.

the developments in networking through software-defined networking, can alleviate these impediments and encourage the cognitive ML systems to be built for automating the management and control of DCN. Applying ML in DCN is an interesting research area as it can aid in solving many complex DCN problems [4], yet it requires a well understanding of DCN problems, ML techniques and challenges in relevant to both networking and machine learning applications.

The primary goal of this work is to provide a holistic overview of the body of knowledge on ML methods in support of DCN. We also complement the discussion with key insights into the ML techniques employed, their efficiency and limitations, and their challenges in DCN. We further identify open promising opportunities while we draw our vision for ML-enabled future DCN. Our contributions are summarized as follows:

- We provide a comprehensive review of ML approaches in DCN.
- We discuss the features, techniques, efficiency and feasibility of ML techniques in DCN.
- We identify the key challenges of ML in DCN.
- We propose future research opportunities and draw our vision pertaining to the future generation of automatic, intelligent and autonomous DCN.

II. DATA CENTER NETWORKING

A data center network (DCN) [5], depicted in Figure 1, is a complex arrangement of the constellation of networking resources such as switches, routers, and interfaces. DCN interconnects a variety of computing and storage entities in a data center pool of resources to ensure a high level of performance, storage and processing of applications, services and data. DCN holds a pivotal role in a data center, as it interconnects all of the resources together, hence it has to be scalable and efficient to connect tens or even tens of thousands of servers to handle the growing demands of cloud computing, edge/fog computing and Internet of Things.

Technological advancements require data centers to support programmability and adaptability, which has been realized in the form of a software-defined network (SDN). SDN allows the network administrator to dynamically change the network configuration and workflows to accommodate for the fluctuating workloads more efficiently and effectively. SDN has two main components: a data plane that manages the forwarding and control of flows in the switches and routers, and a control plane that defines the set of policies on how to handle data from different flows. An SDN controller defines the control plane policies for changes in data plane mapping, while a workflow is in progress, without risking the connections that bind the network elements together.

A. CHALLENGES IN DCN

1) DATA CENTER TOPOLOGIES

A DCN topology is the key enabler for many applications and their performance. A key factor in the design of DCN topology is the deployment cost. As a result, many different DCN topologies (such as clos [6], Fat-Tree [7], BCube [8], portland [9] etc.,) have been designed to minimize deployment cost, provide uniform high bandwidth to its applications, and to provide performance isolation. For example, a very common Fat-Tree consists of a three-layer topology (edge, aggregation and core) and can be built using cheap switches with uniform capacity. (1) The access layer where switches are connected to servers and located on top of the rack, (2) The aggregation layer where aggregation switches connect to access switches to provide a variety of services like network analysis and firewall, combining response from access switches, and (3) The core layer where its switches provide intra and inter data center connectivity. A key benefit is that – Fat-Tree has identical bandwidth at any bisection, and hence all the sources can send data at maximum link rates. With the emerging applications, it becomes challenging to design DCN topologies that can meet application demands. However, with new types of DCN topologies, it gets increasingly complex to automate the management and control of traffic within the data center.

2) UNPREDICTABLE TRAFFIC MATRIX

A traffic matrix denotes the volume of communication between all pairs of sources and destinations in a network. Yet, the traffic matrices inside a data center change rapidly and unpredictably and are highly divergent. This is an important problem to address and a key challenge that complicates the optimization of the network performance and capacity planning.

3) MIX OF FLOW TYPES AND SIZES

On account of a variety of multi-tenant applications sharing the data center infrastructure, a mix of flows with different types and sizes are produced. Some of these flows might be short, constrained with a deadline or requiring low latency, whereas others can be large and throughput-oriented, requiring a high transfer rate. Moreover, the flow sizes might be unknown for specific applications. Such characteristics complicate the flow scheduling process over a shared link for the following reason. It needs to achieve a faster completion time to reduce communication delays and improve application responsiveness while taking into account the different requirements of flows simultaneously.

4) TRAFFIC BURSTINESS

Many applications and services like live video broadcasting and interactive distributed gaming have bursty traffic, meaning that instant data transmission varies quickly. Nevertheless, the on and off cycles of bursty traffic may incur undesirable effects on the quality of service of the network, leading to delays, congestion and even idleness. Spikes in the traffic might impose overloading which leads to longer delays for applications that can only tolerate very small latency like web browsing. Therefore, it is still challenging to put forward

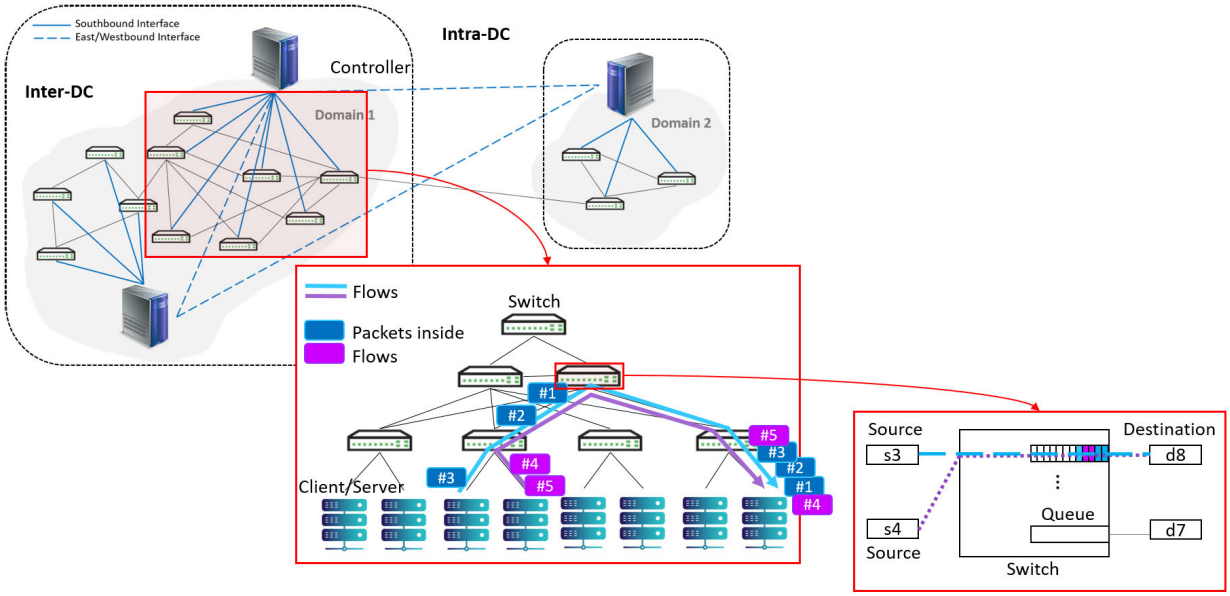

FIGURE 1. Data Center Network: comprises of several key elements and various communication interfaces within (Inter-DC) a single data center and between (Intra-DC) different data centers, each having their own network domains (such as Domain1, Domain2, etc.). Each Domain consists of the connections between the switches and servers and the flows and packets flowing between sources (s3, s4) and destinations (d7, d8). Each switch has multiple queues to serve those packets.

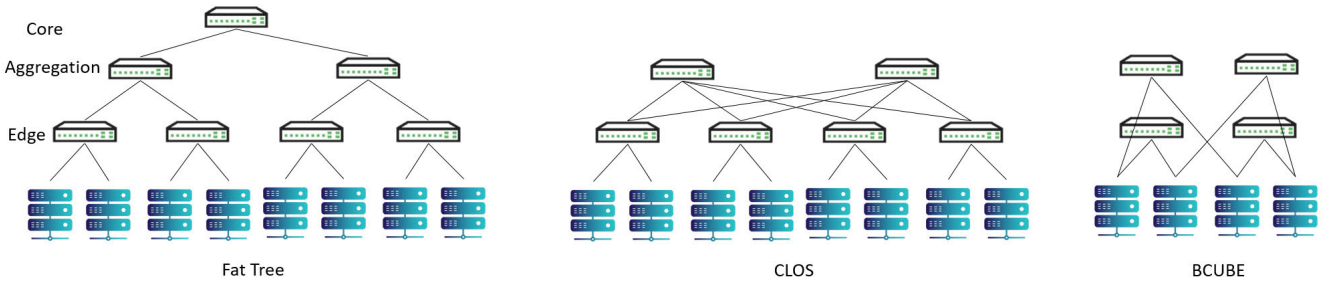

FIGURE 2. Data Center Network Topology: Many different DCN topologies (such as CLOS, Fat-Tree, BCube) have been proposed to minimize deployment cost, provide uniform high bandwidth to its applications, and provide performance isolation.

a responsive control mechanism that can manage the buffer space usage and deal with bursts quickly.

5) INCAST

Data center network supports heterogeneous applications such as data storage, computation-intensive, social networking, web and videos data hosting. Although DCN provides links with high bandwidth, low transmission delays and switches with a small buffer size, it supports on the other hand many-to-one communication patterns which can lead to a large number of incoming flows transmitted simultaneously to a single end-point. If not appropriately and proactively controlled, this in turn would overload the switch buffer leading to congestion, packet loss and higher latency with throughput reduction.

B. WHY ML IS NEEDED FOR DCNs?

Next Generation DCNs are built to provide greater bandwidth, increased throughput, diminished latency, and many more advanced techniques that can optimize the cost while maximizing the network usage. Typically, a DCN should create an infrastructure that is stable, secure, and reliable in line with the industry regulations and meet with the organizations/customers/users needs, able to support networking requirements for modern technologies like cloud computing, virtualization, big data and IoT, and should be scalable enough to easily meet the communications needs in peak scenarios.

These next-generation DCNs urge the need for automation, intelligence and autonomy. Traditionally, the network design and management heavily rely on the expertise of the telecom experts and their extensive understanding and knowledge of the network topology, consumer's mobility and the traffic patterns in order to design, construct and configure the management policies that constantly orchestrate the network. However, network topology will grow more and more complex, which makes the management process not only tedious but even non-achievable solely by humans due to unpredictable patterns/behaviors in denser topology. Machine

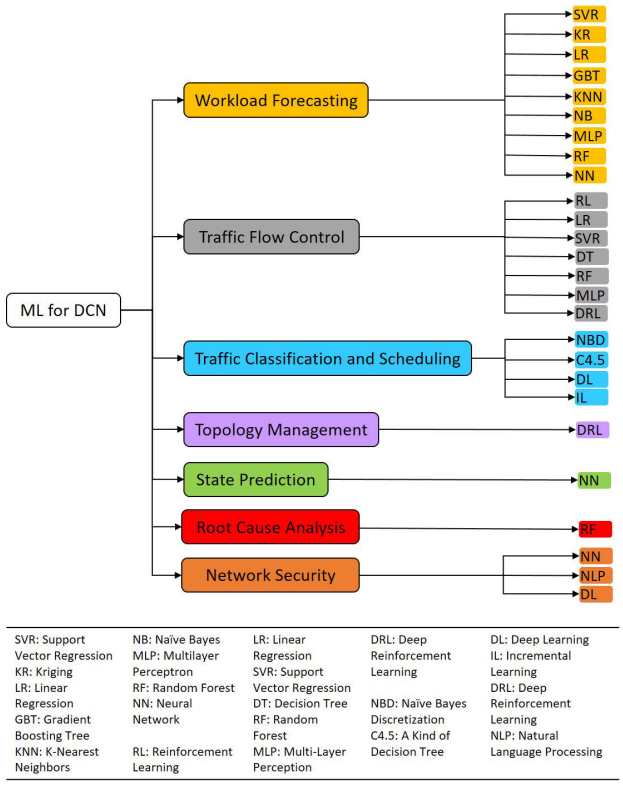

FIGURE 3. Machine Learning for Data Center Networking: Applications of machine learning in DCN are classified into seven areas along with the different machine learning techniques applied in each.

learning is helping achieve many of these objectives as discussed in the next section.

III. ML APPLICATIONS IN DCN

Artificial Intelligence (AI) is playing a fundamental role in assisting operators in rolling out, operating, and managing the DCNs. More and more processes are becoming AI-based, leveraging continuous measurements from the network and machine learning (ML) models for automated and intelligent management and optimization of the network. While AI is certainly becoming a game-changer in building Next-Generation Data Center Networks, we aim in this section to touch on the critical areas in DCN that leverage machine learning. We discuss the machine learning techniques adopted, the data needed for building the learning models, their technical objective and the main evaluation results. We provide hereafter a first high-level taxonomy illustrated in Figure 3, that shows the key technical challenges in DCN that leverage machine learning and the techniques adopted for each.

As you see in Figure 3, there are many types of ML/AI technologies used in DCN. To better understand why a specific method is suitably applied in a specific DCN problem, we classify the ML/AI methods into the following categories:

- *Supervised learning*: The training data includes the features and their corresponding desired results, i.e., labels. The typical problems in supervised learning are regression and classification. The former tries to predict a numeric value and the latter attempts to predict a category that the instance falls in. We note that some regression methods can be applied for classification tasks (e.g., logistic regression is used in classification), and vice versa (e.g., decision tree can be used in regression). The most commonly known supervised learning methods consist of linear regression, logistic regression, k-nearest neighbors, support vector machine, decision tree, random forest, neural network.
- *Unsupervised learning*: The training data only includes features, i.e., unlabeled. The typical problems in unsupervised learning are clustering, dimension reduction, anomaly detection, association rule learning. Clustering tries to identify groups (or subgroups in hierarchical clustering) of similar instances. Dimension reduction tries to reduce the dimension of features without losing much important information. Anomaly detection detects unusual instances which look different from normal instances. Association rule learning reveals the hidden relation between the attributes/features, based on which it makes predictions. The most popular unsupervised learning methods are k-Means, hierarchical cluster analysis, expectation maximization, principal component analysis (with or without kernels), locally-linear embedding.
- *Semi-supervised learning*: The training data contains both labeled samples (usually a little bit) and unlabeled samples (usually a lot). The typical problems are similar to that of supervised learning. For example, unsupervised learning conducts clustering first with a large amount of unlabeled data where the grouped results are processed by supervised learning which has been trained with the limited amount of labeled data. As most of the semi-supervised learning combines supervised learning and unsupervised learning methods, the combination naturally has big flexibility. Some of the semi-supervised learning methods are SGAN (semi-supervised generative adversarial network), S3VM (semi-supervised support vector machine), semi-supervised deep belief network.
- *Reinforcement learning*: There is an agent in a contextual environment which can take actions and observe the outcome of the actions, either a reward or a penalty (negative reward). By using the accumulative reward as feedback, the agent gradually improves its strategy (named as a policy) in taking action. The typical problem suitable for reinforcement learning is decision-making based on long-term episodic experience. The well-known methods include the classical tabular methods (such as policy iteration, value iteration, n-step temporal-difference learning, Sarsa, etc.), approximation-based methods (such as deep Q-network, proximal policy optimization, deep deterministic policy

gradient, etc.), and multi-agent methods (such as Team-Q, Nash-Q, etc.).

The performance measure of ML/AI methods has many quantitative metrics. For regression problems, the typical metric is the root mean square error (RMSE), which is the standard derivation of the prediction error of the ML/AI methods. For classification problems, the performance measure is not as straightforward as that of regression problems. The commonly-used metrics are $precision = \frac{true\ positives}{true\ positives+false\ positives}$ (describing the accuracy of positive predictions) and $recall = \frac{true\ positives}{true\ positives+false\ negative}$ (describing the true positive rate). Although we can use a confusion matrix to list all ratio measure metrics, it is always convenient to combine precision and recall as one metric, called F_1 score, which is defined as $F_1 = \frac{2}{\frac{1}{precision}+\frac{1}{recall}}$. F_1 score tries to solve the tradeoff between precision and recall by favoring the classification with similar values of precision and recall.

A. WORKLOAD FORECASTING

1) PROBLEM DEFINITION

Accurate forecasting of resource utilization in data centers allows consumers to dynamically adjust the leased resources for hosting their applications in a way to maintain the desired performance and quality of service while minimizing their expenses. Furthermore, such accurate estimation enables providers to efficiently maximize the utilization of data center resources while minimizing their operational costs. Forecasting future resources needs helps to achieve efficient capacity planning, workload placement, job scheduling, proactive auto-scaling, and load balancing. On the other hand, an inaccurate estimation leads to either over or under-provisioning of data center resources, as depicted in Figure 4, resulting into wastage of resources, unnecessary power consumption and violations of the service level agreement. A leading challenge in workload forecasting is the presence of multi-tenant co-hosted applications characterized by nonlinear, dynamic and time-varying nature [10]. For instance, at any time, millions of requests could be generated, whereas at the next instance of time, very few requests or even none might be issued, resulting in sudden peaks and rock bottoms in workload patterns. Therefore, a workload forecasting method becomes a significant research problem where the estimation strategy must be capable of accurately estimating future resource needs while adapting to dynamic workload demands in a data center environment.

2) LITERATURE

Iqbal *et al.* proposed a method that can adaptively and automatically identify the appropriate model for resource utilization estimation [10]. It trains a classifier through different scenarios and a corresponding resource estimator for each, in order to learn the best regression model to produce the workload prediction. Classical machine learning classification and regression methods have been adopted for both classifier and resource utilization predictor modules respectively.

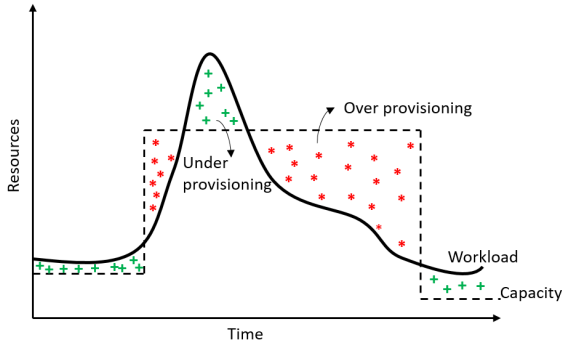

FIGURE 4. DCN topology configuration and management are tightly coupled with the accuracy of workload forecasting. Inaccurate workload forecasting results in under and over-provisioning of resources.

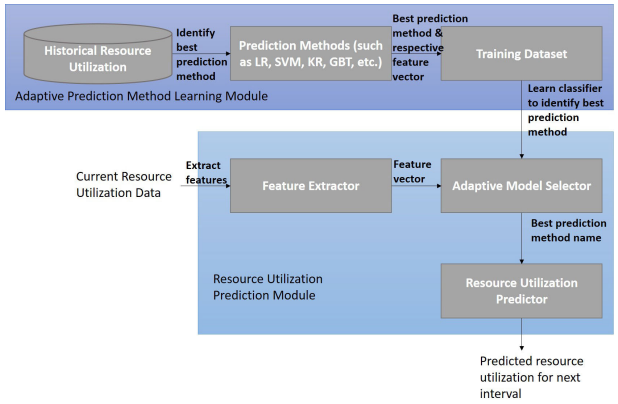

FIGURE 5. Adaptively estimate resource utilization by choosing the most accurate ML model. Various prediction models (such as linear regression (LR), support vector machine (SVM), Kriging (KR), gradient boosting tree (GBT), etc.) are trained by the historical resource utilization data, followed by a random decision forest to choose the most accurate model to be used.

In details as shown in Figure 5, the historical resource utilization data divided by sliding windows are used to train different ML models, such as linear regression (LR), support vector machine (SVM), Kriging (KR), gradient boosting tree (GBT), etc. Both the feature vectors in the aforementioned models and the corresponding best-fitting model name are stored in the training dataset. When the current resource utilization data comes, the feature vector is extracted followed by a model selector which applies random decision forest (RDF) to identify the most accurate model to be adopted. At last, the predicted resource utilization is estimated by the most accurate model. Their key observations showed the significant effect of the window size selection on the estimation accuracy and more noticeably how hard it is to forecast in the presence of bursts.

The work by Li *et al.* focused on predicting the total volume of future incoming and outgoing traffic on the inter-data center link which is typically conquered by elephant flows [11]. The authors applied wavelet transform for the decomposition of raw time series in order to capture both the time and frequency features, whereas elephant flows were added as separate feature dimensions. To mitigate the

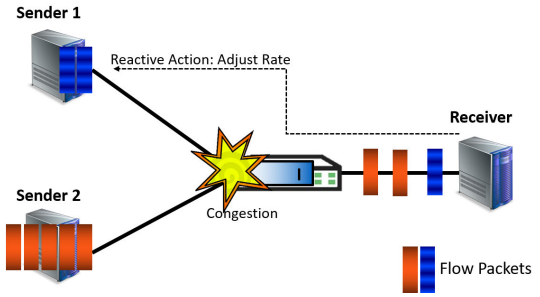

FIGURE 6. Traditional Congestion (Flow) Control: Reactive management strategy of adjusting packets sending rate after the congestion in the switch queue.

significant cost entailed by collecting all elephant flows at high frequencies, information of such flows (including the total traffic and sublink traffic for both incoming and outgoing directions, a sample of elephant flows) was collected at lower frequencies and interpolated to fill in the missing values. A simple artificial neural network (ANN) with one input layer, one hidden layer, and one output layer was used to build the prediction model to handle the non-linearity of patterns where the training optimizer is the classical stochastic gradient descent (SGD). Predicting incoming and outgoing traffic volume is done through the same model by incorporating information about both categories at the training stage. Tested on the inter-DC link at Baidu Internet Company, the combination of ANN and wavelet transform model was able to reduce the internet service provider's peak bandwidth billed utilization by around 9%.

B. TRAFFIC FLOW CONTROL

1) PROBLEM DEFINITION

Data center traffic has diverse communication patterns and requires efficient flow control mechanisms. Current DCN flow control protocols are mainly a part of the congestion control solutions such as TCP, DCTCP, DCQCN. An important challenge in this space is the incast problem, which is created when several hosts send data to a single receiver host. This sudden arrival of flows creates congestion at the switch buffer of the receiver link, as illustrated in Figure 6. While various traffic control techniques are designed to rapidly react to incast flows, they still fail to proactively identify and avoid such events leading to packets and goodput loss. Hence the need to design a proactive technique that predicts the network state by forecasting the future traffic matrices and accordingly adjusts the rates at the sending hosts.

2) LITERATURE

While there is no existing system to compare congestion control solutions for data center networking, Ruffy *et al.* proposed a data center emulator, called Iroko, in this regard [12]. Particularly, it allows studying the needs and limitations of reinforcement learning in data center network to support not only different topologies but also a variety of congestion control algorithms under different deployment scenarios.

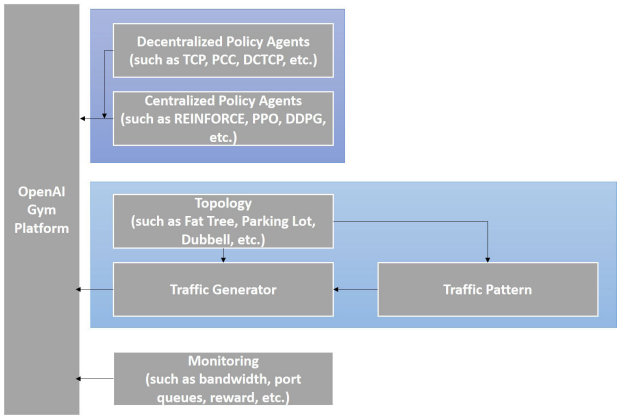

FIGURE 7. The architecture of the Iroko emulator to testify the reinforcement learning-based (such as REINFORCE, PPO, DDPG, etc.) and classical congestion control methods (such as TCP, PCC, DCTCP, etc.) under the same network traffic condition.

The emulator as shown in Figure 7 consists of a traffic generator based on specific network topology and traffic pattern, monitors to observe the network performance and RL reward, and an agent to enforce the congestion control policy. It supports centralized arbiters which operate as reinforcement learning (RL) policies for congestion control (such as REINFORCE, PPO, DDPG, etc.) as well as decentralized host-level congestion control approaches which are traditional TCP algorithms (such as DCTCP, TIMELY, or PCC). The mentioned arbiters aim to find the optimal fair bandwidth allocation while minimizing switch queues. The performed experiments show that DCTCP remains unbeaten as it is highly optimized with continuous kernel support, yet all tested RL-based algorithms (i.e., DDPG, PPO, and REINFORCE) show promising results as they were able to beat TCP New Vegas minimizing the queue buildup on the congested link.

Nougnanke *et al.* proposed lately a framework for incast performance prediction in data center networks [13]. This framework aims to infer incast completion time at run-time, where such information can be eventually used by any flow optimization algorithm or smart adaptive buffering method to adjust system parameters dynamically and hence achieve efficient performance for many-to-one communication traffic. The prediction model is constructed offline based on historical data. Each sample in this historical dataset represents a combination of features and a target value, where the learning is done in a supervised fashion. The features include:

- congestion algorithm used
- queuing discipline at the switch level
- number of competing senders
- bottleneck bandwidth
- round-trip-time
- server request unit
- minimum retransmission timeout

The target attribute is the incast completion time. The built model is then deployed as real-time incast performance

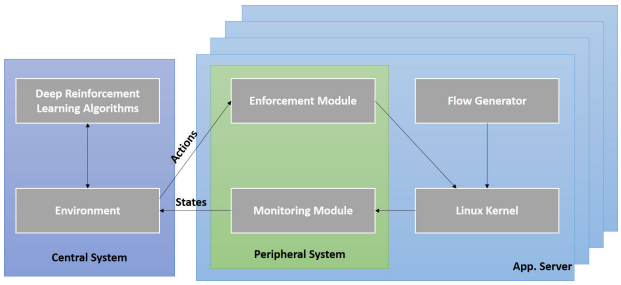

FIGURE 8. AuTO achieves data center-scale traffic optimization by running peripheral systems on the end-hosts locally for the minimum delay of short flows. Deep reinforcement learning algorithms (a deep deterministic policy gradient algorithm and a policy gradient algorithm) provides thresholds for MLFQ (multi-level feedback queuing) for short flow and rates, routes, etc. for long flows.

inference agent. The inferred incast traffic information is then used in the control plane for the smart buffering algorithm or traffic flow optimization algorithm. Several classic machine learning algorithms have been investigated as the inference agent, including Linear Regression, Support Vector Regressor, Decision tree, Random Forest, and Multilayer Perceptron, where only Random Forest gave good results.

Another interesting work by Chen *et al.* proposed two-level architecture, named AuTO, for automatic traffic optimization in DCN [14]. The proposed architecture consists of a peripheral system and a central system mimicked from animals' nervous system as shown in Figure 8.

The idea is to make short flow operations on the end-host (i.e., application server) and use deep reinforcement learning (DeepRL) methods to make traffic optimization decisions for long flows. The peripheral system makes local traffic optimization decisions in order to reduce the delay for short flows. Specifically, it applies multi-level feedback queuing (MLFQ) to schedule flows based on local information of bytes sent. Whereas the central system is composed of two deep reinforcement learning (DeepRL) agents: one for controlling MLFQ and the other for determining rates, routes, and priorities for long flows. The results show that the AuTO is able to achieve up to 48.14% performance improvement as compared with heuristic methods (i.e., shortest-job first and least-attained-service-first).

C. TRAFFIC CLASSIFICATION AND SCHEDULING

1) PROBLEM DEFINITION

Data center traffic has diverse communication patterns and is a mix of varying flow sizes that have different objectives as shown in Figure 9. More specifically, some flows (mice) are latency-sensitive such as web search and online gaming, and some flows (elephant) are more throughput-oriented like virtual machine migration and data backup.

To meet the requirement of these flows, network devices in modern data centers are designed with a shallow buffer in order to minimize the queuing delay of packets, which satisfies the low latency constraint of some applications. Nonetheless, such practice does not meet with the

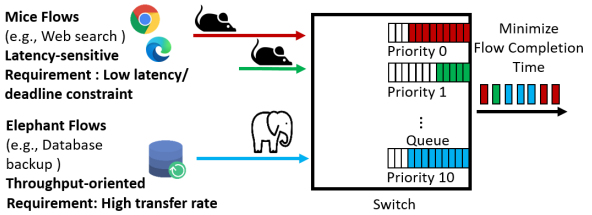

FIGURE 9. Traffic Classification and Scheduling: It is challenging to classify/identify and schedule flows having different requirements while minimizing the flow completion time (FCT).

applications requiring high bandwidth for which DCN needs a great capacity to handle bursty traffic and avert packet drop. Therefore, it is important to identify the latency-sensitive traffic from the throughput-sensitive traffic. One such method is based on the flow size classification, where mice flows are separated from the elephant flows. Furthermore, efficient data center transmission control schemes should be able to schedule flows in a way to minimize the flows completion time (FCT) by handling both, the low latency requirement of user-interactive flows and the high transfer rate needed by the throughput-hungry streams while ensuring full utilization of the network.

2) LITERATURE

Wang *et al.* presented a machine learning-based classification approach to detect elephant flows in packet-switched optical data center networks (PSON) for intra data center network [15]. Taking into consideration of not only accuracy but also computational performance, the proposition used supervised learning, i.e., C4.5 decision tree [16] and Naïve Bayes Discretization (NBD) [17] for flow classification. The chosen feature set includes:

- packet length (minimum, maximum, mean, and standard deviation)
- time between inter-arrival packets (minimum, maximum, mean, and standard deviation)

According to the experiments, C4.5 and NBD can achieve 95% and 90% recall with window sizes larger than 30 and 50, respectively. The resulting classification is then used to schedule flows based on a priority-aware algorithm.

Differently, an interesting work by Zhu *et al.* [18], bears that the flow information is not known as a prior and hence proposed a deep learning-based architecture, called SmartTrans, to classify traffic and predict the flow size rank. The proposed solution also includes multilevel priority queues that allow differentiated scheduling. On one hand, the flow classification output lets latency-insensitive flows give way to latency-sensitive ones, and on the other hand, the flow size forms an important aspect that affects the flow completion time, hence both factors are used in the scheduling technique. Considering the packets transmitted in sequence, recurrent neural network (RNN) was adopted to describe the sequential information, as shown in Figure 10. Specifically, each packet is divided by 4 bytes and each byte is embedded to a number between 0 and 255. As a special case of RNN

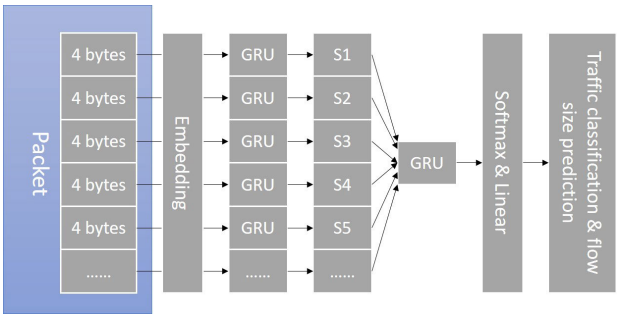

FIGURE 10. Design of the traffic classification and flow size trank prediction neural network (TCFSNN) treats IP packet communication as a natural language processing with 256 vocabularies (by embedding each byte to be a number in 0-255). GRUs (gated recurrent unit) are used to keep long short-term memory when processing the packets.

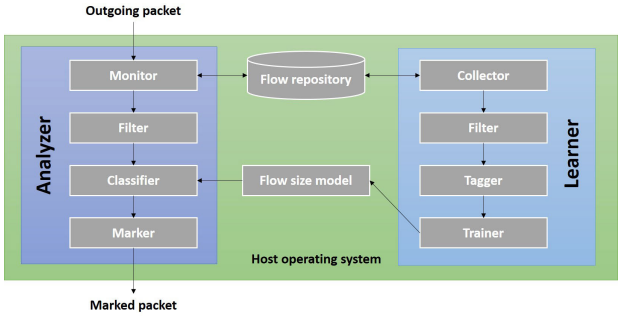

FIGURE 11. NELLY uses incremental learning to detect elephant flows at the server in DCN. The analyzer applies a classifier to identify and mark the elephant flows. The learner uses an incremental learning method to update the flow size model.

unit, a gated recurrent unit (GRU) was used for its better performance in mitigation of gradient vanishing. The output of each GRU is concatenated to feed to another GRU followed by softmax and linear fully connected network to finally output traffic classification flow and flow size, respectively. When a flow is recognized, the first three packets of it are fed into the aforementioned network. Promising results have been shown with good performance on traffic classification (F1-score: >99%) under different workloads and load levels in various scenarios (browsing, email, FTP, P2P, Youtube, Spotify).

Estrada-Solano *et al.* [19] proposed NELLY, which leverages incremental learning from software-defined networking (SDN) to identify elephant flows of great magnitude in the network accurately in a reasonable time while generating low control overhead. NELLY aims to address the inaccuracy, high overhead and poor scalability of flow detection in software-defined data center networks. The proposition operates as a software component deployed in each and every server in the data center network. The proposed architecture consists of two subsystems namely an analyzer and a learner, as shown in Figure 11. For the analyzer, the following information from the outgoing packet is monitored and filtered: source IP, source port, destination IP, destination port, IP protocol, size, and time. A flow ID together with the above info is stored in the flow repository. The classifier leverages the flow size classification model for on-the-fly detection and marking of elephant flows and mice flows. For the learner, the collector pulls records from the flow repository. The tagger simply compares the actual size of the flows with a threshold so that they can be tagged as either mice or elephant flow. The trainer applies incremental learning algorithms in order to build and update the flow size classification model. Different incremental learning algorithms were considered and compared in this work for classifying flows. According to true positive ratio and false positive ratio in the experiments, the top-5 best models are AHOT (adaptive Hoeffding option tree), ARF (adaptive random forest), Hoeffding tree, OAUE (online accuracy updated ensemble), OzaBag (Oza and Ressel's Bagging).

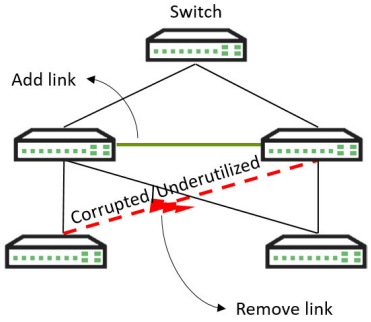

FIGURE 12. Topology Management: A key challenge in topology management is dynamically adding new links and/or removing corrupted and underutilized links.

D. TOPOLOGY MANAGEMENT

1) PROBLEM DEFINITION

Different topology models for data center networks have been designed to provide rich connectivity among the network devices and satisfy all applications requirements. The networking community has come up with various techniques for improving and managing their performance. Such techniques include topology improvements through flexible links or cutting out corrupted or underutilized ones as shown in Figure 12. These techniques are formalized as optimization models solved with greedy heuristics to create approximation solutions. This is because finding the optimal location to add augmenting links, or the set of links to be removed under various and rapidly changing circumstances is complex and even of non-deterministic polynomial-time hardness. Nevertheless these heuristics are mostly domain-specific and need to be redesigned for any minor change in the application pattern or network details and even for hardware dissimilarities, which raise the need for automated intelligent techniques for topology management in DCN.

2) LITERATURE

A work by Salman *et al.* proposed a machine learning-based architecture for topology management [20]. The proposition replaces domain specific rule-based heuristics with DeepRL agent. The DeepRL agent interacts with the environment

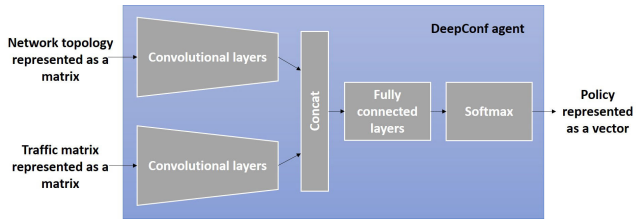

FIGURE 13. DeepConf RL agent utilizes a convolutional neural network model to generate/deploy RL policy. Convolutional layers extract high-level features from network topology and traffic matrix, followed by a Concat layer to combine the features together for fully connected and Softmax layers.

(network) according to its state (topology) with actions (select links to activate), receive rewards (based on link utilization and flow duration), and update its policy (i.e., state-action mapping). More specifically,

- *state space*: the network topology (represented as a sparse matrix where entries correspond to the activated links).
- *action space*: different possible link combinations (represented as a vector where elements indicate the possiblity of the corresponding link to be picked up).
- *reward*: maximize link utilization and minimize the average FCT (flow completion time), i.e.,

$$R = \sum_{f \in F} \sum_{l \in f} \frac{b_f}{d_f} \tag{1}$$

where F denotes all completed flows and l denotes the used links by f. b_f is the total number of the transmitted bytes of flow f. d_f is the total duration of flow f.

A network simulator is used to train offline the agents which encapsulate the data center functionality. The learning model utilizes a convolutional neural network (CNN) having the network topology and traffic matrix as input state and the policy that dictates the links that should be activated in the network topology as output, as shown in Figure 13. The convolutional layers extract the spatial features in the network topology matrix and traffic matrix, followed by concatenated together for fully connected layers and softmax to compute the final probability vector (policy vector). Such an ML-based solution demonstrated its effectiveness with the ability to learn in a reasonable number of episodes and its capability to learn a solution close to the optimal one across different data center topologies.

E. NETWORK STATE PREDICTION

1) PROBLEM DEFINITION

Performing measurements and monitoring is an essential aspect in order to understand the performance of the network and debug many issues that can come across distributed applications in the data center. The main problem as illustrated in Figure 14 is how to accurately spot and measure events of interest, at the scale of a data center network consisting of thousands of nodes connected with high link rates and

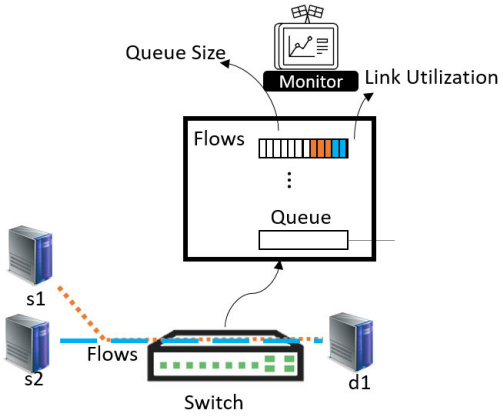

FIGURE 14. Network State: Predicting network state like queue size, delay and link utilization from the flows between senders (s1, s2) and destination (d1, d2) needs to be accurate, at data center scale, and in near real-time.

hence variable phenomena frequency, and in a near real-time manner.

2) LITERATURE

Geng *et al.* proposed SIMON for accurate and scalable network telemetry in data centers [21]. It reconstructs network states such as link utilization, packet queuing delays in network switches, flow-level queue and link compositions. The system uses a machine learning inference algorithm (Lasso, Least Absolute Shrinkage and Selection Operator Regression) in order to obtain the related variables. As Lasso tries to minimize the cost function in the form of

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i| \tag{2}$$

where θ is the parameter to be inferred; MSE stands for mean squared error; α is a hyperparameter, it requires BGD (bunch gradient descent) to iteratively to approach the optimum solution, which leads to a scalability limitation for large systems. Furthermore, neural networks are also designed to accelerate the afore-mentioned reconstruction. In detail, the neural network is a three-layer ReLU (rectified linear unit [22]) neural network with only one hidden layer. For example, if we denote D as probe delays and Q as queue sizes, the estimate of queue sizes $\widehat{Q}$ is

$$\widehat{Q} = M \times ReLU(LD + b) + l \tag{3}$$

where (L, b) and (M, l) are the network parameters in the hidden layer and output layer respectively (as shown in Figure 15). The proposed neural network-based acceleration approach is shown to accelerate measurement by a factor of 5000x to 10000x, allowing it to run in near real-time.

F. ROOT CAUSE ANALYSIS

1) PROBLEM DEFINITION

Troubleshooting network failures (e.g., Latency, packet reordering, Sporadic packet drops and Bandwidth throttling)

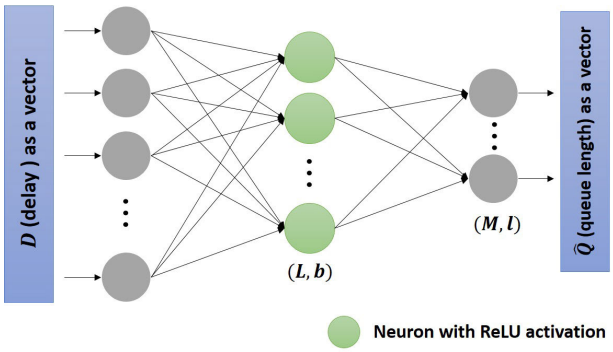

FIGURE 15. A neural network for network measurement reconstruction (e.g., estimate delays by feeding the queue lengths). (L, b) and (M, l) are the weights and biases of the neurons. More specifically, the neurons in the middle layer use ReLU (rectified linear unit) as their activation function.

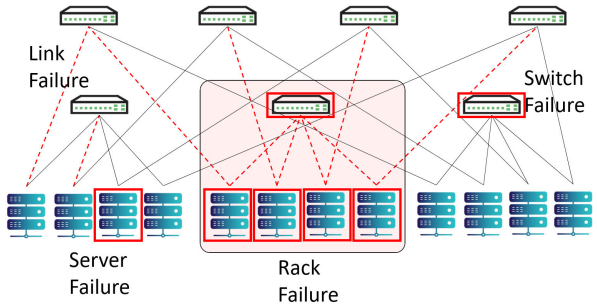

FIGURE 16. Root Cause of Network Failures: Various failures (highlighted in red) can occur in DCN at different layers like links, switches, servers and racks and the root cause of these failures is to be identified. A failure can be as simple as a link or as worse as the entire rack. It is challenging to automatically find the root cause and location of these failures.

to identify their root cause, depicted in Figure 16, has become a more critical mission with the increase of data rates. It involves real-time monitoring of wide-ranging metrics and data sources including packets. However, to find the root cause and accelerate the resolution of a problem, assessing the network metrics is not enough but rather a clear interpretation of the correlation between the user experience, the behavior of the network and the underlying problems in the network, is needed. An ideal situation would be the ability to rapidly pinpoint the most potential source failure, yet the whole process is time-consuming and expensive and spans over different IT teams and subsystems. Furthermore, the same symptoms of a particular problem might be observed yet the actual cause might differ which makes the troubleshooting process even more complex.

2) LITERATURE

Arzani *et al.* proposed NetPoirot, which train random forest models on historical network data to identify anomalies and root causes of failures, correspondingly [23]. It is a single node solution where end-hosts independently run pre-trained classification models on local TCP statistics aiming to localize the root cause of a failure to a remote server, a local client, or network after which more adequate management

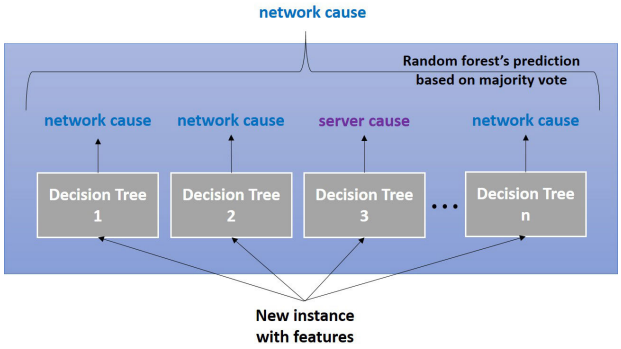

FIGURE 17. NetPoirot uses random forest (composed of several decision trees) as an ensemble learning method to decide the root cause by majority vote. When a new instance with features comes, only Decision Tree 3 recognizes the root cause as server cause while the rest recognizes it as network cause. Based on the Law of Large Numbers, the output from the random forest is network cause.

techniques can be used. Specifically, in the training phase, a fault injector is triggered to inject various kinds of failures to either the local machine (end-host) connecting with remote servers which results in different failures including server issues (such as high I/O on server, slow reading, etc.), client issues (such as high CPU load, high memory load, etc.), and network issues (such as packet loss, random connection drops, high latency, etc.). Once the features (either in the form of raw value or statistic metric) and their corresponding labels (server, client, or network) are prepared, the subsets of the training data are fed into decision trees to compute the splitting thresholds in features which leads to decision-making (cause of failures) in the leaf node. In the deployment phase, an ensemble method based on decision trees, called random forest, is applied which makes root cause decision by considering the voting score from multiple decision trees generated in the training phase. The Law of Large Number guarantees the higher accuracy and recall of random forest compared with a single decision tree. As an example shown in Figure 17, most of the decision trees output network cause which leads to the final decision of the random forest as network cause according to the majority vote criterion.

G. NETWORK SECURITY

1) PROBLEM DEFINITION

Data centers host a vast amount of user data and therefore the network security is a key requirement for any DCN to protect the usability and integrity of both the network and user data. DCNs often face targeted attacks from malicious entities and traditional firewalls are not capable of handling emerging threats. Moreover, encrypted traffic makes deep packet inspection even harder for the network operators to identify such attacks automatically. Therefore, DCNs require sophisticated intrusion detections systems (such as shown in Figure 18) that can detect malicious entities, generate signatures for the malicious traffic and install rules (based on inferred signatures) using SDN controllers to prevent these attacks in real-time.

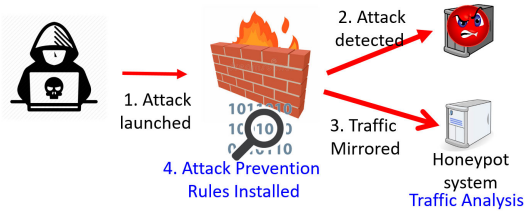

FIGURE 18. Security: An IDS detects malicious entities, generates signatures for the malicious traffic, and install rules (based on inferred signatures) using SDN controllers to prevent these attacks in real-time.

2) LITERATURE

ProHacker uses a non-parametric semi-supervised machine learning approach to infer protocol formats [24]. It works in two stages as shown in Figure 19. During the offline stage, it collects the network traffic and then analyses the collected traffic to learn keywords. To learn keywords, it splits input data into n-grams and applies a heavy hitter algorithm to identify important (recurring) keywords. It then builds a protocol language model using a hierarchical Pitman-Yor process on the learned keywords. The output of this function is a language model corresponding to each protocol type found in the network trace. Next, it applies ensemble learning based approach to train several weak classifiers using a small amount of labeled data and protocol keywords. During the online stage, it extracts the keywords from the real traffic and uses the pre-trained classifiers (from the offline stage) to detect malicious traffic. Even though the proposed scheme is lightweight and fast, it requires a large amount of traffic to extract the keywords accurately.

SANTaClass uses an unsupervised machine learning technique to generate protocol signatures [25]. It takes network traffic as an input and generates application protocol signatures at the output (See Figure 19). First, it uses a common substring matching algorithm to first identify protocol keywords, and then use these keywords to separate known flows from unknown flows using a trie-like data structure. Next, it generates protocol signatures for the unknown traffic by first identifying common terms in the protocol set, and refining them by removing short and unrelated terms. Lastly, it uses the generated signatures to identify traffic in real-time. The proposed mechanism works well for encrypted traffic and achieves an accuracy of up to 80%.

Deep packet is a deep learning based network traffic classification framework that can handle traffic belonging to various applications. Deep packet can also identify encrypted traffic and can distinguish between VPN and non-VPN networks. The core of this framework is two deep neural network structures i.e., stacked autoencoder (SAE) and convolutional network (CNN). The key difference from the above methods is that it does not require feature (or keyword extraction). Instead, it pre-processes packets based on the link layer and TCP headers and truncates irrelevant data from the packet. Next, it feeds the data to SAE and CNN which classify the input traffic to one of the categories.

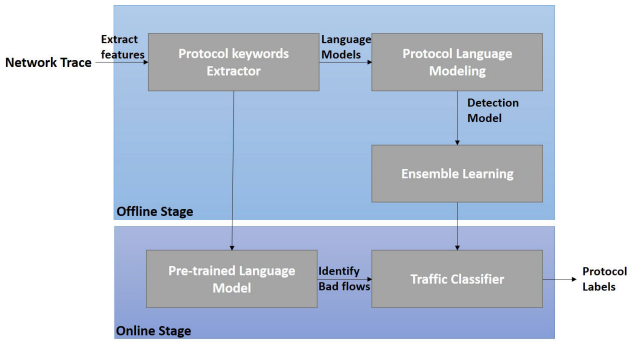

FIGURE 19. Prohacker Signature Generation uses a common substring matching algorithm to first identify protocol keywords and then applies the keywords to generate protocol signatures.

H. DISCUSSION

Table 1 summarizes the ML techniques and their objectives in solving DCN challenges. Several interesting observations can be concluded from this Table. First, it is hard to create ground truth data for the ML techniques. Many of these works employ classical supervised learning techniques, which require data labeling, to solve particular DCN problems. This is challenging in a dynamic and continuously evolving environment like DCNs. Second, it is hard to collect the training dataset for supervised learning. Many of these works shed the light on the non-availability or accessibility to real DCN datasets needed for the machine learning analysis. Third, these solutions require a high level of accuracy for the monitoring tools to get the right insights of the features at the adequate granularity necessary for the ML techniques. Lastly, the time to collect the telemetry data, process the data and make decisions should be very little (on the order of microseconds). Despite these challenges, many of these works present interesting results that show the potential of machine learning in making DCN better and more efficient. In contrast to these works, Fu *et al.*, [26] make an interesting observation that ML cannot provide an accurate prediction of resources without losing any of the simplicity or generality aspects of existing networking solutions.

IV. CHALLENGES FOR ADOPTING ML IN DCN

1) DECISION MAKING LATENCY

While DCN traffic is a mix of mice and elephant flows, short flows constitute the majority of the traffic and hence they can be easily gone before any control or optimization decision is made rendering most decisions useless. Hence the challenge of achieving all of fine-grained traffic monitoring, data preparation, models training and inference rapidly and the challenge of choosing the appropriate machine learning architecture that can meet such requirements, where decisions ideally should be taken on orders of microseconds.

2) DATA AVAILABILITY

Training data is the backbone of supervised machine learning without which it will be impossible for the model to learn

TABLE 1. Taxonomy of ML-based solutions for DCN.

Work	Features	ML Model	ML Objective	Data Sets	Key Results
[10]	Hundred and six features including: • Standard deviation • Kurtosis • Skewness • Absolute some of the changes • Auto-correlation at different lags • Partial auto-correlation at different lags • The first location of minimum • Linear least-squares regression	• Support Vector Regression • Kriging • Linear Regression • Gradient Boosting Tree • K-Nearest Neighbors • Naive Bayes • Multi-layer Perceptron • Random Forest	• Resource predictor selection • Resource utilization prediction	AliBaba [27], BitBrains [28] and Google [29] data sets	• High accuracy of resource utilization prediction for different type of workloads • Window size has significant effect on estimation accuracy • Hard to forecast in the presence of burst
[30]	Time and Frequency features	Neural Network	Prediction of incoming/outgoing traffic volume on inter-DC link	Six weeks of inter-DC network traffic data from a production data center with tens of thousands of servers from Baidu	Between 5% - 10% reduction in prediction error
[12]	• switch buffer occupancy • interface utilization • active flows	• Deep Deterministic Policy Gradient • Proximal Policy Optimization • REINFORCE	Congestion Control	Simulated data set	• DCTCP remains unbeaten • All algorithms (DDPG, PPO, REINFORCE) beat TCP New Vegas minimizing the queue buildup on the congested link
[13]	• Congestion algorithm used • Queuing discipline at the switch level • Number of competing senders • Bottleneck bandwidth • Round-trip-time • Server request unit • Minimum re-transmission timeout • Target attribute is the incast completion time	• Linear Regression • Support Vector Regression • Decision Tree • Random Forest • Multi-Layer Perceptron	Incast performance prediction	Simulated data set	• High prediction performance score • Machine learning is generalizable for diverse congestion control and queuing discipline schemes
[14]	• Flow sizes and Completion times of all finished flows • On-going long flows	Deep Reinforcement Learning	• MLFQ thresholds optimization • Determination of rates, routes, and priorities for long flows	Web search workload [31] and data mining workload [32]	Significant reduction in average FCT
[15]	• Packet length (minimum, mean, maximum, and standard deviation) • Inter-arrival time between packets (minimum, mean, maximum, and standard deviation)	• Decision Tree C4.5 • Naive Bayes Discretization	Traffic-Flow Classification	Simulated data set	• C4.5 and NBD are not able to detect elephant flows accurately with small window size. • Window size affect accuracy. • Buffer size affect performance in terms of packet loss.
[18]	• Flow Type • Flow Size (rank/interval)	Deep Learning	Flow Information (Type and size rank) Prediction	Four datasets including UNB CIC VPN-nonVPN dataset [33], and three datasets from two universities and an academic dataset at a college [34]	• Good performance on predicting the elephant flow. • Less average FCT of latency-sensitive flows under different workloads and load levels. • High average throughput of latency-insensitive flows.
[19]	• Packet header • Size • Inter-arrival time of first N packets (13 features)	• Adaptive Hoeffding option tree • Adaptive Random Forest • Hoeffding Tree • K-nearest Neighbors with probabilistic adaptive windowing • Naive Bayes • Online accuracy updated Ensemble	Elephant flow detection	Two real packet traces captured in university data centers	• Accurate and supports low classification time with adaptive decision trees algorithms • Low traffic overhead
[20]	• Network Topology • Traffic Matrix	Deep Reinforcement Learning	Topology management	Trace-driven flow-level simulator using large scale map-reduce traces from Facebook [35]	• Able to learn solution close to the optimal one • Outperforms greedy heuristics
[21]	• 5-tuple header information (source and destination addresses, source and destination port numbers, protocol port number) • Transmit and receive time stamps at the corresponding NICs • Length of the packets	Neural Network	Accelerating reconstruction of network state variables	Three Testbeds of 20, 500 and 8 racks with 12, 24 and 16 servers in each respectively	Acceleration by 5000x–10000x enabling near real-time reconstruction
[23]	• Number of bytes received by TCP • Duration in which connection has been open • Number of bytes delivered to the application • Maximum smooth RTT estimate observed • Maximum congestion window • Number of triple duplicate ACKs • Ratio of the number of bytes posted by the application to the number of bytes sent	Random Forest	Root cause analysis of failures	Six months of traffic in Azure production cloud	Around 96% identification accuracy for some failure types

and predict. However, one of the important challenges for ML in DCN is the unavailability of real or realistic data center networking data sets for the academic research and industrial communities since the topology and the traffic patterns are often considered as private and proprietary and the industry cannot expose their customers' data for privacy preservation. Another important aspect is the granularity of data that can be collected at different levels whether on the hosts, controllers, switches, or more fine-grained per-flow information. While data differ from one level to another, there is a need for a knowledge-sharing mechanism in order to build ML models that can efficiently learn and adapt to different scenarios.

3) COMPUTATIONAL RESOURCES

Adopting data plane or control plane to run machine learning are both effective solutions, yet each raises its own challenges. Deploying switch-based machine learning is limited to the hardware capability whereas the server-based approach questions the significant latency entailed and bandwidth needed to communicate the data needed for the machine learning analysis. Another challenge that we shed the light on is whether to adopt online or offline analysis and the frequency of the latter which affect all of the computational needs, freshness and efficiency of the machine learning models.

4) PRIVACY

Data center networking tools facilitate communication and information exchange between their connected components and both internal and external networks. While data centers networks usually offer built-in security guarantees, privacy remains a critical aspect especially in environments in which raw data needed for training machine learning models are inconvenient to share neither between network elements in the same domain nor in intra-DC communication. Hence the challenge of the need for the data to build efficient ML models and the risks that will be put on the line by allowing such data to leave its host.

V. VISION AND OPEN RESEARCH DIRECTIONS

Regardless of the hurdles machine learning has to overcome, ML is likely to make its way in data center networking and become one of its core pillars. Hereafter, we present our vision while we discuss some open directions where and how ML can be a potential game-changer in the direction of future DCN automation, autonomy and intelligence.

A. VISION

The need for decentralized and scalable solutions is even more significant. The fact that traffic has to be evaluated in real-time and monitored data has to be communicated with a central entity for analysis and decision making, leads to a slow learning curve, network overload, and latency. Additionally, optimizing a network of tens of hosts is already a substantial task, since each node is an independent actor with unpredictable behavior, which impedes scalability. Therefore, we envision that any ML solution for DCNs should consider divergence in its architecture and the aspects that are usually omitted or left as after-thoughts in the proposed solutions. We discuss all these aspects in the sequel.

B. ML APPLICATION SCENARIOS

Many ML algorithms have been proposed to work in DCNs. However, many of these ML solutions need to be customized and enhanced to cater to the constraints posed by the DCN consumers, users, network operators, cloud service providers, etc. For instance, ML can help with operations, but it might also be applied under the hood without network operators being aware of it. For traffic control, existing works have proved the sensitivity and effect of the training window size on the accuracy of the machine learning model, yet none has proposed automatic and adaptive techniques to set the window size effectively. Moving window size for the training set is one of the potential tracks to investigate in these areas for its significant effect on the ML model estimation error. Reinforcement learning (RL) has shown its potential with very promising results for traffic control. In spite of the traffic volatility in DCN, an agent is able to learn a reasonable set of parameters to define traffic control policy. A major challenge for RL algorithms is that the decisions have to be made on a scale of milliseconds or even microseconds, the tolerance for error in data center network is small and, compared to a traditional TCP algorithm, the RL agent needs to cope with a significant delay in its actions on a local host. Reducing the reaction granularity and allowing more complex actions such as providing a series of actions for the next few seconds are some of the promising research directions in this space.

ML for traffic optimization is currently the most investigated area, however, typically these methods rely on classifying a flow as mice or elephant. Currently, there is no reliable and accepted method for defining the threshold value that differentiates between mice and elephant flows in DCNs. Proposing an adaptive automatic method to select the appropriate threshold value for the traffic and routing requirements is still an open research direction to explore.

On the other hand, the ML works for topology management, state reconstruction and root cause analysis in DCN have shown very encouraging results as well, yet very limited works have been advanced in these areas which open the door for countless further enhancements.

Collecting network traces in a privacy-preserving manner is a critical task. While few data sets are available [27]–[29], there are no adequate data sets that are publicly available for the evaluation of ML solutions for many other areas in DCN explored in this work, which pushed researchers to create their own data sets or use simulated data making them unreliable for real evaluation as shown in Table 1. This creates an interesting track for the industry to consider, yet definitely not to forget about their customers' privacy.

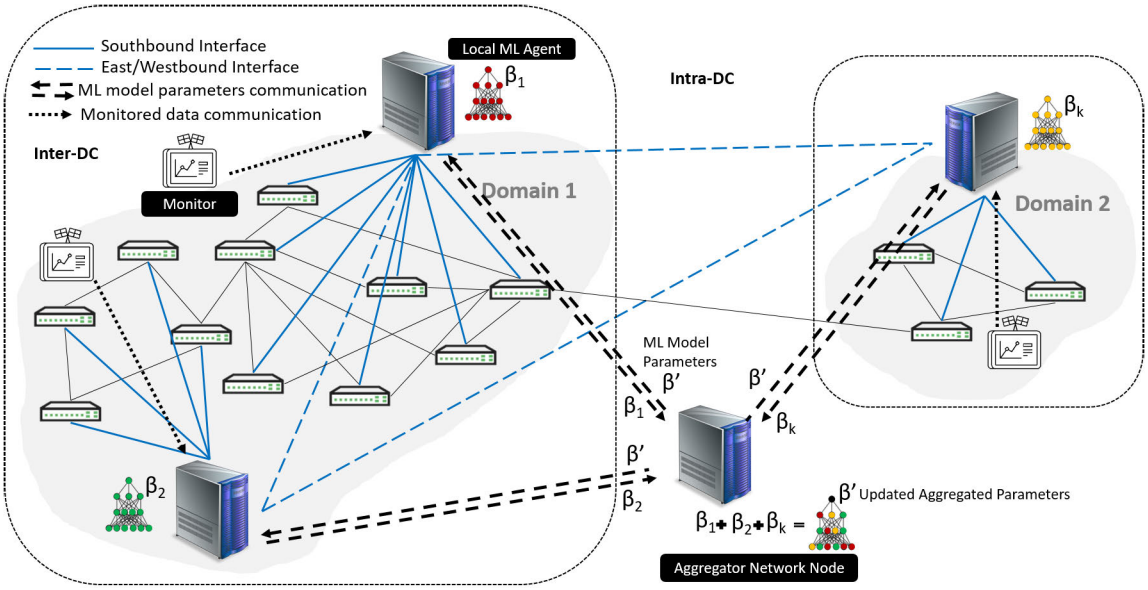

FIGURE 20. Federated Learning (FL)-Enabled DCN: our vision of re-architectured machine learning deployment for DCN.

C. ML ARCHITECTURE FOR DCN

1) WHAT, WHEN AND HOW OF ML TECHNIQUES

The chaotic and opaque nature of data center networks makes it unpredictable, hence there is a need to investigate applications of reinforcement learning more thoroughly rather than relying on supervised learning techniques that require some manual intervention to build good solutions. Applying ML analysis requires continuous monitoring to capture observations, and data transfer over the network to train ML models, which if not efficiently handled, can cause latency in the network and bottleneck on the communication links. Therefore, we envision that any ML architecture should consider when, whether online or offline, and how frequently to perform training and inference tasks to guarantee model freshness without causing additional overhead on the network. None of the existing works consider or clearly justifies these aspects despite their criticality.

2) ML DEPLOYMENT MODEL

While the debate is always whether to apply centralized or decentralized ML analysis, we envision more advanced paradigms to apply machine learning in DCN. We believe that ML analysis should be done close to the generated data in order to avoid latency and overhead and save bandwidth. This is very important for DCNs, where the decisions (such as scheduling or forwarding) have to be done ideally in the order of microseconds. While centralized architecture cannot meet these requirements, decentralized deployment can. However, distributed architectures lack a global view of the network and impose the need for data coordination. Direct distributed coordination between agents can in turn impose significant overhead on links and nodes in the network. From our perspective, Federated Learning (FL) [36] is the most suitable paradigm for the learning process in DCN which is capable of addressing these challenges and meeting these requirements. We give in Figure 20 a generic overview of the proposed re-architected ML deployment which depicts our vision of future generation networking in data centers as FL-enabled DCN. In this architecture, metrics tailored to the specific problem at hand, are monitored through the monitors in the network and the gathered data is then communicated with the local ML engine for model training. Model parameters (Betas in Figure 20) from local agents are then communicated with the central entity for aggregation rather than raw data, which minimizes the overhead of data communication on the links and reduces the associated bandwidth usage overheads. This also offers a significant reduction in latency over the network. Further, it provides a certain level of privacy whereby data is never communicated with external entities in intra-DC communication. The new enhanced model is then exchanged by the central entity with the local agents for knowledge sharing and a global network-wide view of the network. Local agents can be used for real-time fast analysis needed for instance to meet mice flows requirements whereas the central analysis can be dedicated for decisions that tolerate higher latency and require a global view of the network. Furthermore, such architecture meets well with the scalability need in DCN. Additional improvements on the model can be further applied by each of the local agents until a certain level of convergence is achieved. However, identifying the convergence level tailored for each DCN problem is a challenging task.

3) NETWORK TELEMETRY

Building an efficient network telemetry solution is important for DCNs. Monitoring is a critical aspect for designing and

deploying appropriate and effective management solutions for data center network. Currently, passive probes continuously monitor and trace desired performance metrics (such as the status of the queue, packets and byte counters and statistics on latency) in the network. While such passive monitoring is effective and straightforward, the information provided is local and does not cover the paths in the network or dropped packets, which does not provide network-wide vision. On the other hand, some protocols also use messages in the control plane to identify bandwidth, delays and packet loss.

Network telemetry has become even more essential for network with the continuous increase in DCN scale and with the evolution of network technologies like 5G. Furthermore, the constant flow of critical data through the data center network has raised the need for continuous monitoring and tracing to detect current or potential failures, anomalies, and congestion and to respond to them in a timely manner. Such criticality raises the value of investing in network telemetry. Particularly, measurements methods and metrics granularity, which play important role in the ML analysis as they ensure the availability of the needed data for model training and inference, are critical aspects to examine and important promising directions. In its turn, machine learning has also proven to be an efficient means to accelerate the reconstruction of the network state like link utilization, packet queuing delays in network switches, flow-level queue and link compositions, yet still not extensively studied, which makes it a promising open track to explore.

Another dimension is to build lightweight distributed telemetry blocks that can become part of future architectures by default and assist in timely information collection (over long times) and processing (quick) across the network. Such a system can act as an end-to-end closed-loop system that collects and processes telemetry data to take certain actions. Different use cases require solutions at different timescales. For example, traffic engineering needs to make decisions at the minutes level compared to the link level flow control that requires decision making at microsecond to millisecond timescale. This requires building lightweight blocks for ML-based telemetry that collects information over a long time but is able to take actions quickly (at different timescales) to adapt to any changes or events in the network. Monitoring generates a huge amount of data and collects it at a central node. Processing this data and making decisions in real-time is a tedious task.

4) SECURITY AND PRIVACY

Recent studies demonstrate that machine learning models are themselves an easy target for security and privacy threats [37]. This includes poisoning of training data, which can lead to either diminished accuracy or error attack. Another type of attack is a well-designed backdoor in the training set which is able to trigger critical consequences in the ML agent. Also, an adversarial example or a well-crafted disorder in the test input can lead to the wrong model. Further, sensitive training data can be recovered through stealing the model, an inference or an inversion attack. Such threats can lead to critical consequences in the ML engines and eventually in the network especially in security and privacy critical applications. Considering users and customers as separate entities, DCN typically belongs to a single authority and therefore, privacy might not be a critical aspect in inter-DC communication. However, data shared in intra-DC communication to train ML models are still subject not only to security but also privacy attacks. Protecting both the parameters of the model from the service provider's perspective, and the data privacy from the user's perspective has gained increasing attention in recent years. Differential privacy techniques have been proposed where noise is injected at various levels, yet this can still hurt the model and lead to a higher error rate. Additionally, fair accuracy can only be maintained with a small number of ML agents involved, which questions the scalability of such techniques in the scale of DCN. On the other hand, cryptographic methods have been also advanced and considered as loss-less, yet still need improvements as they introduce high overhead to the training and can lead to performance degradation of the ML model and some might not be able to defend against poisoning attacks. Designing robust security and privacy-preserving methods for machine learning with the least loss of accuracy possible is a promising direction.

An interesting research direction is establishing a mechanism that can evaluate the security and privacy of machine learning systems. So far, very little work has been done to evaluate the robustness of the ML systems against security and privacy attacks. Also, there is no unified method and metrics to evaluate the performance of existing defenses.

VI. CONCLUSION

Undeniably, future DCN will have to support the explosive growth in the volume of traffic entailed by smart connected devices and multi-tenant applications and services with remarkable capabilities that meet their needs. Yet the exceptional scale and volatility will undoubtedly increase the complexity of network operations and management, urge the need for automation, intelligence and autonomy. Over the last past decades, ML has proved its capability in different domains including networking. In this article, we provide a comprehensive branch of perception on the applicability of ML solutions to support data center network operations and management including workload forecasting, traffic control and optimization, topology management, network state prediction, failures analysis, and security.

We review the existing literature works that proposed ML approaches for addressing these DCN problems and explore their proposed techniques while studying their requirements and analyzing their feasibility and performance. Although ML has shown promising results in addressing these

problems, yet many challenges are still there impeding the efficient application of ML-based solutions. We also examine and discuss these challenges in this article along with some opportunities while we draw our vision pertaining to the intelligent and autonomic operation and management of future data center networks.

We believe that our findings offer fundamental insights into the future research progress and field advancement and motivate the need for more research with advanced and re-architected solutions to the realized future vision of intelligent automatic DCN.

ACKNOWLEDGMENT

(Haiwei Dong and Ali Munir contributed equally to this work.)

REFERENCES

[1] Y. Zhou, H. Dong, and A. El Saddik, "Deep learning in next-frame prediction: A benchmark review," *IEEE Access*, vol. 8, pp. 69273–69283, 2020.

[2] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1686–1721, 3rd Quart., 2020.

[3] J. Wang, B. Hafidh, H. Dong, and A. E. Saddik, "Sitting posture recognition using a spiking neural network," *IEEE Sensors J.*, vol. 21, no. 2, pp. 1779–1786, Jan. 2021.

[4] Cloudfabric 2.0. *Leading Data Center Networks into the Intelligence Era*. Accessed: May 10, 2021. [Online]. Available: https://e.huawei.com/en/videolist/networking/98a893118b72467fbdc41bd18f5b0c6a

[5] Y. Liu, J. K. Muppala, M. Veeraraghavan, D. Lin, and M. Hamdi, *Data Center Networks: Topologies, Architectures Fault-Tolerance Characteristics*. Cham, Switzerland: Springer, 2013.

[6] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, 2009.

[7] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[8] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, 2009.

[9] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, 2009.

[10] S.-U.-R. Baig, W. Iqbal, J. L. Berral, A. Erradi, and D. Carrera, "Adaptive prediction models for data center resources utilization estimation," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1681–1693, Dec. 2019.

[11] Y. Li, H. Liu, W. Yang, D. Hu, X. Wang, and W. Xu, "Predicting inter-data-center network traffic using elephant flow and sublink information," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 782–792, Dec. 2016.

[12] F. Ruffy, M. Przystupa, and I. Beschastnikh, "Iroko: A framework to prototype reinforcement learning for data center traffic control," 2018, *arXiv:1812.09975*. [Online]. Available: http://arxiv.org/abs/1812.09975

[13] K. B. Nougnanke, Y. Labit, M. Bruyere, S. Ferlin, and U. Aivodji, "Learning-based incast performance inference in software-defined data centers," in *Proc. 24th Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Mar. 2021, pp. 118–125.

[14] L. Chen, J. Lingys, K. Chen, and F. Liu, "AuTO: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2018, pp. 191–205.

[15] L. Wang, X. Wang, M. Tornatore, K. J. Kim, S. M. Kim, D.-U. Kim, K.-E. Han, and B. Mukherjee, "Scheduling with machine-learning-based flow detection for packet-switched optical data center networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 10, no. 4, pp. 365–375, Apr. 2018.

[16] R. Kohavi and J. R. Quinlan, "Data mining tasks and methods: Classification: Decision-tree discovery," in *Handbook of Data Mining and Knowledge Discovery*. London, U.K.: Oxford Univ. Press, 2002, pp. 267–276.

[17] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. 11th Conf. Uncertainty Artif. Intell.*, 1995, pp. 338–345.

[18] K. Zhu, G. Shen, Y. Jiang, J. Lv, Q. Li, and M. Xu, "Differentiated transmission based on traffic classification with deep learning in datacenter," in *Proc. IFIP Netw. Conf.*, Jun. 2020, pp. 599–603.

[19] F. Estrada-Solano, O. M. Caicedo, and N. L. S. Da Fonseca, "NELLY: Flow detection using incremental learning at the server side of SDN-based data centers," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1362–1372, Feb. 2020.

[20] S. Salman, C. Streiffer, H. Chen, T. Benson, and A. Kadav, "DeepConf: Automating data center network topologies management with machine learning," in *Proc. Workshop Netw. Meets AI ML (NetAI)*, 2018, pp. 8–14.

[21] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosenblum, and A. Vahdat, "SIMON: A simple and scalable method for sensing, inference and measurement in data center networks," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement.*, 2019, pp. 549–564.

[22] V. Nair and G. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[23] B. Arzani, S. Ciraci, B. T. Loo, A. Schuster, and G. Outhred, "Taking the blame game out of data centers operations with NetPoirot," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 440–453.

[24] Y. Wang, X. Yun, Y. Zhang, L. Chen, and T. Zang, "Rethinking robust and accurate application protocol identification," *Comput. Netw.*, vol. 129, pp. 64–78, Dec. 2017.

[25] A. Tongaonkar, R. Keralapura, and A. Nucci, "SANTaClass: A self adaptive network traffic classification system," in *Proc. IFIP Netw. Conf.*, 2013, pp. 1–9.

[26] S. Fu, S. Gupta, R. Mittal, and S. Ratnasamy, "On the use of ML for blackbox system performance prediction," in *Proc. 18th USENIX Symp. Netw. Syst. Design Implement.*, 2021, pp. 763–784.

[27] *Alibaba Cluster Log*. Accessed: May 10, 2021. [Online]. Available: https://github.com/alibaba/clusterdata

[28] *Bitbrains Cluster Log*. Accessed: May 10, 2021. [Online]. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains

[29] *Google Cluster Log*. Accessed: May 10, 2021. [Online]. Available: https://github.com/google/cluster-data

[30] Y. Li, H. Liu, W. Yang, D. Hu, and W. Xu, "Inter-data-center network traffic prediction with elephant flows," in *Proc. NOMS IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2016, pp. 206–213.

[31] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-agnostic flow scheduling for commodity data centers," in *Proc. 12th USENIX Symp. Netw. Syst. Design Implement.*, 2015, pp. 455–468.

[32] W. Bai, L. Chen, K. Chen, and H. Wu, "Enabling ECN in multi-service multi-queue data centers," in *Proc. 13th USENIX Conf. Netw. Syst. Design Implement.*, 2016, pp. 537–549.

[33] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 407–414.

[34] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th Annu. Conf. Internet Meas. (IMC)*, 2010, pp. 267–280.

[35] *Statistical Workload Injector for Mapreduce*. Accessed: May 10, 2021. [Online]. Available: https://github.com/SWIMProjectUCB/SWIM/wiki/Workloads-repository

[36] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst.*, vol. 1, 2019, pp. 374–388.

[37] M. Xue, C. Yuan, H. Wu, Y. Zhang, and W. Liu, "Machine learning security: Threats, countermeasures, and evaluations," *IEEE Access*, vol. 8, pp. 74720–74742, 2020.

HAIWEI DONG (Senior Member, IEEE) received the M.Eng. degree in control theory and control engineering from Shanghai Jiao Tong University, Shanghai, China, in 2008, and the Ph.D. degree in computer science and systems engineering from Kobe University, Kobe, Japan, in 2010. He was a Principal Engineer with Noah's Ark Laboratory, Huawei Technologies Canada, Toronto, ON, Canada, a Research Scientist with the University of Ottawa, Ottawa, ON, a Postdoctoral Fellow with New York University, New York City, NY, USA, a Research Associate with the University of Toronto, Toronto, and a Research Fellow (PD) with Japan Society for the Promotion of Science, Tokyo, Japan. He is currently a Principal Researcher with Data Center Network Laboratory, Huawei Technologies Canada, Waterloo, ON, and a Registered Professional Engineer in Ontario Province. His research interests include artificial intelligence, multimedia communication, multimedia computing, and robotics. He also serves as a Department Editor for *IEEE Multimedia Magazine*.

ALI MUNIR received the bachelor's and M.S. degrees from NUST, Pakistan, and the Ph.D. degree from Michigan State University USA. He held a postdoctoral position with Cornell University, USA. He also worked as a Researcher with Hewlett Packard Labs, IBM Research T. J. Watson, AT&T Labs Inc., and Microsoft Research Cambridge. He is currently a Principal Engineer with the Data Center Network Laboratory, Huawei Technologies Canada, Waterloo, ON, Canada. His research interests include measurement, modeling, design, optimization, and management of networking and security systems as well as machine learning algorithms, he is dedicated to making networks efficient, intelligent, reliable and secure.

HANINE TOUT received the Ph.D. degree in software engineering from the École de Technologie Supérieure (ÉTS), Montreal, QC, Canada. She was a Research Scientist with Ericsson, Montreal. She is currently a Researcher with the DCN Laboratory, Huawei, Canada, working in the area of IP network automation and intelligence. Her research interests include AI, federated learning, machine learning, optimization, and data center networks. She is a TPC member and a reviewer of prestigious conferences and journals.

YASHAR GANJALI is currently a Chief Scientist and the Director of the Data Center Networking Laboratory, Huawei, and a Professor of computer science with the University of Toronto. His research interests include data center networking, packet switching architectures/algorithms, software defined networks, network application integration, and congestion control. He has served as a Member of the Executive Committee for ACM SIGCOMM, from 2013 to 2017, as well as an Associate Editor for IEEE/ACM TRANSACTIONS ON NETWORKING, from 2014 to 2018. He has received several awards for his research, including the Best Paper Award in Internet Measurement Conference 2008, the Best Paper Runner Up in IEEE INFOCOM 2003, the Best Demo Runner Up in SIGCOMM 2008, first and second prizes in NetFPGA Design Competition 2010, the Best Paper Award IEEE/IFIP Network Operations Management Symposium 2020, Cisco Research Award, and a Distinguished Faculty Award from Facebook.