This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges

Junfeng Xie*, F. Richard Yu†, Tao Huang*, Renchao Xie*, Jiang Liu*, Chenmeng Wang‡, and Yunjie Liu*
*State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing 100876, P.R. China
†Dept. of Systems and Computer Eng., Carleton University, Ottawa, ON, Canada
‡Chongqing Key Lab of Mobile Communications Technology
Chongqing University of Posts and Telecommunications, Chongqing 400065, P.R. China

***Abstract*—In recent years, with the rapid development of current Internet and mobile communication technologies, the infrastructure, devices and resources in networking systems are becoming more complex and heterogeneous. In order to efficiently organize, manage, maintain and optimize networking systems, more intelligence needs to be deployed. However, due to the inherently distributed feature of traditional networks, machine learning techniques are hard to be applied and deployed to control and operate networks. Software Defined Networking (SDN) brings us new chances to provide intelligence inside the networks. The capabilities of SDN (e.g., logically centralized control, global view of the network, software-based traffic analysis, and dynamic updating of forwarding rules) make it easier to apply machine learning techniques. In this paper, we provide a comprehensive survey on the literature involving machine learning algorithms applied to SDN. First, the related works and background knowledge are introduced. Then, we present an overview of machine learning algorithms. In addition, we review how machine learning algorithms are applied in the realm of SDN, from the perspective of traffic classification, routing optimization, Quality of Service (QoS)/Quality of Experience (QoE) prediction, resource management and security. Finally, challenges and broader perspectives are discussed.**

***Index Terms*—Software defined networking, machine learning, traffic classification, resource management**

I. Introduction

Recently, with the rapid development of intelligent devices (e.g., smart phones, smart cars and smart home devices) and network technologies (e.g., cloud computing and network virtualization), data traffic in our world is growing exponentially. In order to optimize the traffic distribution and manage a large number of devices, networks are becoming more heterogeneous and complex. A production network usually involves a multitude of devices, runs a multitude of protocols, and supports a multitude of applications. For example, in wireless networks, various types of cells with different transmission coverage, power and working mechanisms (e.g., macro-cells, pico-cells, femto-cells, Relays, and RRHs), and different communication technologies, such as ZigBee, WiMAX, IEEE 802.11 ac/ad, Bluetooth and LTE, have been applied. The heterogeneous network infrastructure increases the complexity of networks and poses a number of challenges in effectively organizing, managing and optimizing network resources.

Deploying more *intelligence* in networks is one possible way to solve these issues. A few years ago, a Knowledge Plane (KP) approach [1] has been proposed to bring automation, recommendation and intelligence to the Internet, by applying *Machine Learning (ML)* and cognitive techniques. However, at the time of this writing, the KP has not been prototyped or deployed. One of the major reasons is the inherently distributed feature of traditional network systems, where each node, such as router or switch, can only view and act over a small portion of the system. Learning from nodes that have only a small partial view of the complete system to perform control beyond the local domain is very complex [2]. Fortunately, recent advances in *Software Defined Networking (SDN)* will ease the complexity of learning.

SDN decouples the control plane and the data plane. The network resources in SDN are managed by a logically centralized controller, which acts as the Networking Operating System (NOS). The SDN controller can program the network dynamically. Furthermore, the centralized controller has a global view of the network by monitoring and collecting the real-time network state and configuration data, as well as packet and flow-granularity information. Applying machine learning techniques in SDN is suitable and efficient for the following reasons. First, recent advances in computing technologies such as Graphics Processing Unit (GPU) and Tensor Processing Unit (TPU) provide a good opportunity to apply promising machine learning techniques (e.g., deep neural networks) in the network field [3], [4]. Second, data is the key to the data-driven machine learning algorithms. The centralized SDN controller has a global network view, and is able to collect various network data, which facilitate the applications of machine learning algorithms. Third, based on the real-time and historical network data, machine learning techniques can bring intelligence to the SDN controller by performing data analysis, network optimization, and automated provision of network services. Finally, the programmability of SDN enables that the optimal network solutions (e.g., configuration and resource allocation) made by machine learning algorithms can

This work is jointly supported by the National Natural Science Foundation of China (No. 61501042), the Fundamental Research Funds for the Central Universities. (Corresponding author: Tao Huang.)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

be executed on the network in real time [5].

In this article, we survey the state-of-the-art machine learning techniques that can be developed and applied in SDN. Research on adopting machine learning techniques to improve the performance, smartness, efficiency and security of SDN will be discussed, followed by a brief introduction and summary of future research directions in related areas with proper depth and sufficient breadth. A road map of our approach is given in Fig. 1. As shown in the figure, we identify five aspects of the ML-based SDN, on which we would like to focus: background knowledge, overview, machine learning in SDN, challenges and broader perspectives.

The rest of the article is organized as follows. First, related works are presented in Section II. Then, background knowledge of SDN is briefly introduced in Section III. In Section IV, we give a brief explanation of the most widely-used ML algorithms in SDN. Section V reviews how ML algorithms are applied in the realm of SDN, from the perspective of traffic classification, routing optimization, Quality of Service (QoS)/Quality of Experience (QoE) prediction, resource management and security, and provides a detailed explanation of how machine learning efforts can be applied within each category. Challenges and future research directions are discussed in Section VI, including high-quality training datasets, distributed multi-controller platform, improving network security, cross-layer network optimization, and incrementally deployed SDN. In Section VII, we present some broader perspectives, such as software defined edge computing, software defined vehicular networks, software defined mobile networks, etc. Finally, we conclude this study in Section VIII.

II. Related work

The applications of machine learning have attracted a lot of attention. Patcha *et al.* [6] have given a detailed description of the applications of machine learning techniques in the domain of intrusion detection. Nguyen *et al.* [7] focus on IP traffic classification by using machine learning. Bkassiny *et al.* [8] have studied several challenging learning problems in Cognitive Radio Networks (CRNs), and surveyed existing ML-based methods to address them. How ML techniques can be applied to address common issues in wireless sensor networks has been surveyed in [9]. Wang *et al.* [10] have presented the state-of-the-art Artificial Intelligence (AI)-based techniques applied to evolve the heterogeneous networks, and discussed future research challenges. Buczak *et al.* [11] have researched on ML and Data Mining (DM) methods for cyber security intrusion detection. Klaine *et al.* [12] have surveyed the machine learning algorithms and their solutions in self organizing cellular networks, and given valuable classification and comparison. How to apply machine learning techniques to improve the network traffic control has been surveyed in [13]. Similar to [6], Hodo *et al.* [14] also focus on ML-based Intrusion Detection System (IDS). The main difference between [6] and [14] is that deep learning-based IDS has also been described detailedly in [14]. Zhou *et al.* [15] focus on using machine learning techniques and cognitive radio technology to enhance spectrum utilization and energy efficiency of wireless networks. Chen *et al.* [16] have studied the neural networks-based solutions to solve problems in wireless networks such as communication, virtual reality and edge caching. Usama *et al.* [4] have studied how to apply unsupervised learning techniques in the domain of networking.

Although machine learning techniques have been applied in various domains, no existing works focus on the applications of machine learning in the domain of SDN. To fill this gap, in this paper, we provide a comprehensive survey of machine learning techniques applied to SDN. We hope that our discussion and exploration can give readers an overall understanding of this field and foster more subsequent studies on this issue. In Table I, we provide a brief comparison of our paper with existing survey papers discussed above.

III. Background Knowledge of SDN

In this section, we present a brief background knowledge of SDN from the perspectives of architecture and workflow of SDN.

A. Architecture of SDN

SDN has attracted widespread attention in recent years. The Open Networking Foundation (ONF) [17] is a nonprofit consortium dedicated to the development, standardization, and commercialization of SDN. The ONF gives a definition of SDN as follows: "In the SDN architecture, the control plane and data plane are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications" [18].

Based on the definition, a high-level SDN architecture is presented, which is composed of three main planes, including data plane, control plane and application plane. The architectural components of each plane and their interactions are shown in Fig. 2. In the following, we will give a detailed representation of these three planes and their interactions.

1) Data Plane: The data plane, also known as infrastructure plane, is the lowest layer in SDN architecture. This plane is comprised of forwarding devices including physical switches and virtual switches. Virtual switches are software-based switches, which can run on common operating systems such as Linux. Open vSwitch [19], Indigo [20] and Pantou [21] are three implementations of virtual switches. Physical switches are hardware-based switches. There are two types of physical switches, one is implemented on open network hardware (e.g., NetFPGA [22]) and the other is implemented on networking hardware vendors' merchant switches. SwitchBlade [23] and ServerSwitch [24] are two NetFPGA-based physical switches. Nowadays, many networking hardware vendors such as HP, NEC, Huawei, Juniper and Cisco, have supported SDN protocols in their merchant switches. Virtual switches typically support complete features of SDN protocols, while physical switches lack the flexibility and feature completeness. However, physical switches generally have a higher flow forwarding rate than virtual switches.

These switches in data plane are responsible for forwarding, dropping and modifying packets based on policies received from the Control Plane (CP). They communicate with CP

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

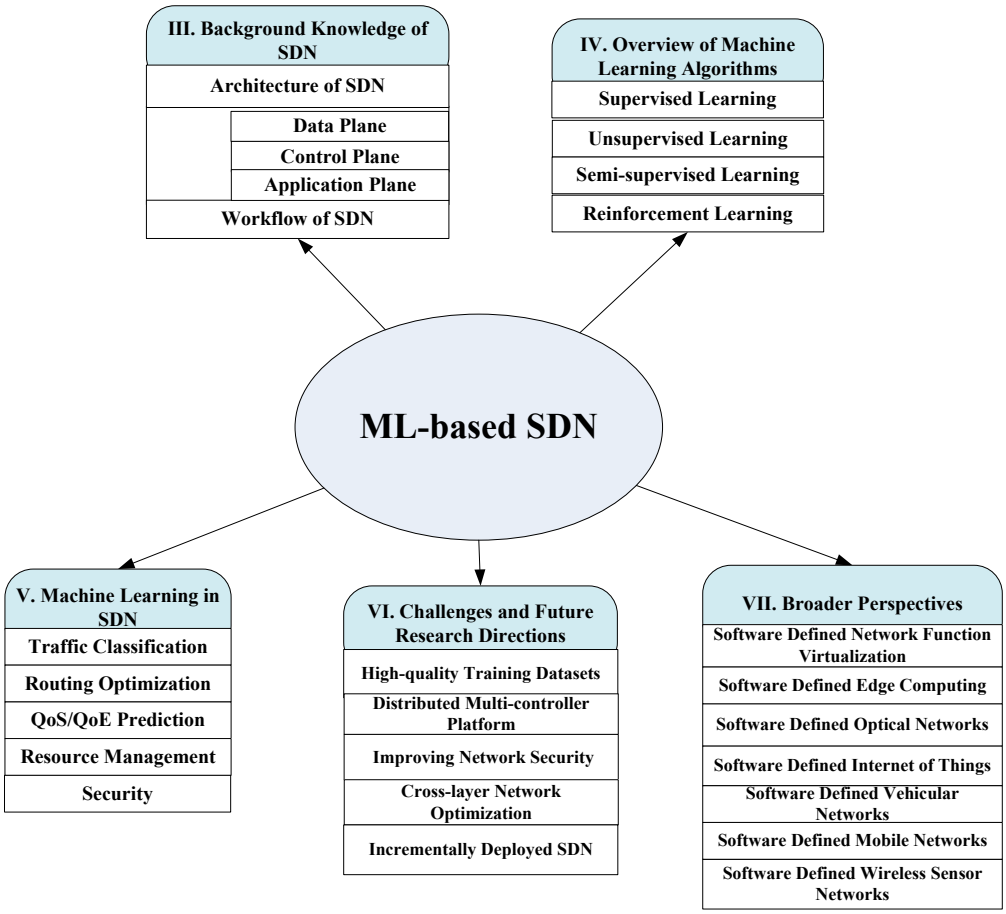


Fig. 1. Road map of ML-based SDN.

through Southbound Interfaces (SBIs), with which the CP can control the data plane's processing and forwarding capabilities.

2) Control Plane: The control plane is the "brain" of SDN systems, which can program network resources, update forwarding rules dynamically, and make network administration flexible and agile. The main component of CP is the logically centralized controller, which controls the communication between forwarding devices and applications. On one hand, the controller exposes and abstracts network state information of the data plane to the application plane. On the other hand, the controller translates the requirements from applications into custom policies and distributes them to forwarding devices. Additionally, the controller provides essential functionalities that most of network applications need, such as shortest path routing, network topology storage, device configuration and state information notifications etc. There are many controller architectures, such as NOX [25], POX [26], Floodlight [27], Ryu [28], OpenDayLight [29] and Beacon [30]. Three communication interfaces allow the controllers to interact: southbound, northbound and eastbound/westbound interfaces.

- The SBIs are also named as Control-Data-Plane Interfaces (CDPIs), which are defined between the control plane and the data plane. They allow forwarding devices to exchange network state information and control policies with the CP and provide functions such as programmatic control of all device-capability advertisements, forwarding operations, event notifications and statistics reports. To date, OpenFlow [31] promoted by ONF is the first and the most popular open standard SBI, but it is not the only one. There exist other less popular proposals such as OVSDB [32], ForCES [33], Protocol-Oblivious Forwarding (POF) [34], NETCONF [35], LISP [36], OpFlex [37] and OpenState [38].
- The Northbound Interfaces (NBIs) are defined between the control plane and the application plane. Using NBIs, applications can exploit the abstract network views provided by the CP to express network behaviors and requirements, and facilitate automation, innovation and management of SDN networks. The ONF is trying to define the standard NBIs and a common information model [39].
- The eastbound/westbound interfaces are used in the multi-controller SDN networks. When deploying SDN in large-scale networks where a vast amount of data flows need to be processed, due to the limited processing capacity of one controller, the large-scale networks are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE I
A BRIEF COMPARISON OF OUR PAPER WITH EXISTING SURVEY PAPERS.

Ref.	Published in	Year	Areas focused	ML techniques
Patcha *et al.* [6]	Computer networks	2007	Network intrusion detection	Supervised and unsupervised learning
Nguyen *et al.* [7]	IEEE COMST	2008	Internet traffic classification	Supervised and unsupervised learning
Bkassiny *et al.* [8]	IEEE COMST	2013	Cognitive radio networks	Supervised, unsupervised and reinforcement learning
Alsheikh *et al.* [9]	IEEE COMST	2014	Wireless sensor networks	Supervised, unsupervised and reinforcement learning
Wang *et al.* [10]	IEEE Access	2015	Heterogeneous networks	AI-based techniques
Buczak *et al.* [11]	IEEE COMST	2016	Cyber security intrusion detection	Supervised and unsupervised learning
Klaine *et al.* [12]	IEEE COMST	2017	Self organizing cellular networks	Supervised, unsupervised and reinforcement learning
Fadlullah *et al.* [13]	IEEE COMST	2017	Network traffic control	Deep learning
Hodo *et al.* [14]	ArXiv	2017	Network intrusion detection	Supervised and unsupervised learning
Zhou *et al.* [15]	ArXiv	2017	Wireless networks	Supervised, unsupervised and reinforcement learning
Chen *et al.* [16]	ArXiv	2017	Wireless networks	Neural networks
Usama *et al.* [4]	ArXiv	2017	Networking	Unsupervised learning
This paper	-	2018	SDN	Supervised, unsupervised and reinforcement learning

always partitioned into several domains. Each domain has its own controller. In order to provide a global network view to the upper-layer applications, the communication among multiple controllers is necessary to exchange information. The eastbound/westbound interfaces are responsible for the communication. Onix [40] and HyperFlow [41] are two distributed control architectures. Because their eastbound/westbound interfaces are private, they cannot communicate with each other. To enable the communication between different types of SDN controllers, SDNi [42], East-West Bridge [43] and Communication Interface for Distributed Control plane (CIDC) [44] have been proposed as eastbound/westbound interfaces to exchange network information. However, to the best of our knowledge, the eastbound/westbound interfaces have not yet been standardized.

3) Application Plane: The highest layer in the SDN architecture is the application plane, which is composed of business applications. These applications can provide new services and perform business management and optimization. In general, the applications can obtain the required network state information through controllers' NBIs. Based on the received information and business requirements, the applications can implement the control logic to change network behaviors.

The SDN-based applications have attracted a lot of attention from academia. Mendiola *et al.* [45] have discussed the impact of SDN on Traffic Engineering (TE) and surveyed the SDN-based TE solutions. Security in SDN has been surveyed in [46]–[51]. Especially, Yan *et al.* [50] have researched on Distributed Denial of Service (DDoS) attacks in SDN-based cloud computing systems, and discussed future research challenges. Fault management in SDN has been surveyed in [52], which gives an identification and classification of the main fault management issues, and does valuable surveys and discussions about efforts that address those issues. Guck *et al.* [53] have studied the centralized QoS routing mechanisms in SDN, and introduced a novel Four-Dimensional (4D) evaluation framework.

Due to the inherent advantages (e.g., logically centralized control, global view of the network, software-based traffic analysis, and dynamic updating of forwarding rules), SDN has been deployed in many networks, such as transport networks [54], optical networks [55], wireless networks [56], [57], Internet of Things (IoT) [58], edge computing [59], Wide Area Networks (WAN) [60], cloud computing [61], Network Function Virtualization (NFV) [62], [63].

For a more insightful discussion on SDN, please refer to [64]–[71].

B. Workflow of SDN

To understand the SDN architecture, it is important to recall its basic operation. Fig. 3 shows the working procedure of the OpenFlow-based SDN network [72]. Each OpenFlow switch has a flow table and uses the OpenFlow protocol to communicate with the SDN controller. The messages transmitted between the OpenFlow-based switches and the software-based controller are standardized by the OpenFlow protocol [31]. The flow table in the OpenFlow switch is comprised of flow

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

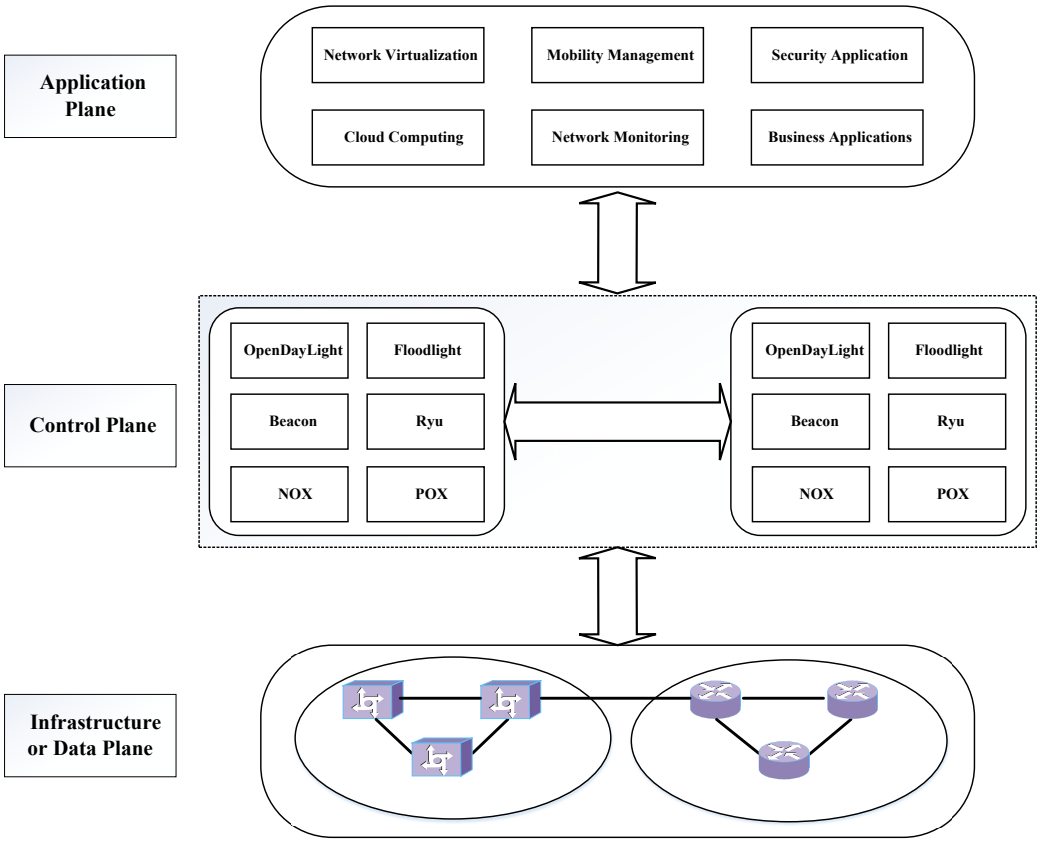


Fig. 2. The general SDN architecture. The data plane consists of physical and virtual forwarding devices. The southbound interface connects the data plane and the control plane. The control plane is the "brain" of SDN architecture. The eastbound/westbound interface enables communication among multiple controllers. The northbound interface connects the control plane and the application plane. The application plane is composed of SDN-based business applications.

entries to determine the processing actions of different packets on the data plane. When an OpenFlow switch receives a packet on the data plane, the packet header fields will be extracted and matched against flow entries. If a matching entry is found, the switch will process the packet locally according to the actions in matched flow entry. Otherwise, the switch will forward an OpenFlow PacketIn message to the controller (arrows 2 and 5). The packet header (or the whole packet, optionally) is included in the OpenFlow PacketIn message. Then, the controller will send OpenFlow FlowMod messages to manage the switch's flow table by adding flow entries (arrows 3 and 6), which can be used to process subsequent packets of the flow.

IV. Overview of Machine Learning Algorithms

Machine learning is evolved from a collection of powerful techniques in AI areas and has been extensively used in data mining, which allows the system to learn the useful structural patterns and models from training data.

A machine learning approach usually consists of two main phases: training phase and decision making phase as illustrated in the Fig. 4. At the training phase, machine learning methods are applied to learn the system model using the training dataset. At the decision making phase, the system can obtain the estimated output for each new input by using the trained model.

Machine learning algorithms are basically distinguished into four categories: supervised, unsupervised, semi-supervised and reinforcement learning, which are shown in Fig. 5. In this section, many widely-used machine learning algorithms are introduced. Each algorithm is briefly explained with some examples. For a more insightful discussion on machine learning theory and its classical concepts, please refer to [73]–[76].

A. Supervised Learning

Supervised learning is a kind of labelling learning technique. Supervised learning algorithms are given a labeled training dataset (i.e., inputs and known outputs) to build the system model representing the learned relation between the input and output. After training, when a new input is fed into the system, the trained model can be used to get the expected output [77], [78]. In the following, we will give a detailed representation of widely-used supervised learning algorithms, such as k-nearest neighbor, decision tree, random forest, neural network, support vector machine, Bayes' theory, and hidden markov models.

1) k-Nearest Neighbor (k-NN): The k-NN is a supervised learning technique, where the classification of a data sample is determined based on the k nearest neighbors of that unclassified sample. The process of the k-NN algorithm is very simple: if the most of the k nearest neighbors belong to a certain class, the unclassified sample will be classified into that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

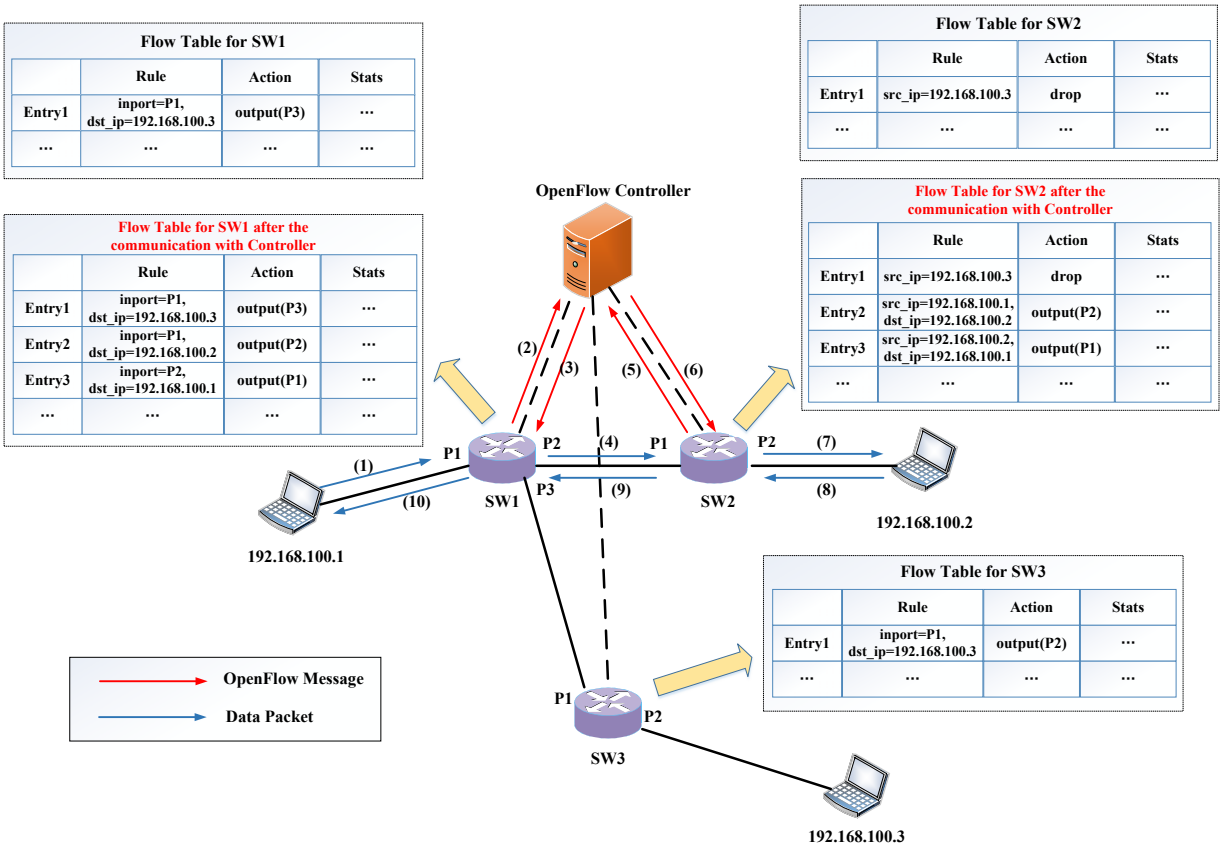


Fig. 3. OpenFlow-based SDN network. The OpenFlow controller can manage the traffic forwarding by modifying flow entries in switches' flow tables. For example, by adding two flow entries (i.e., Entry2 and Entry3) at SW1 and SW2, the communications between 192.168.100.1 and 192.168.100.2 are allowed. However, packets from 192.168.100.3 to 192.168.100.2 are denied at SW2 due to security policies.

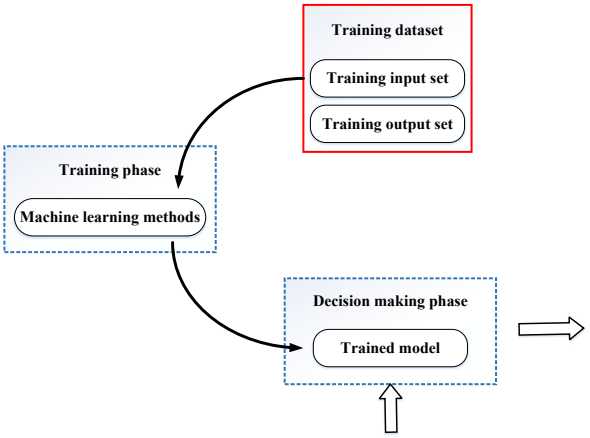


Fig. 4. The general processing procedure of a machine learning approach.

class. Fig. 6 shows a simple example of how k-NN algorithm works. Specially, when $k = 1$, it becomes the nearest neighbor algorithm. The higher the value of k is, the less effect the noise will have on the classification. Since the distance is the main metric of the k-NN algorithm, several functions can be applied to define the distance between the unlabeled sample and its neighbors, such as Chebyshev, City-block, Euclidean and Euclidean squared. For a more insightful discussion on k-NN, please refer to [79].

2) Decision Tree (DT): The DT is one of the classification techniques which performs classification through a learning tree. In the tree, each node represents a feature (attribute) of a data, all branches represent the conjunctions of features that lead to classifications, and each leaf node is a class label. The unlabeled sample can be classified by comparing its feature values with the nodes of the decision tree [80], [81]. The DT has many advantages, such as intuitive knowledge expression, simple implementation and high classification accuracy. ID3 [82], C4.5 [83] and CART [84] are three widely-used decision tree algorithms to perform the classification of training dataset automatically. The biggest difference among them is the splitting criteria which are used to build decision trees. The splitting criteria used by ID3, C4.5 and CART are Information gain, Gain ratio and Gini impurity respectively. Ref. [85] gives a detailed comparison of the three DT algorithms.

3) Random Forest: The random forest method [86], also known as random decision forest, can be used for classification and regression tasks. A random forest consists of many decision trees. To mitigate over-fitting of decision tree method and improve accuracy, the random forest method randomly chooses only a subset of the feature space to construct each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

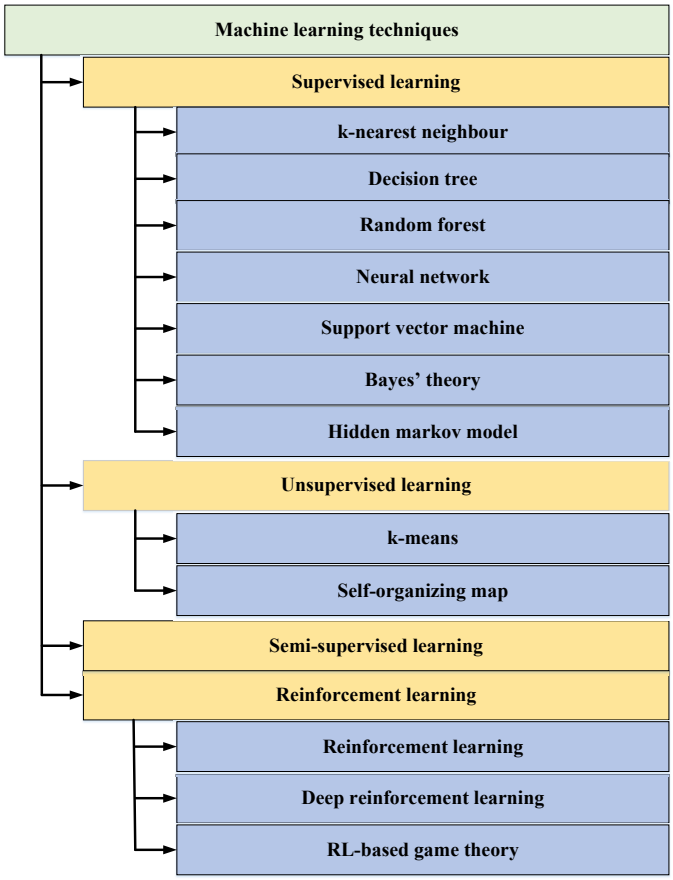


Fig. 5. Common machine learning algorithms applied to SDN.

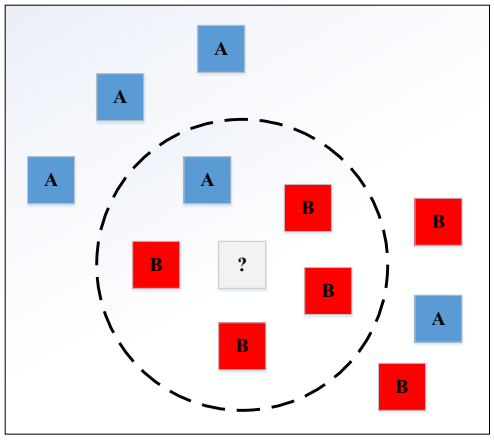


Fig. 6. Example of k-NN algorithm, for $k = 5$. Among the five closest neighbors, one neighbor belongs to class A and four neighbors belong to class B. In this case, the unlabeled example will be classified into class B.

decision tree. The steps to classify a new data sample by using random forest method are: (a) put the data sample to each tree in the forest. (b) Each tree gives a classification result, which is the tree's "vote". (c) The data sample will be classified into the class which has the most votes.

4) Neural Network (NN): A neural network is a computing system made up of a large number of simple processing

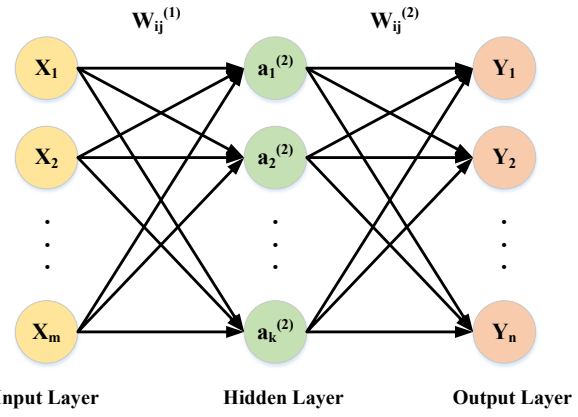


Fig. 7. A basic neural network with three layers: an input layer, a hidden layer and an output layer. An input has m features (i.e., $X_1, X_2, \ldots, X_m$) and the input can be assigned to n possible classes (i.e., $Y_1, Y_2, \ldots, Y_n$). Also, W_{ij}^l denotes the variable link weight between the ith neuron of layer l and the jth neuron of layer $l+1$, and a_k^l denotes the activation function of the kth neuron in layer l.

units, which operate in parallel to learn experiential knowledge from historical data [87]. The concept of neural networks is inspired by the human brain, which uses basic components, known as neurons to perform highly complex, nonlinear and parallel computations. In a NN, its nodes are the equivalent components of the neurons in the human brain. These nodes use activation functions to perform nonlinear computations. The most frequently used activation functions are the sigmoid and the hyperbolic tangent functions [88]. Simulating the way neurons are connected in the human brain, the nodes in a NN are connected to each other by variable link weights.

A NN has many layers. The first layer is the input layer and the last layer is the output layer. Layers between the input layer and the output layer are hidden layers. The output of each layer is the input of the next layer and the output of the last layer is the result. By changing the number of hidden layers and the number of nodes in each layer, complex models can be trained to improve the performance of NNs. NNs are widely used in many applications, such as pattern recognition. The most basic NN has three layers, including an input layer, a hidden layer and an output layer, which is shown in Fig. 7.

There are many types of neural networks, which are often divided into two training types, supervised or unsupervised [89]. In the following, we will give a detailed representation of supervised neural networks which have been applied in the field of SDN. In Subsection IV-B2, self-organizing map, a representative type of unsupervised neural networks, will be described.

a) Random NN: The random NN can be represented as an interconnected network of neurons which exchange spiking signals. The main difference between random NN and other neural networks is that neurons in random NN exchange excitatory and inhibitory spiking signals probabilistically. In random NN, the internal excitatory state of each neuron is represented by an integer, which is called "potential". The potential value of each neuron rises when it receives

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

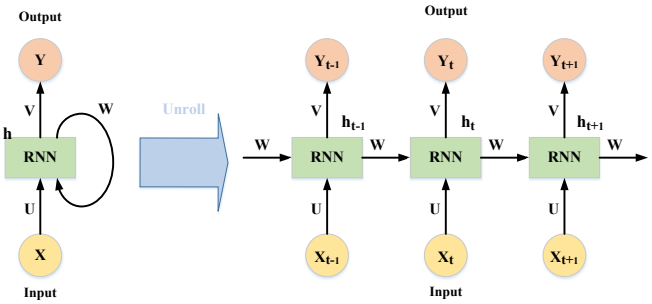


Fig. 8. A typical recurrent NN and its unrolled form. X_t is the input at time step t. h_t is the hidden state at time step t. Y_t is the output at time step t. U, V and W are parameters in the recurrent NN.

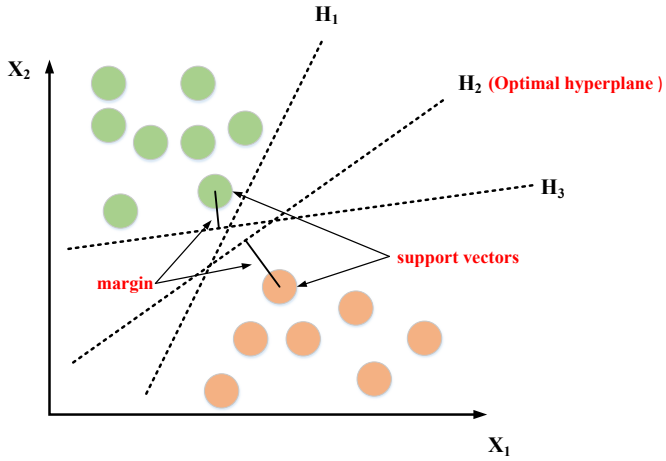


Fig. 9. An example of SVM classifier with an optimal linear hyperplane. There are two classes in the figure, and each class has one support vector. As it can be seen, there are many possible separating hyperplanes between two classes, such as H_1, H_2 and H_3, but only one optimal separating hyperplane (i.e., H_2) can maximize the margin.

an excitatory spiking signal and drops when it receives an inhibitory spiking signal. Neurons whose potential values are strictly positive are allowed to send out excitatory or inhibitory spiking signals to other neurons according to specific neuron-dependent spiking rates. When a neuron sends out a spiking signal, its potential value drops one. The random NN has been used in classification and pattern recognition [90]. For a more insightful discussion on random NN, please refer to [90]–[92].

b) Deep NN: Neural networks with a single hidden layer are generally referred to as shallow NNs. In contrast, neural networks with multiple hidden layers between the input layer and the output layer are called deep NNs [93]–[95]. For a long time, shallow NNs are often used. To process high-dimensional data and to learn increasingly complex models, deep NNs with more hidden layers and neurons are needed. However, deep NNs increase the training difficulties and require more computing resources. In recent years, the development of hardware data processing capabilities (e.g., GPU and TPU) and the evolved activation functions (e.g., ReLU) make it possible to train deep NNs [96]. In deep NNs, each layer's neurons train a feature representation based on the previous layer's output, which is known as feature hierarchy. The feature hierarchy makes deep NNs capable of handling large high-dimensional datasets. Due to the multiple-level feature representation learning, compared to other machine learning techniques, deep NNs generally provide much better performance [96].

c) Convolutional NN: Convolutional NN and recurrent NN are two major types of deep NNs. Convolutional NN [97], [98] is a feed-forward neural network. Local sparse connections among successive layers, weight sharing and pooling are three basic ideas of convolutional NN. Weight sharing means that weight parameters of all neurons in the same convolution kernel are same. Local sparse connections and weight sharing can reduce the number of training parameters. Pooling can be used to reduce the feature size while maintaining the invariance of features. The three basic ideas reduce the training difficulties of convolutional NNs greatly.

d) Recurrent NN: In feed-forward neural networks, the information is transmitted directionally from the input layer to the output layer. However, recurrent NN [99], [100] is a stateful network, which can use internal state (memory) to handle sequential data. A typical recurrent NN and its unrolled form are shown in Fig. 8. X_t is the input at time step t. h_t is the hidden state at time step t. h_t captures information about what happened in all the previous time steps, so it is called "memory". Y_t is the output at time step t. U, V and W are parameters in the recurrent NN. Unlike a traditional deep NN, which uses different parameters at each layer, the recurrent NN shares the same parameters (i.e., U, V and W) across all time steps. This means that at each time step, the recurrent NN performs the same task, just with different inputs. In this way, the total number of parameters needed to be trained is reduced greatly. Long Short-Term Memory (LSTM) [101], [102] is the most commonly-used type of recurrent NNs, which has a good ability to capture long-term dependencies. LSTM uses three gates (i.e., an input gate, an output gate and a forget gate) to compute the hidden state.

5) Support Vector Machine (SVM): SVM is another popular supervised learning method, invented by Vapnik and others [103], which has been widely used in classification and pattern recognition. The basic idea of SVM is to map the input vectors into a high-dimensional feature space. This mapping is achieved by applying different kernel functions, such as linear, polynomial and Radial Based Function (RBF). Kernel function selection is an important task in SVM, which has effect on the classification accuracy. The selection of kernel function depends on the training dataset. The linear kernel function works well if the dataset is linearly separable. If the dataset is not linearly separable, polynomial and RBF are two commonly-used kernel functions. In general, the RBF-based SVM classifier has a relatively better performance than the other two kernel functions [104], [105].

The objective of SVM is to find a separating hyperplane in the feature space to maximize the margin between different classes. Note that, the margin is the distance between the hyperplane and the closest data points of each class. The corresponding closest data points are defined as support vectors. An example of SVM classifier is shown in Fig. 9. From the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

figure, there are many possible separating hyperplanes between two classes, but only one optimal separating hyperplane can maximize the margin. For a more insightful discussion on SVM, please refer to [106]–[108].

6) Bayes' Theory: Bayes' theory uses the conditional probability to calculate the probability of an event occurring given the prior knowledge of conditions that might be related to the event. The Bayes' theory is defined mathematically as the following equation:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{1}$$

where E is a new evidence, H is a hypothesis, $P(H|E)$ is the posterior probability that the hypothesis H holds given the new evidence E, $P(E|H)$ is the posterior probability that of evidence E conditioned on the hypothesis H, $P(H)$ is the prior probability of hypothesis H, independent of evidence E, and $P(E)$ is the probability of evidence E.

In a classification problem, the Bayes' theory learns a probability model by using the training dataset. The evidence E is a data sample, and the hypothesis H is the class to assign for the data sample. The posterior probability $P(H|E)$ represents the probability of a data sample belonging to a class. In order to calculate the posterior probability $P(H|E)$, $P(H)$, $P(E)$ and $P(E|H)$ need to be calculated first based on the training dataset using the probability and statistics theories, which is the learning process of the probability model. When classifying a new input data sample, the probability model can be used to calculate multiple posterior probabilities for different classes. The data sample will be classified into the class with the highest posterior probability $P(H|E)$. The advantage of the Bayes' theory is that it requires a relatively small number of training dataset to learn the probability model [109]. However, there is an important independence assumption when using the Bayes' theory. To facilitate the calculation of $P(E|H)$, the features of data samples in the training dataset are assumed to be independent of each other [110]. For a more insightful discussion on Bayes' theory, please refer to [109], [111]–[114].

7) Hidden Markov Models (HMM): HMM is one kind of Markov models. Markov models are widely used in randomly dynamic environments which obey the memoryless property. The memoryless property of Markov models means that the conditional probability distribution of future states only relates to the value of the current state and is independent of all previous states [115], [116]. There are other Markov models, such as Markov Chains (MC). The main difference between HMM and other models is that HMM is often applied in environments where system states are partially visible or not visible at all.

B. Unsupervised Learning

In contrast to supervised learning, an unsupervised learning algorithm is given a set of inputs without labels (i.e., there is no output). Basically, an unsupervised learning algorithm aims to find patterns, structures, or knowledge in unlabeled data by clustering sample data into different groups according to the similarity between them. The unsupervised learning techniques are widely used in clustering and data aggregation [76], [78]. In the following, we will give a detailed representation of widely-used unsupervised learning algorithms, such as k-means and self-organizing map.

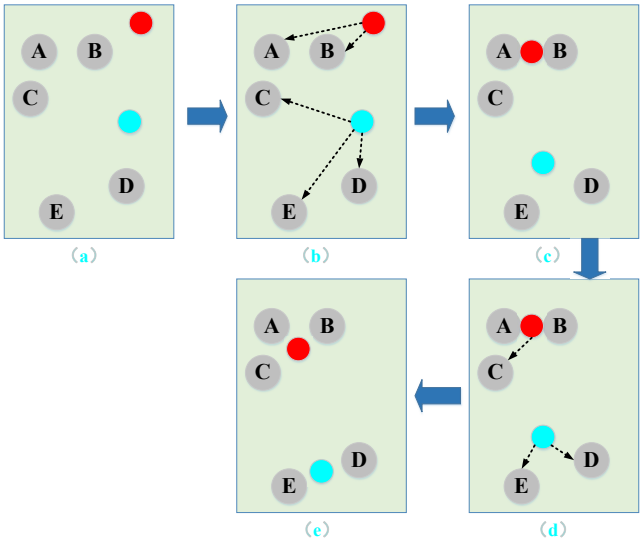


Fig. 10. Example of k-means algorithm, for $k = 2$. (a) Randomly choosing two data points as two centroids; (b) label each node with the closest centroid, resulting that node A and B are a class, node C, D and E are another class; (c) assign new centroids; (d) label each node with the closest centroid again, resulting that node A, B and C are a class, node D and E are another class; (e) the algorithm is converged.

1) k-Means: The k-means algorithm is a popular unsupervised learning algorithm, which is used to recognize a set of unlabeled data into different clusters. To implement the k-means algorithm, only two parameters (i.e., the initial dataset and the desired number of clusters) are needed. If the desired number of clusters is k, the steps to resolve node clustering problem by using k-means algorithm are: (a) initialize k cluster centroids by randomly choosing k nodes; (b) use a distance function to label each node with the closest centroid; (c) assign new centroids according to the current node memberships and (d) stop the algorithm if the convergence condition is valid, otherwise go back to step (b). An example procedure of k-means algorithm is shown in Fig. 10. For a more insightful discussion on k-means, please refer to [78], [117].

2) Self-Organizing Map (SOM): SOM, also known as Self-Organizing Feature Map (SOFM), is one of the most popular unsupervised neural network models. SOM is often applied to perform dimensionality reduction and data clustering. In general, SOM has two layers, an input layer and a map layer. When SOM is used to perform data clustering, the number of neurons in the map layer is equal to the desired number of clusters. Each neuron has a weight vector. The steps to resolve data clustering problem by using SOM algorithm are: (a) initialize the weight vector of each neuron in the map layer; (b) choose a data sample from the training dataset; (c) use a distance function to calculate the similarity between the input data sample and all weight vectors. The neuron whose weight vector has the highest similarity is called the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

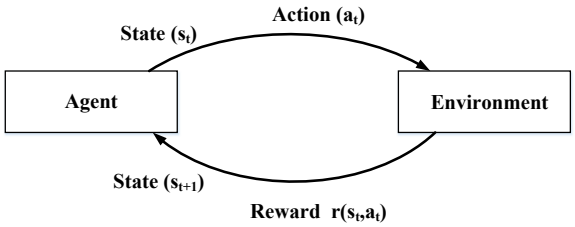


Fig. 11. A basic diagram of a RL system. The agent takes an action according to the current state and then receives a reward. $r(\mathrm{s}_t, a_t)$ denotes the immediate reward that the agent receives after performing an action a_t at the state s_t.

Best Matching Unit (BMU). The SOM algorithm is based on competitive learning, which means that there is only one BMU each time. (d) The neighborhood of the BMU is calculated. (e) The weight vectors of the neurons in the BMU's neighborhood (including the BMU itself) are adjusted towards the input data sample. (f) Stop the algorithm if the convergence condition is valid, otherwise go back to step (b). For a more insightful discussion on SOM, please refer to [118], [119].

C. Semi-supervised Learning

Semi-supervised learning [120], [121] is a type of learning which uses both labeled and unlabeled data. Semi-supervised learning is useful for a few reasons. First, in many real-world applications, the acquisition of labeled data is expensive and difficult while acquiring a large amount of unlabeled data is relatively easy and cheap. Second, effective use of unlabeled data during the training process actually tends to improve the performance of the trained model. In order to make the best use of unlabeled data, assumptions have to be hold in semi-supervised learning, such as smoothness assumption, cluster assumption, low-density separation assumption, and manifold assumption. Pseudo Labeling [122], [123] is a simple and efficient semi-supervised learning technique. The main idea of Pseudo Labeling is simple. Firstly, use the labeled data to train a model. Then, use the trained model to predict pseudo labels of the unlabeled data. Finally, combine the labeled data and the newly pseudo-labeled data to train the model again. There are other semi-supervised learning methods, such as Expectation Maximization (EM), co-training, transductive SVM and graph-based methods. Different methods rely on different assumptions [124]. For example, EM builds on cluster assumption, transductive SVM builds on low-density separation assumption, while graph-based methods build on the manifold assumption.

D. Reinforcement Learning

1) Reinforcement Learning (RL): Another quite popular learning technique is RL [125], [126]. RL involves an agent, a state space S and an action space A. The agent is a learning entity which interacts with its environment to learn the best action to maximize its long-term reward. The long-term reward is a cumulative discounted reward and relates to both the immediate reward and future rewards. When applying RL to SDN, the controller generally works as an agent and the network is the environment. The controller monitors the network status and learns to make decisions to control data forwarding. Fig. 11 shows a basic diagram of a RL system. Specifically, at each time step t, the agent monitors a state s_t and chooses an action a_t from the action space A, receives an immediate reward r_t which indicates how good or bad the action is, and transitions to the next state s_{t+1}. The objective of the agent is to learn the optimal behavior policy π which is a direct map from the state space S to the action space A ($\pi : S \rightarrow A$) to maximize the expected long-term reward. From the behavior policy π, the agent can determine the best corresponding action given a particular state. In RL, value function is used to calculate the long-term reward of an action given a state. The most well-known value function is Q-function, which is used by Q-learning to learn a table storing all state-action pairs and their long-term rewards [127].

2) Deep Reinforcement Learning (DRL): The main advantage of RL is that it works well without prior knowledge of an exact mathematical model of the environment. However, the traditional RL approach has some shortcomings, such as low convergence rate to the optimal behavior policy π and its inability to solve problems with high-dimensional state space and action space. These shortcomings can be addressed by DRL [128]–[130]. The key idea of DRL is to approximate the value function by leveraging the powerful function approximation property of deep NNs. After training the deep NNs, given a state-action pair as input, DRL is able to estimate the long-term reward. The estimation result can guide the agent to choose the best action.

3) RL-based Game Theory: Game theory is a mathematical tool that focuses on strategic interactions among rational decision-makers. A game generally involves a set of players, a set of strategies and a set of utility functions. Players are decision-makers. Utility functions are used by players to select optimal strategies. Cooperative game theory and non-cooperative game theory are two branches of game theory. In cooperative games, players cooperate and form multiple coalitions. Players choose strategies that maximize the utility of their coalitions. On the contrary, in non-cooperative games, players compete against each other and choose strategies individually to maximize their own utility. In the network field, it is often assumed that nodes are selfish [131]–[137]. Thus, in this paper, we only focus on non-cooperative game theory.

In non-cooperative games, players do not communicate with each other, and at the beginning of each play round, players do not have any information about the strategies selected by the other players. At the end of each play round, all players broadcast their selected strategies, which are the only external information. However, each player's utility can be affected by the other players' strategies. In this case, adaptive learning methods should be used to predict the strategies of the other players, based on which each player chooses its optimal strategy. RL is a widely-used adaptive learning method, which can help players select their optimal strategies by learning from historical information such as network status, the other players' strategies and the corresponding utility [132], [134], [138]. Thus, RL-based game theory is an effective decision-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

making technique.

In summary, supervised learning algorithms are generally applied to conduct classification and regression tasks, while unsupervised and reinforcement learning algorithms are applied to conduct clustering and decision-making tasks respectively. In order to give readers a better understanding of machine learning algorithms, we provide a comparison of pros and cons of all the machine learning algorithms discussed above in Table II.

V. Machine Learning in SDN

The centralized SDN controller has a global network view, which makes the network easy to control and manage. Machine learning techniques can bring intelligence to the SDN controller by performing data analysis, network optimization, and automated provision of network services. In other words, the learning capability enables the SDN controller to autonomously learn to make optimal decisions to adapt to the network environments. In this section, we review existing machine learning efforts to address issues in SDN, such as traffic classification, routing optimization, QoS/QoE prediction, resource management and security. We will give readers a summary on how ML algorithms are applied in the realm of SDN.

A. Traffic Classification

Traffic classification is an important network function, which provides a way to perform fine-grained network management by identifying different traffic flow types. With the help of traffic classification, network operators can handle different services and allocate network resources in a more efficient way.

The widely-used traffic classification techniques include port-based approach, Deep Packet Inspection (DPI) and machine learning [139]–[141]. Port-based approach uses TCP and UDP port numbers to determine applications. In the past, many applications used well-known ports such as TCP port 80 for HTTP protocol. Nowadays, most applications run on dynamic ports, which makes the port-based approach no longer effective.

DPI matches the payload of traffic flows with predefined patterns to identify the applications that traffic flows belong to. The patterns are defined by regular expressions. The DPI-based approach generally has high classification accuracy. However, it has some shortcomings. First, DPI can only recognize applications whose patterns are available. The exponential growth of applications makes the pattern update difficult and impractical. Second, DPI incurs high computational cost as all traffic flows need to be checked. Third, DPI cannot classify encrypted traffic on the Internet.

ML-based approaches can correctly recognize encrypted traffic and incur much lower computational cost than DPI-based approach. Thus, ML-based approaches have been extensively studied. To do traffic classification, a large number of traffic flows are first collected, and then ML techniques are applied to extract knowledge from the collected traffic flows. In SDN, the controller has a global network view, which facilitates the traffic collection and analysis. Thus, the ML-based approaches are generally implemented in the controller. Many studies have been done to classify traffic from different perspectives, such as elephant flow-aware, application-aware and QoS-aware traffic classification. In this subsection, we will summarize related studies.

1) Elephant Flow-aware Traffic Classification: Elephant flow-aware traffic classification aims to identify the elephant flows and the mice flows. Elephant flows are the long-lived, bandwidth-hungry flows, while mice flows are the short-lived, delay-intolerant flows. In a data center, 80% of the traffic flows are mice flows. However, the majority of bytes are carried in elephant flows [142]. To control the traffic flows in data centers effectively, it is necessary to identify the elephant flows.

Ref. [143] studies the traffic flow scheduling issue in a hybrid data center network. First, machine learning techniques are used to do elephant flow-aware traffic classification at the edge of the network. Then, the centralized SDN controller can utilize the classification result to implement efficient traffic flow optimization algorithms.

In [144], a cost-sensitive learning method is proposed in SDN to detect elephant flows. The proposed elephant flow detection strategy is composed of two stages. In the first stage, head packet measurement is adopted to distinguish suspicious elephant flows from mice flows. In the second stage, decision tree is used as the detection method to analyze whether these suspicious elephant flows are elephant flows or not.

2) Application-aware Traffic Classification: Application-aware traffic classification aims to identify the applications of traffic flows. In [139], the authors study the application-aware traffic classification in the enterprise network. A simple OpenFlow-based SDN system is deployed in an enterprise network to collect traffic data. Then several classifier algorithms are applied to classify traffic flows into different applications.

Ref. [145] proposes MultiClassifier to identify applications by combining ML-based classifier and DPI-based classifier. Upon a new flow arrival, ML-based classifier is first selected to do the classification. If the reliability of ML-based classifier's result is larger than a threshold value, it will be the MultiClassifier's result directly. Otherwise, the DPI-based classification will be done. If DPI-based classifier does not return "UNKNOW", its result will be selected as the MultiClassifier's result.

Ref. [146] focuses on the classification of applications running over UDP protocol. A behavioral classification engine is proposed to give an accurate application-aware traffic classification. Specifically, SVM algorithm is used to classify UDP traffic according to Netflow records (e.g., the counts of received packets and bytes). Simulation results demonstrate that the classification accuracy of the proposed SVM-based classification engine is over 90%.

Ref. [140] focuses on the mobile application classification. A framework, called Atlas, is proposed to identify the mobile applications. A crowd sourcing approach is used to collect ground truth data from end devices. The collected data is used to train the decision tree. The trained model is able to identify the mobile applications of traffic flows. Simulation results demonstrate that the average classification accuracy of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE II
ADVANTAGES AND SHORTCOMINGS OF THE ML ALGORITHMS.

ML algorithm	Problem type	Advantages	Shortcomings
k-NN	Classification, regression	• Simple to implement • Flexible to choose distance functions	• Computationally expensive due to the distance calculation of each training data sample to classify a new sample • Memory-intensive to store all the training dataset
DT	Classification, regression	• Simple to understand decisions • Ability of selecting the most discriminatory features • Data classification without much calculation • Handling both continuous and discrete data	• Instability: even a small change in the training dataset can result in large changes of the DT model • Over-fitting: noise can cause the over-fitting of the DT model
Random forest	Classification, regression	• Work well on large training dataset • Reduced instability (relative to DT) • Mitigate the over-fitting of the DT model	• Low training speed • Less effective to train imbalanced dataset
Neural network	Classification, regression	• Once trained, prediction is fast • Able to approximate an arbitrary function • Work well on high-dimensional training dataset	• Computationally expensive due to the requirement for tuning a large number of parameters • Hard for humans to interpret the trained NN model • There is little theory to guide researchers to set the optimal NN structure (e.g., the number of layers and nodes)
SVM	Classification, regression	• Handle high-dimensional dataset well • Work well on both linearly separable and non-linearly separable dataset	• Hard to train large dataset because the training is computationally expensive • Less effective on noisier dataset due to over-fitting issues
Bayes' theory	Classification	• Simple to understand and implement • Easy to train the model, even with a small training dataset	• Rely on independence assumption, which is not always the case in real world • Hard to handle continuous data
HMM	Classification	• Strong statistical foundation • Work well in environments where system states are partially visible	• Hard to train large dataset because the training is computationally expensive • Instability: even a small change of the HMM parameters can affect the performance of the HMM model dramatically
k-Means	Clustering	• Simple to implement • Easy to interpret the clustering results	• Sensitive to initial points and outliers • Computational cost is linear with the number of training data
SOM	Clustering	• Easy to understand the data mapping • Capable of handling high-dimensional dataset well	• Computationally expensive, especially for large maps with lots of training data
Semi-supervised learning	Classification, regression, clustering	• Use both labeled and unlabeled data	• Rely on assumptions such as manifold, cluster, and smoothness assumptions
RL	Decision-making	• Work well without prior knowledge of an exact mathematical model of the environment • Once trained, decision making is fast	• Low convergence rate to the optimal behavior policy π • Hard to solve problems with high-dimensional state space and action space
DRL	Decision-making	• Can handle problems with high-dimensional state space and action space • Rely on the powerful function approximation and representation learning properties of deep neural networks • After training, decision making is fast	• Require lots of computational resources to train the model

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

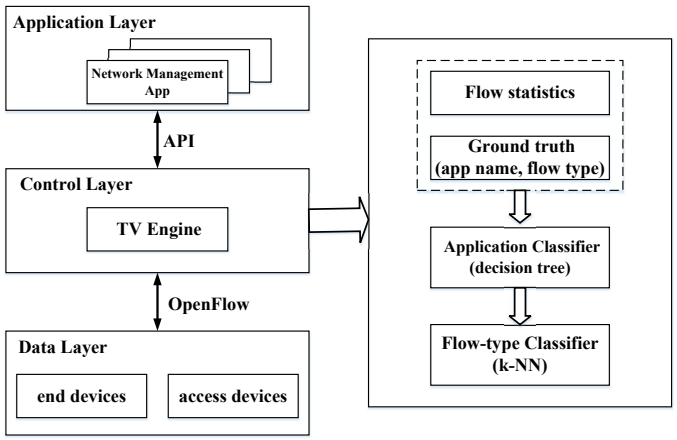


Fig. 12. High-level SDN-enabled TrafficVision architecture and the workflow of TV Engine [148]. TV Engine has three major tasks: (1) Collecting flow statistics and ground-truth training data from end devices and access devices. (2) A decision tree classifier is applied to identify application name. (3) A k-NN classifier is applied to identify flow types.

Atlas is over 94% for the top 40 applications on Google Play.

In [147], deep NN is used to identify mobile applications. Mobile network traffic is collected from an experimental network. Five flow features (i.e., destination address, destination port, protocol type, TTL and packet size) are selected to train a 8-layer deep NN model. Simulation results demonstrate that the trained model can achieve 93.5% accuracy for the identification of 200 mobile applications.

A mobile application often generates various flow types. For example, Facebook application can generate video, voice, Instant Messaging (IM) and file sharing flows, and so on. To provide a fine-grained mobile application-aware traffic classification, it is necessary to identify both the mobile applications and the flow types. In [148], a fine-grained mobile application-aware traffic classification system, called TrafficVision, is proposed in SDN-enabled wireless edge network. TrafficVision Engine (TV Engine) is the main component of TrafficVision. The high-level SDN-enabled TrafficVision architecture and the workflow of TV Engine are shown in Fig. 12. The TV Engine has three major tasks: (1) Collecting, storing and extracting flow statistics and ground-truth training data from end devices and access devices. (2) A decision tree classifier is used to identify the application name, such as YouTube, Facebook, Amazon etc. (3) A k-NN classifier is applied to identify the flow types, such as video content, audio file, video chat etc.

3) QoS-aware Traffic Classification: QoS-aware traffic classification aims to identify the QoS classes of traffic flows. With the exponential growth of applications on the Internet, it is difficult and impractical to identify all the applications. However, applications can be divided into different QoS classes according to their QoS requirements (e.g., delay, jitter and loss rate). Many different applications may belong to a QoS class. Thus, it is a more effective way to classify traffic flows according to their QoS requirements.

In [141], a QoS-aware traffic classification system is proposed by leveraging semi-supervised learning algorithm and DPI. DPI is applied to label a part of traffic flows of known applications. Then the labeled training dataset is used by semi-supervised learning algorithms, such as Laplacian SVM, to classify the traffic flows of unknown applications. In this way, traffic flows of both known and unknown applications are categorized into different QoS classes. Simulation results indicate that the system has a high classification accuracy (i.e., over 90%).

4) Analysis: From these related studies on the traffic classification, we can give the following analysis and summary.

- Elephant flow-aware traffic classification is often applied in data centers. One main objective of data centers is to schedule traffic flows rapidly. Fine-grained traffic classification methods (i.e., application-aware and QoS-aware traffic classification) can increase the traffic processing delay, so they are not suitable for data centers.
- Application-aware traffic classification is often applied for the fine-grained network management. However, with the exponential growth of applications on the Internet, it is impractical to identify all the applications. Existing works only identify the most popular applications. For example, [139] uses ML approaches to identify the widely-used eight applications. [140] focuses on the top 40 applications on Google Play.
- QoS-aware traffic classification can be used by network operators to optimize network resource allocation for traffic flows according to their desired QoS.
- In general, supervised and semi-supervised learning algorithms can be used for the traffic classification. Supervised learning algorithms need a labeled training dataset, in which traffic flows are labeled with known classes, such as elephant flow, applications or QoS classes. DPI is a common method to label traffic flows, but it incurs high computational cost when a large number of traffic flows are labeled. Moreover, ever-increasing new applications also make supervised learning algorithms less effective. On the contrary, semi-supervised learning algorithms only need a small part of labeled data, so they are more effective to do fine-grained traffic classification.
- The performance of supervised learning algorithms depends on training datasets. Thus, in Table III, we provide a detailed comparison of the related works discussed above, from the perspectives of objective, learning model, training dataset input and output. In Table VIII, we summarize the performance of the ML-based traffic classification solutions, from the perspectives of complexity, lower bound, upper bound, and average classification accuracy.
- Because the experiments are conducted based on different training datasets, we cannot compare the performance of different supervised learning algorithms directly. The experiments in [140], [141], [144] show that the classification accuracy is related to the dimension and volume of training datasets. In general, with the increasing of dimension and volume of training datasets, the classification accuracy can be improved. Compared with conventional machine learning algorithms, deep learning algorithms are more suitable for processing large high-dimensional training datasets due to the ability of feature learning and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

hierarchical feature extraction [147]. However, the deeper the neural network is, the more time it takes to train the neural network model. In other words, the deeper neural network has higher complexity.

B. Routing Optimization

Routing is a fundamental network function. In SDN, the controller can control the routing of traffic flows by modifying flow tables in switches. For example, the controller can guide switches to discard a traffic flow or route it through a specific path. Inefficient routing decisions can lead to the over-loading of network links and increase the end-to-end transmission delay, which affect the overall performance of SDN. Thus, how to optimize the routing of traffic flows is an important research problem.

Shortest Path First (SPF) algorithm and heuristic algorithms [149] are two types of widely-used routing optimization approaches. SPF algorithm routes packets according to simple criteria such as hop-count or delay. Despite its simplicity, the SPF algorithm is a best-effort routing protocol and does not make the best use of network resources [150]. Heuristic algorithms (e.g., ant colony optimization algorithm) are another approach to solve the routing optimization problem. The high computational complexity is the main shortcoming of heuristic algorithms [150], [151].

In SDN, the controller is responsible for calculating the routing policy for each new flow. In this case, heuristic algorithms are not suitable because they increase the computational burden of the controller. Many studies have tried to solve the routing optimization problem using machine learning algorithms. Compared with heuristic algorithms, machine learning algorithms have some advantages. On one hand, once trained, machine learning algorithms can give the near-optimal routing solutions quickly. On the other hand, machine learning algorithms do not need an exact mathematical model of the underlying network. The routing optimization problem can be considered as a decision-making task. Thus, reinforcement learning is an effective approach. Supervised learning algorithms are also applied by many studies to optimize routing. In the subsection, we will summarize related studies on routing optimization.

1) Supervised Learning-based Routing Optimization: Labeled training datasets are the foundation of supervised learning algorithms. In general, when applying supervised learning algorithms to optimize routing, the network and traffic states are often the input of the training datasets, and the corresponding routing solutions of heuristic algorithms are the output. By training supervised learning algorithms, the optimal heuristic-like routing solutions can be obtained in real time.

In [150], an architecture with supervised ML-based meta-layer, shown as Fig. 13, is presented to solve the dynamic routing problem. The ML-based meta-layer uses training dataset, which includes the input of heuristic algorithm and its corresponding output, to obtain real-time heuristic-like results.

Ref. [151] proposes a dynamic routing framework called NeuRoute. In NeuRoute, LSTM is used to estimate future network traffic. Then, the network state and the estimated network traffic as the input, and the corresponding routing solution calculated by heuristic algorithms as output, are used to train the neural network model. After training, the trained neural network model can be applied to obtain the real-time heuristic-like results.

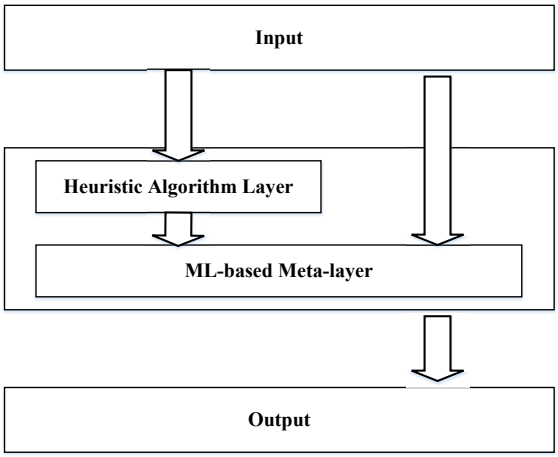


Fig. 13. A dynamic routing architecture [150]. The architecture consists of a ML-based meta-layer and a heuristic algorithm layer. The input of heuristic algorithm and its corresponding output are the training dataset of the ML-based meta-layer. After the training phase, the ML-based meta-layer can make the optimal routing decisions directly and independently.

2) RL-based Routing Optimization: RL algorithms are generally applied to solve decision-making problems. When applying RL algorithms to optimize routing, the controller works as an agent and the network is the environment. The state space is composed of network and traffic states. The action is the routing solution. The reward is defined based on optimization metrics such as network delay.

The work in [152] presents a distributed intelligent routing protocol in SDN by using RL method. The proposed routing protocol is able to select the optimal data transmission paths according to the network status.

In [153], the authors study the routing optimization scheme in the SDN-enabled inter-data center overlay network. A logically centralized Cognitive Routing Engine (CRE) is proposed to find the optimal overlay paths between geographically-dispersed data centers, by using random NN and RL. The SDN-enabled overlay network with CRE is shown in Fig. 14. Based on random NN and RL, the proposed CRE can work well even in highly chaotic environments.

Ref. [155] focuses on the routing optimization in multi-layer hierarchical SDN. A QoS-aware Adaptive Routing (QAR) method is presented to enable time-efficient adaptive packet forwarding by utilizing RL algorithm. The routing path with the maximum QoS-aware reward is selected based on traffic types and users' applications.

In [156], DRL model is applied to optimize routing. The objective of the DRL model is to select the optimal routing paths for all source-destination pairs given the traffic matrix to minimize the network delay.

3) Traffic Prediction: Traffic prediction is an important research issue in the field of routing optimization. Traffic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE III
ML-BASED TRAFFIC CLASSIFICATION SOLUTIONS IN SDN.

Ref.	Objective	Learning model	Dataset input	Dataset output
[144]	Elephant flow-aware classification	Decision tree	Basic five-tuple and statistical flow features	Elephant flow, mice flow
[139]	Application-aware classification	Random forest	12 flow features such as packet size, packet time stamp, inter-arrival time, flow duration, and so on	8 applications: YouTube, Vimeo, Facebook, Linkedin, Skype, Bittorrent, Web Browsing (HTTP) and Dropbox
[145]	Application-aware classification	ML classifier	Not mentioned	Not mentioned
[146]	Application-aware classification	SVM	Basic five-tuple and Netflow records (e.g., the counts of received packets and bytes)	8 applications: PPlive, TVAnts, SopCast, Joost, Edonkey, BitTorrent, Skype, DNS
[140]	Application-aware classification	Decision tree	Flow features such as the sizes of the first 'N' packets, source and destination ports and IP addresses	Top 40 applications in Google Play Store
[147]	Application-aware classification	Deep NN	Destination address, destination port, protocol type, TTL and packet size	200 mobile applications
[148]	Application-aware classification	Decision tree, k-NN	Flow features summarized in [7]	Dataset 1: top 37 applications in Google Play Store. Dataset 2: 45 applications
[141]	QoS-aware classification	Semi-supervised learning	9 flow features such as the average packet inter-arrival time, Hurst parameter, port number, and so on	Four QoS classes: voice/video conference, interactive data, streaming, bulk data transfer

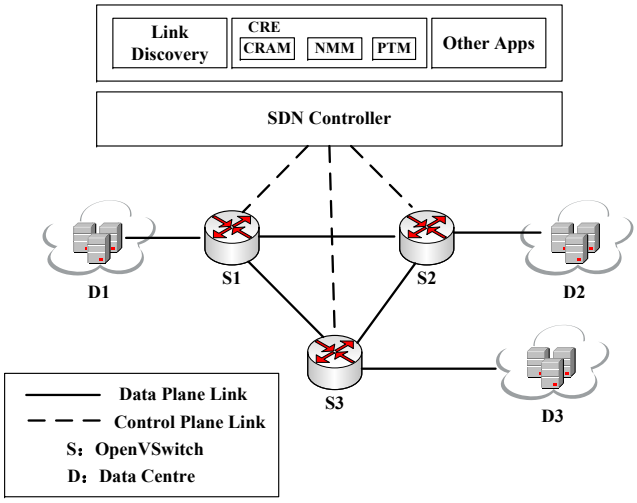


Fig. 14. The SDN-enabled overlay network with CRE [153]. CRE consists of three main modules: CRAM, NMM and PTM [154]. CRAM stands for Cognitive Routing Algorithm Module, NMM for Network Monitoring Module, PTM for Pathto-OF Translator Module. CRAM uses random NN and RL to find network paths which maximize the objective function set by the tenant's traffic engineering policy. NMM is in charge of collecting required network state information to update the random NN in the CRAM. PTM is responsible for converting the optimal paths found by the CRAM into the appropriate OpenFlow messages, which guide the SDN controller to reroute the network traffic.

prediction aims to predict the trend of traffic volume by analyzing the historical traffic information [157]. Based on traffic prediction results, the SDN controller can make traffic routing decisions in advance and distribute the proactive routing policies to forwarding devices in the data plane to guide traffic flow routing in the near future. In this way, the SDN controller can take appropriate actions before traffic congestion occurs. Furthermore, traffic prediction can facilitate the proactive provision of network resources to improve QoS.

Ref. [158] focuses on the software defined mobile metro-core network and studies the dynamic optical routing issue. To solve the problem, the authors propose a dynamic optical routing matheuristic algorithm, which includes three phases: Off-Line Scheduling, Off-Line Planning and On-Line Routing. Neural network is used in the Off-Line Scheduling phase to forecast the network traffic load. The prediction results are used to calculate the optimal resource allocation according to the predicted traffic load in advance. Then the system uses a minimum-cost path algorithm to make on-line routing decisions.

In [159], a load balance strategy is proposed to optimize path load. Four flow features (i.e., transmission hop, transmission latency, packet loss rate and bandwidth utilization ratio) are selected by the SDN controller to predict the load of each path using neural network model. Then, the least loaded path will be selected as the transmission path of new traffic flows.

Ref. [160] proposes a LSTM-based framework called

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

NeuTM to predict network traffic matrix. Real-world traffic data from the GEANT backbone network [161] is used to train the LSTM model. Simulation results demonstrate that the LSTM model converges quickly and has a good prediction performance.

4) Others: Ref. [162] focuses on the routing protocol in environments with strict compliance requirements. An efficient risk-based swarm routing protocol is proposed. The proposed protocol first uses k-means algorithm to cluster network traffic into several clusters of risk ratios in an off-line mode. Then, the Ant Colony Optimization (ACO) is used to select paths with minimized privacy exposure and compliance risks for a given data transmission session in an on-line mode.

5) Analysis: In Table IV, we provide a comparison of the related works discussed above. From the related studies on the routing optimization, we can give the following analysis and summary.

- Supervised learning algorithms, especially neural networks, are effective to obtain the optimal heuristic-like routing solutions. However, the main shortcoming is that the acquisition of labeled training datasets has high computational complexity.
- Compared with supervised learning algorithms, RL algorithms have some advantages. On one hand, RL algorithms do not need labeled training datasets. On the other hand, optimization targets (e.g., energy efficiency, throughput and delay) can be adjusted flexibly through different reward functions.
- Traffic prediction can promote the implementation of routing pre-design, which is an effective way to reduce the transmission delay in the data plane by modifying switches' flow tables in advance. Neural network models, especially LSTM, are often used for traffic prediction.
- The experiments in [151] show that the optimal neural network architecture (e.g., the number of hidden layers and the number of neurons in each hidden layer) is problem dependent. Conducting experiments is an approach to determine the optimal neural network architecture. Specifically, based on the measurement of training time and learning performance for different neural network architectures, researchers can choose the optimal neural network architecture which has the best tradeoff between training time and learning performance.

C. QoS/QoE Prediction

QoS parameters such as loss rate, delay, jitter and throughput, are network-oriented metrics, which are usually used by network operators to assess network performance. On the other hand, with the popularization and widespread of multimedia technologies, user perception and satisfaction are becoming more and more important to both network operators and service providers. The notion of QoE has emerged as user-oriented metrics to assess the user satisfaction of a service. Based on QoS/QoE prediction, network operators and service providers can offer high-quality services to increase customer satisfaction and prevent customer churn. SDN is a centralized architecture and can collect statistics from the switches at per port and per flow granularity levels, based on which ML algorithms can be applied to perform the QoS/QoE prediction. In the following, the related research on ML-based QoS/QoE prediction will be summarized.

1) QoS Prediction: QoS parameters (e.g., loss rate, delay, jitter and throughput) are related to network Key Performance Indicators (KPIs) such as packet size, transmission rate and queue length, etc. Discovering the quantitative correlations between KPIs and QoS parameters can improve the QoS management by predicting QoS parameters according to KPIs. As QoS parameters are generally continuous data, the QoS prediction problem can be considered as a regression task. Thus, supervised learning is an effective approach.

Ref. [163] studies the network delay estimation. This paper aims to train a model automatically to estimate the network delay given the traffic load and the overlay routing policy. Two different models (i.e., traditional M/M/1 network model and neural network model) are proposed to perform delay estimation. The experimental results show that the NN-based estimator has better performance than M/M/1 model in the accuracy of delay estimation. However, as it is hard for humans to interpret the trained NN model, compared with NN model, M/M/1 model is easier for humans to understand..

In [164], a two-phase analysis mechanism is proposed in SDN to improve the QoS prediction. Firstly, decision tree is used to discover correlations between KPIs and QoS parameters. Then, a linear regression ML algorithm (i.e., M5Rules) is applied to perform root cause analysis and discover each KPI's quantitative impact. The proposed mechanism can predict traffic congestion and provide recommendations on QoS improvement.

Ref. [165] focuses on application-aware QoS estimation. The authors use two learning methods (i.e., random forest and regression tree) to estimate two QoS metrics of Video-on-Demand (VoD) application (i.e., frame rate and response time) according to the collected operating system-granularity statistics, port-granularity statistics and flow-granularity statistics. Additionally, in order to reduce the computational cost, a forward-stepwise-selection technique is applied to reduce the feature set size while maintaining a low level of QoS estimation error. The simulation results demonstrate that the application-aware QoS estimation accuracy is over 90%.

2) QoE Prediction: QoE is a subjective metric to quantify the user satisfaction of a service. A widely-used QoE metric is Mean Opinion Score (MOS) [166], [167]. MOS divides the QoE values into five levels, including excellent, good, fair, poor and bad. The QoE values are usually obtained using subjective methods where a number of users are invited to rate the quality of a service. The subjective methods are time consuming. As the QoE values heavily rely on network QoS parameters (e.g., loss rate, delay, jitter and throughput), to obtain the QoE values in real time, understanding how QoS parameters affect the QoE values is very important. Machine learning is an effective method to learn the relationship between QoS parameters and the QoE values. As the QoE values are generally discrete data, the QoE prediction problem can be considered as a classification task. Thus, the best way to do QoE prediction is supervised learning.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE IV
ML-BASED ROUTING OPTIMIZATION SOLUTIONS IN SDN.

Ref.	Objective	Learning model	Complexity	Brief summary
[150]	Routing optimization	Neural network	Fair	Using ML-based meta-layer to obtain real-time heuristic-like traffic routing results
[151]	Routing optimization	Neural network	Fair	A ML-based dynamic routing framework called NeuRoute
[152]	Routing optimization	RL	Fair	Using RL to select the optimal forwarding paths to minimize network cost considering delay, loss rate and bandwidth
[155]	Routing optimization	RL	Fair	Using RL to select the optimal forwarding paths to maximize QoS-aware reward
[156]	Routing optimization	DRL	High	Using DRL to select the optimal forwarding paths to minimize network delay
[158]	Traffic prediction	Neural network	Fair	Using NN model to predict traffic load which is used to calculate the optimal routing decisions in advance
[159]	Path load prediction	Neural network	Fair	Using NN model to predict the load of each path
[160]	Traffic matrix prediction	LSTM	High	Using LSTM to predict network traffic matrix
[162]	Traffic clustering	k-means	Low	A ML-based data privacy preservation routing protocol in SDN

Ref. [166] focuses on the QoE prediction for video streaming services in SDN. Network parameters (e.g., RTT, jitter, bandwidth and delay) are used to estimate the MOS value. According to the estimation result, the SDN controller can adjust video parameters (e.g., resolution, frame per second and bitrate) to improve the QoE.

The authors of [167] use four ML algorithms (i.e., DT, neural network, k-NN and random forest) to predict the QoE values based on video quality parameters (SSIM and VQM). Two metrics, Pearson correlation coefficient and Root-MeanSquare-Error, are applied to assess the performance of these algorithms.

3) Analysis: From the related studies on QoS/QoE prediction, we can give the following analysis and summary.

- QoS prediction aims to discover the quantitative correlations between KPIs and QoS parameters, while QoE prediction aims to discover the quantitative correlations between QoS parameters and QoE values. In SDN, the controller can use the prediction results to configure devices in the data plane flexibly to improve the QoS/QoE.
- QoS prediction is generally considered as a regression task, while QoE prediction is considered as a classification task. Thus, supervised learning techniques can be used for QoS/QoE prediction. However, it is difficult to collect a large labeled training dataset due to the cost and time consumption of obtaining subjective QoE values. Semi-supervised learning algorithms only need a small part of labeled data, so they are also effective for QoE prediction.
- The performance of supervised learning algorithms depends on training datasets. Thus, in Table V, we provide a detailed comparison of the related works discussed above, from the perspectives of objective, learning model, training dataset input and output. In Table VIII, we summarize the performance of the ML-based QoS/QoE prediction solutions, from the perspectives of complexity, lower bound, upper bound, and average prediction accuracy. Ref. [165] compares the performance of random forest and regression tree. Because random forest is an ensemble method that considers the results of many decision trees, the prediction accuracy of random forest is higher than regression tree. However, the complexity of regression tree is lower than random forest.

D. Resource Management

Efficient network resource management is the primary requirement of network operators to improve network performance. SDN separates the control plane from the data plane, making the network programmable via a centralized controller with a global network view. SDN facilitates network resource management to maximize the utilization of network resources. In the following, we will review recent studies on resource management in SDN.

1) Data Plane Resource Management: There are three types of resources in the data plane, including networking, caching and computing resources. Networking resources, such as spectrum, bandwidth and power, are used to deliver data through networks to meet QoS/QoE requirements. Caching resources are used to store the frequently requested data at devices in the data plane. This way not only reduces the data transmission delay, but also decreases duplicate data transmission. With the development of new applications such as augmented reality and face recognition, more computational capability is required to run these applications normally. Due to the limited computing resources and battery capacity, users' devices tend to fail in handling all computing tasks. To offload computing tasks, computing resources have been deployed closer to end users using Edge Computing technologies [168], [169]. In the data plane, networking, caching and computing resources should be managed efficiently.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE V
ML-BASED QOS/QOE PREDICTION SOLUTIONS IN SDN.

Ref.	Objective	Learning model	Dataset input	Dataset output
[163]	Delay prediction	Neural network	Traffic load and routing policy	Network delay
[164]	QoS prediction	Decision tree	24 network KPIs	QoS parameters (e.g., loss rate, delay, jitter and throughput)
[165]	Application-aware QoS prediction	Random forest, regression tree	Operating system-granularity statistics, port-granularity statistics and flow-granularity statistics	Frame rate, response time
[166]	QoE prediction	ML algorithm	Network parameters (e.g., RTT, jitter, bandwidth and delay)	MOS value
[167]	QoE prediction	Decision tree, neural network, k-NN, random forest	Video quality parameters (SSIM and VQM)	MOS value

There are two scenarios in SDN: single-tenancy and multi-tenancy SDN network. In the first scenario, a logically centralized controller belonging to the single tenant can control all resources in the data plane, while in the second scenario, resources in the data plane are shared by multiple tenants. In multi-tenancy SDN network, tenants use their own SDN controllers to control their isolated resources in the data plane.

a) Resource Allocation in Single-tenancy SDN Network: Many works have studied the data plane resource allocation in single-tenancy SDN network.

In [170], an integrated framework is proposed to enhance the performance of sofware-defined virtualized Vehicular Ad-hoc Network (VANET), by allocating the networking, caching and computing resources dynamically. The resource allocation problem is formulated as a joint optimization problem, considering the gains of networking, caching and computing. A novel DRL algorithm is proposed to solve the complex optimization problem and obtain the resource allocation policy. Similar to [170], [171] focuses on the joint resource allocation optimization issue in smart cities.

In [172], the authors propose a novel Roadside Unit (RSU) cloud architecture, shown in Fig. 15. The RSU cloud is able to instantiate, replicate, and migrate services dynamically. In order to reduce reconfiguration cost, a RL-based heuristic approach is designed to select configurations that minimize the long-term reconfiguration cost in the network.

Content distribution is one of the most popular services in people's life. Content distribution optimization can improve user satisfaction. Many works have studied the content distribution optimization in SDN.

Ref. [173] presents a RL-based context-aware content distribution scheme to improve the content distribution QoS. Based on network status, such as content distribution time, network throughput and network overhead, the proposed context-aware content distribution scheme can select the optimal protocol (i.e., TCP/IP or Content Centric Networking (CCN)) to deliver different services.

In [174], an intelligent streaming architecture called SDNHAS is proposed in SDN to help the HTTP Adaptive Streaming (HAS) players decide the optimal bitrate and quality when streaming media content over the Internet, by grouping the HAS players into a small number of clusters and leveraging RL algorithm to make the bitrate and quality decisions in each cluster. Experimental results demonstrate that compared with six well-known bitrate adaptation schemes (e.g., BBA [175] and PANDA [176]), SDNHAS increases QoE fairness over 33%, video stability over 32% and network resource utilization over 29% on average.

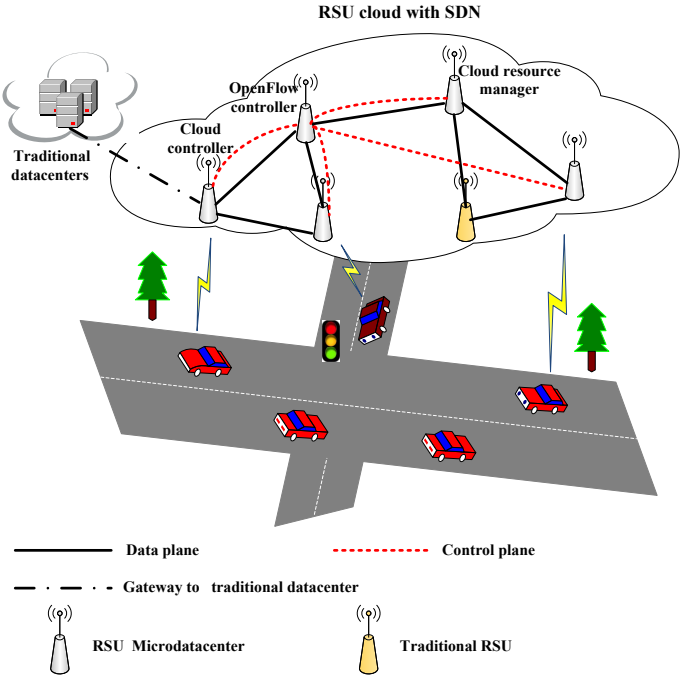


Fig. 15. RSU cloud architecture [172]. RSU cloud is composed of traditional RSUs and RSU microdatacenters. Traditional RSUs as fixed roadside infrastructures can perform Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communications. RSU microdatacenters are the fundamental components of the RSU clouds, which host services to satisfy the demands of mobile vehicles. The difference between traditional RSUs and RSU microdatacenters is that additional hardware and software components are deployed in RSU microdatacenters to provide virtualization and communication capabilities.

Ref. [177] focuses on the deployment of cache-enabled Unmanned Aerial Vehicles (UAVs) in a Cloud Radio Access

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

Network (C-RAN) to optimize users' QoE and UAVs' transmit power. ESN, a type of recurrent NNs, is used to predict the content request distribution and mobility pattern of each user. Based on the prediction results, the optimal solutions related to the user-UAV association, each UAV's location and the content to cache at each UAV can be derived.

b) Resource Allocation in Multi-tenancy SDN Network: In multi-tenancy SDN network, resources in the data plane are shared by multiple tenants. How to allocate data plane resources among multiple tenants to maximize their utility is an important research issue.

Ref. [131] studies the RRH assignment issue among Mobile Network Operators (MNOs) in C-RAN. The RRH assignment problem is formulated as a non-cooperative game-theoretic problem, where the players are RRHs, the action is MNO selection, and the utility of each player is to maximize its Received Signal Strength (RSS) level. In order to solve the problem with low computational complexity, regret-matching-based learning algorithm is used by each player to select the optimal set of MNOs.

Ref. [132] studies the computation offloading issue in MEC. The computation offloading problem is formulated as a non-cooperative game-theoretic problem, where the players are MEC servers, the two actions of each player are active and inactive, and the utility of each player is to minimize its energy consumption. In the proposed game-theoretic model, the RL algorithm is applied by each player to learn the optimal action (i.e., active or inactive).

In [133], the spectrum sharing problem in LTE and WiFi coexistence system is studied. In the coexistence system, LTE and WiFi belong to different operators and have their own controllers, while LTE and WiFi share the same unlicensed spectrum resources. The spectrum sharing problem is formulated as a non-cooperative game-theoretic problem, where the players are the two controllers, the action of each player is the amount of shared spectrum resources, and the utility of each player is to maximize the spectrum resource utilization. At the beginning of each play round, controllers use decision tree algorithm to predict the opponent's network status. Then, based on the prediction results, controllers make their optimal spectrum resource sharing decisions.

Ref. [134] studies the network function assignment issue in SDN/NFV system. The SDN/NFV system is considered as a network function market, where servers and users are the sellers and buyers of network functions respectively. The network function assignment problem is formulated as a two-stage Stackelberg game, where servers act as leaders and users act as followers. Servers and users try to make their optimal decisions to maximize their utility and benefit. The existence of Stackelberg equilibrium has been proved, and the RL algorithm is applied to converge to the equilibrium.

c) Admission Control (AC): The rapid growth of various devices in the data plane such as mobile phones and IoT devices increases the service requests significantly. However, resources in the data plane are limited. Thus, admission control in SDN is very important. The goal of admission control is to manage a large number of service requests by accepting or rejecting new incoming requests according to the resource availability.

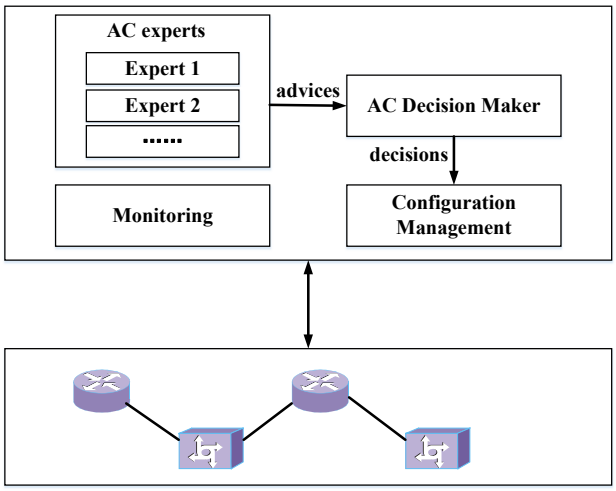


Fig. 16. A ML-based AC method in the SDN controller [178], [179]. Each expert is an online AC algorithm, such as Greedy, Agrawal [180] and AAP-pd [181], [182]. AC Decision Maker selects the best decision from all experts according to varying traffic conditions.

Ref. [178], [179] focus on the admission control in SDN. A ML-based method in the SDN controller is proposed to select the most proper AC algorithm from a pool of online algorithms (experts) according to varying traffic conditions. The ML-based method is shown in Fig. 16.

2) Control Plane Resource Management: In this part, we will review recent studies on control plane resource management in SDN from resource allocation and controller placement.

a) Control Plane Resource Allocation: The advances of network virtualization technologies facilitate the development of multi-tenancy SDN network to share resources in the data plane among multiple tenants. In multi-tenancy SDN network, network hypervisors are introduced between the data plane and the control plane. Network hypervisors allow tenants to use their own controllers to control their isolated resources in the data plane. FlowVisor [183] and OpenVirteX [184] are two widely-used network hypervisors. Network hypervisors have to process all tenants' control traffic between the data plane and the control plane, but they generally have limited computing resources (e.g., CPU). Thus, how to allocate the limited resources of network hypervisors among multiple tenants to guarantee the communication between each tenant's data plane and control plane is an important research issue. In this scenario, ML algorithms generally run on the network hypervisors to optimize resource allocation.

In [185], a resource monitor tool is used to monitor the CPU consumption of network hypervisors, and the benchmarking tool hvbench is used to measure the control message rate. The collected data is used to train three different regression learning models to learn the mapping between the CPU consumption and the control message rate. The trained mapping models can be used to estimate whether network hypervisors are overloaded in real time according to the measured control message rate. The overload of network hypervisors has a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

significant impact on the processing of each tenant's control messages.

Ref. [186] goes a step further and extends the approach of [185] to scenarios where the available computing resources of hypervisors are fluctuant. A big change of the available computing resources of hypervisors will make the current mapping model invalid. To detect the resource change, the CPU consumption information and the control message rate information are collected continuously. Then, SVM is used to compare the collected data and the current mapping model. If most of the collected data in a time period do not fit the current model, a big resource change has occurred. In this case, the mapping model will be re-trained using the recently collected data.

b) Controller Placement: In SDN, the centralized controller has to process traffic flows from a large number of switches deployed in different locations. The long distance between the controller and switches will increase the traffic flow processing delay. In this case, the controller's location has a significant impact on the network performance. Thus, the controller placement problem should be studied seriously. Heuristic algorithms are an approach to solve this problem. The high computational complexity is the main shortcoming of heuristic algorithms. Ref. [187], [188] have shown that supervised learning algorithms are effective to solve the controller placement problem.

In [187], [188], three supervised learning algorithms (i.e., decision tree, neural network and logistic regression) are applied to obtain the optimal controller's locations. The input of training dataset used to train supervised learning algorithms is traffic distribution, and the output is the corresponding controller placement solutions of heuristic algorithms. After training, the trained model can be applied to predict the controller placement solutions in real time. On one hand, the predicted solutions are used directly as the optimal solutions. On the other hand, the predicted solutions can also be used as initial solutions of heuristic algorithms to reduce the algorithm runtime.

3) Others: Many works also have been done to solve other issues related to resource management in SDN.

In [189], different ML approaches are used in a WiFi-Direct network to estimate the number of active UEs, by leveraging only the network information available at the receiving nodes. Based on the prediction results of [189], the work in [190] estimates the remaining transmission time (i.e., the Estimated Time of Arrival (ETA)) during the file transmission in SDN-based wireless networks. The accurate ETA prediction can help the clients to schedule their application-layer actions.

Ref. [191] studies the autonomic network management in the software defined 5G systems. Autonomic Manager (AM) is a key part to provide the network intelligence in the novel management framework which is proposed by the SELFNET project. Diagnoser, Decision-Maker (DM) and Action Enforcer (AE) are the three main functional blocks in AM. An intelligent control loop, shown in Fig. 17, is designed to provide the network intelligence in the SELFNET framework.

Ref. [192] studies the Service Level Agreements (SLA) management in SDN. Firstly, LSTM is used to calculate

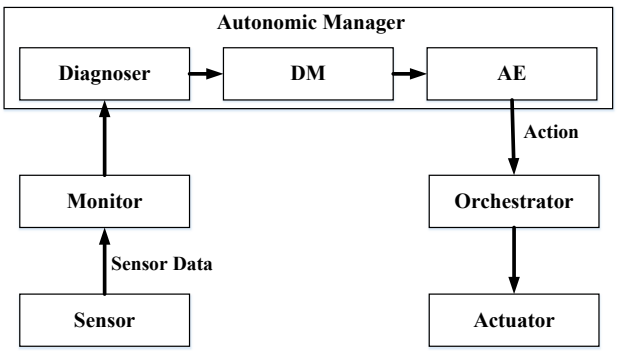


Fig. 17. The intelligent control loop in SELFNET framework [191]. The sensors in the network generate sensor data. The monitor analyzes and aggregates the sensor data to detect network problems. The diagnoser is capable of diagnosing the root cause of network problems reported by the monitor. Based on reasons of these problems, a set of corrective and preventive methods are decided by the Decision-Maker (DM). Action Enforcer (AE) is in charge of providing implementable actions to be enforced in the network infrastructure. The orchestrator and actuator are responsible for executing the implementable actions.

the next values of the system features at time $t+1$ given the numerical values of these features at time t. Secondly, the forecasted values of features are taken by decision tree algorithm to determine the most likely SLA Violation (SLAV) to occur. Then proactive management actions are taken to avoid SLAV before they occur.

In [193], a statistical machine learning approach is proposed to increase the robustness of SDN-based Wireless Sensor Network (WSN) by predicting interference patterns over time. According to the interference prediction, the SDN controller can modify network state to keep the system stable and to prevent network malfunctions.

4) Analysis: In Table VI, we provide a comparison of the related works discussed above. From the related studies on resource management, we can give the following analysis and summary.

- The data plane resource allocation problem is generally considered as a decision-making task. In this case, RL and ML-based game theory are two effective approaches. RL is often used in the single-tenancy SDN network. In multi-tenancy SDN network, resources in the data plane are shared by multiple tenants. Each tenant's utility can be affected by the resource allocation strategies of the other tenants. Thus, ML-based game theory is suitable to solve the data plane resource allocation problem in multi-tenancy SDN network.
- Control plane resource allocation is generally considered by network hypervisors to allocate their limited resources to multiple tenants. The mapping between the resource consumption and control message rate is very important for control plane resource allocation. The mapping problem is generally considered as a regression task. Thus, supervised learning techniques can be used for the mapping task. In the dynamic scenarios where the available resources of hypervisors are fluctuant, the trained mapping model needs to be updated periodically.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE VI
ML-BASED RESOURCE MANAGEMENT SOLUTIONS IN SDN.

Ref.	Objective	Learning model	Complexity	Brief summary
[170]	Data plane resource allocation	DRL	High	Using DRL to jointly allocate networking, caching and computing resources in sofware-defined vehicular networks to maximize network operators' comprehensive revenue
[171]	Data plane resource allocation	DRL	High	Using DRL to jointly allocate networking, caching and computing resources in smart cities to maximize network operators' comprehensive revenue
[172]	Service reconfiguration	RL	Fair	Using RL to select the optimal configuration strategy to minimize the long-term reconfiguration cost in VANET
[173]	Content delivery optimization	RL	Fair	Using RL to select the optimal content delivery protocol (i.e., TCP/IP or CCN) to maximize the content distribution QoS
[174]	HAS traffic delivery optimization	RL	Fair	Using RL to select the optimal bitrate and quality to maximize users' QoE when delivering HAS media content
[177]	Cache-enabled UAVs deployment optimization	Recurrent NN	High	Using ESN, a type of recurrent NNs, to predict the content request distribution and mobility pattern of each user
[131]	RRH allocation	Regret-matching-based game theory	Fair	Utilizing regret-matching-based game theory for RRH allocation among MNOs to maximize RSS levels
[132]	Computation offloading in MEC	RL-based game theory	Fair	Using RL-based game theory method to solve the energy-efficient MEC server activation problem
[133]	Spectrum sharing	DT-based game theory	Fair	Using decision tree to predict the opponent's network status, based on which the controllers make their optimal decisions to maximize the spectrum resource utilization
[134]	Network function assignment	RL-based game theory	Fair	Using RL-based Stackelberg game theory to optimize the network function assignment between servers and users
[185]	Hypervisor resource utilization estimation	Regression learning model	Low	Using regression learning models to learn the mapping between CPU consumption and control message rate
[186]	Hypervisor resource change detection	SVM	Fair	Using SVM to calculate whether the recently collected data fits the current mapping model
[187]	Controller placement	Decision tree, neural network, logistic regression	Fair	Using supervised learning algorithms to obtain the controller placement solutions in real time
[189]	Estimating the number of active nodes	Naive Bayes, SVM, k-NN	Fair	Different ML approaches to predict the number of active nodes in a WiFi-Direct network
[192]	SLA management	LSTM, decision tree	High	Using LSTM to predict the values of system parameters, and using decision tree to determine the most likely SLA violation
[193]	Increasing robustness of WSN	ML approaches	Fair	Using ML techniques to predict interference patterns over time to increase the robustness of SDN-based WSN

E. Security

Security is always an important aspect that needs to be considered by network operators. Only secure networks can be accepted and used by users. Intrusion detection is an important element for network security. An Intrusion Detection System (IDS) is a device or software application and its objective is to monitor the events in a network system and identify possible attacks [194]. IDS is helpful for network operators to take appropriate actions before an attack occurs.

Generally, there are two types of IDSs according to how they identify network attacks: signature-based IDS and anomaly-based IDS [195]–[198]. In signature-based IDS, humans are responsible for creating the signatures of known attacks. When traffic flows arriving, signature-based IDS compares these traffic flows against the known signatures to identify possible malicious activities. The signature-based IDS generally has high accuracy. However, it has some shortcomings. First, signature-based IDS can only recognize attacks whose signatures are available. The growth of new attacks makes the signature update difficult. Second, signature-based IDS incurs high time consumption as all signatures need to be compared.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

In contrast, anomaly-based IDS is a statistical method, which uses the collected data related to the behavior of legitimate users to create a model. When traffic flows arriving, the traffic flows are compared with the model. The behavior which has a significant deviation from the model will be marked as an anomaly. An advantage of anomaly-based IDS is that it can detect new types of attacks. Note that signature-based IDS is the type of payload-based traffic identification which needs to inspect the whole payload of packets, while anomaly-based IDS is the type of flow-based traffic identification based on flow-granularity information such as packet header information. In this paper, we focus on the anomaly-based IDS.

Machine learning methods are widely used in anomaly-based IDS by training a model to identify normal activities and intrusions. The intrusion detection problem can be considered as a classification task. Thus, supervised learning algorithms are often applied for intrusion detection.

In ML-based intrusion detection systems, the high dimensionality of dataset input (i.e., flow features) has impact on the performance of ML algorithms. To speed up the process of intrusion detection while maintaining high detection accuracy, feature reduction is often done to reduce the dimensionality of dataset input. Feature selection and feature extraction are two well-known methods to reduce the dimensionality of flow features. Feature selection is a method to choose a subset of appropriate features from all flow features. Feature extraction is another way to reduce the dimensionality of flow features through feature transformation by extracting a set of new features from the original features.

The capabilities of SDN (e.g., logically centralized control, global view of the network, software-based traffic analysis, and dynamic updating of forwarding rules) facilitate the ML-based intrusion detection and enhance the network security [199]. First, the global network view of the SDN controller simplifies the collection and analysis of network traffic. Moreover, the programmability of SDN makes it easy to react to network attacks immediately when they are detected. Many studies have been done for ML-based intrusion detection in SDN, such as coarse-grained intrusion detection, fine-grained intrusion detection and DDoS attack detection. In the following, the related studies will be summarized.

1) Coarse-grained Intrusion Detection: Coarse-grained intrusion detection aims to classify traffic flows as normal and abnormal classes.

Ref. [200] proposes a threat-aware system to perform detection and make response to network intrusion in SDN, which is composed of data preprocessing, predictive data modeling, and decision making and response subsystem. First, a forward feature selection strategy is used by the data preprocessing subsystem to select appropriate feature sets. Then, decision tree and random forest algorithms are applied by the predictive data modeling subsystem to detect malicious activities. Based on the intrusion detection results, the decision making and response subsystem uses reactive routing to install different flow rules for different flow types. Comprehensive experiments have been done and the results show that by using the forward feature selection strategy, the proposed threat-aware system can reduce the traffic processing time while maintaining high intrusion detection accuracy.

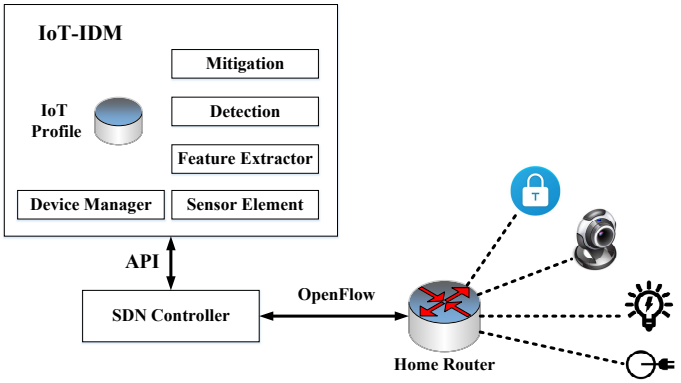


Fig. 18. A typical smart home network architecture with IoT-IDM [203]. The IoT-IDM is composed of five key modules: Device Manager, Sensor Element, Feature Extractor, Detection, and Mitigation. Device Manager is a database, known as IoT Profile, to store security related information of IoT smart devices. Sensor Element is responsible for logging network activities on a target smart device. Feature Extractor can extract features from the captured network traffic. Detection is in charge of identifying suspicious activities. The Mitigation module is able to take appropriate actions before the identified attacks occur.

Ref. [201] proposes an HMM-based Network Intrusion Detection System (NIDS) to predict malicious activities and enhance network security. Five selected flow features (i.e., the length of the packet, source port, destination port, source IP address and destination IP address) are used by HMM to determine the maliciousness of a set of packets.

Ref. [202] proposes a framework called ATLANTIC to perform anomaly traffic detection, classification and mitigation jointly in SDN. The ATLANTIC framework performs anomaly detection and classification in two phases: a lightweight phase and a heavyweight phase. The lightweight phase uses information theory to calculate deviations in the entropy of flow tables. The heavyweight phase utilizes SVM algorithm to classify the abnormal traffic. Then, ATLANTIC takes appropriate mitigation actions to process malicious flows automatically and analyze unknown traffic flows manually by a human administrator.

In [203], an intrusion detection and mitigation architecture, called IoT-IDM, is proposed in smart home environment to protect smart devices, by leveraging machine learning techniques to detect malicious activities. The IoT-IDM, shown in Fig. 18, is composed of five key modules: Device Manager, Sensor Element, Feature Extractor, Detection, and Mitigation. An IoT-IDM prototype is implemented as a module of Floodlight and the applicability and efficiency of IoT-IDM architecture have been demonstrated.

Ref. [204] leverages four ML algorithms (i.e., decision tree, BayesNet, decision table and Naive Bayes) to predict the potential malicious connections and vulnerable hosts. The prediction results are used by the SDN controller to define security rules in order to protect the potential vulnerable hosts and restrict the access of potential attackers by blocking the entire subnet. The performance of these ML algorithms is compared. The results indicate that BayesNet has better

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

performance than the other three algorithms, and the average prediction accuracy achieved by BayesNet is 91.68%.

In [205], deep NN model is used in SDN to detect intrusion activities, by classifying traffic flows into normal and anomaly classes. The deep NN model with an input layer, three hidden layers and an output layer is trained based on the NSL-KDD dataset [206]. When the SDN controller detects a network anomaly, the OpenFlow protocol is used to modify switches' flow tables in order to prevent attacks. The experiment results show that the deep NN model has a good performance in anomaly detection, and the average detection accuracy is 75.75% when using only six basic flow features.

Ref. [207] uses deep recurrent NN in an anomaly-based IDS and proposes a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) algorithm to detect intrusion. To speed up the intrusion detection and reduce the computational cost, only six flow features such as duration and protocol type are used to train the GRU-RNN algorithm.

2) Fine-grained Intrusion Detection: Fine-grained intrusion detection aims to give a fine-grained classification of network traffic and identify different types of attacks.

The authors of [208] propose an improved behaviour-based SVM to categorize network attacks. To increase the intrusion detection accuracy and speed up the learning of normal and intrusive patterns, decision tree is used as a feature reduction approach to outrank raw features and select the most qualified features. These selected features are the input data to train the SVM classifier.

In [209], a novel deep learning-based intrusion detection method called NDAE is proposed. To speed up the intrusion detection while maintaining high detection accuracy, NDAE combines the deep learning approach and random forest, where deep learning approach is applied for feature reduction, and random forest is used for traffic classification and intrusion detection.

3) DDoS Attack Detection: DDoS attack is a major threat to cyber security in SDN. The goal of a DDoS attack is to exhaust system resources by simultaneously sending a large number of fake requests using many puppet machines so that legitimate users' requests are not processed. In SDN, the DDoS attack can exhaust the networking, storage and computing resources in the data plane and the control plane, which will make the SDN network unavailable. Thus, DDoS attack detection is important for the normal running of SDN networks.

In [210], a lightweight DDoS attack detection method is proposed and implemented on a NOX-based SDN network. The NOX controller collects traffic flow feature information from OpenFlow switches. Then SOM [118], [119] is used to perform DDoS attack detection according to the collected traffic flow features.

Ref. [211] implements a new IDS in the SDN controller to detect DDoS attacks. The proposed IDS consists of two modules: Signature IDS and Advanced IDS. Different ML algorithms, such as k-NN, Naive Bayes, k-means and k-medoids, are utilized by the Signature IDS module to classify traffic flows as normal and abnormal and find a set of hosts with anomalous behaviors. Then, the packets sent by these hosts with anomalous behaviors will be checked by the Advanced IDS module to detect whether the hosts are anomalous or the authorized users. In this way, the processing time of the Advanced IDS module is reduced because only hosts with anomalous behaviors need to be analyzed.

In [98], a deep learning model is used to detect DDoS attacks in SDN. Recurrent NN and convolutional NN are included in the deep learning model. The deep learning model consists of an input layer, a forward recursive layer, a reverse recursive layer, a fully connected hidden layer and an output layer. After the collection and analysis of network traffic feature information, the deep learning model is used for feature reduction and DDoS attack detection.

In [212], an SDN-based DDoS detection system is proposed. The system first extracts 68 flow features from the collected network traffic, including 34 features from TCP flows, 20 features from UDP flows, and 14 features from ICMP flows. Then, a deep learning model is applied for feature reduction and DDoS attack detection.

4) Others: There are other works related to the SDN security.

Ref. [213] studies the application software fault identification in SDN. The application faults have significant impact on the SDN network and other applications in the application plane. Two proof-of-concept examples of application faults are presented. ML approaches are utilized to detect these application faults. The detection results can guide the SDN controller to take appropriate network response in real time.

The authors of [214] focus on software defined firewall and propose a framework to match flows quickly and capture user behavior efficiently. HMM is applied to capture state information of user behaviors and identify whether a network connection is legitimate or not. If an illegitimate connection is found, the firewall can block access of that connection in time. Then, these information along with the corresponding packet's fields are utilized to train a neural network model. The trained model is able to match flows quickly instead of comparing a packet flow against each firewall filtering rule.

In [215], machine learning approaches are applied in the SDN-based High Throughput Satellite (HTS) systems (shown as Fig. 19) to predict feeder-link outage. Linear regression, neural network and Bayes methods are used to predict SINR in the future. If the SINR is below a given threshold, link outage may happen. In the circumstance, the SDN controller should reroute the network traffic and orchestrate the gateway handover operations.

5) Analysis: From these related works on network security, we can give the following analysis and summary.

- Fine-grained intrusion detection is often applied for fine-grained network management. By identifying different types of attacks, the SDN controller can make appropriate reactions for each type of network attacks. However, compared with coarse-grained intrusion detection, fine-grained intrusion detection requires a more complex labeled training dataset.
- KDD99 [216] and NSL-KDD [206] are two widely-used datasets for IDS research. NSL-KDD is a modified version of KDD99 dataset and has solved many inherent problems of KDD99 dataset. Thus, when researchers

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

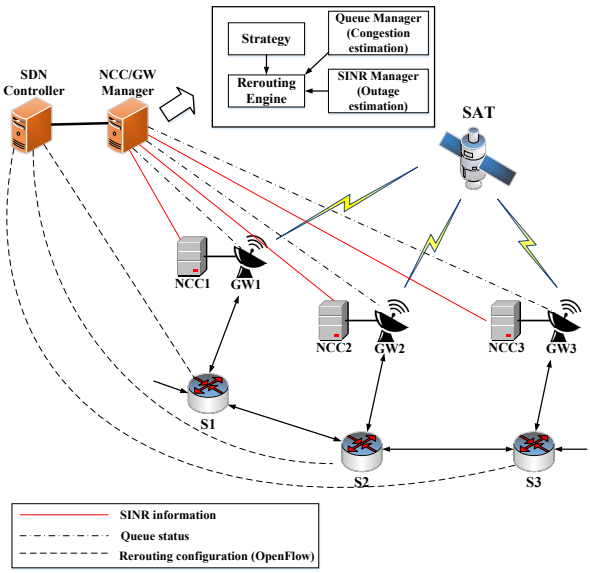


Fig. 19. An example of SDN-based HTS system [215]. NCC stands for Network Control Centre, GW for gateway. The NCC/GW Manager collects the SINR information from NCCs and Queue status from GWs. These information is used by machine learning techniques to estimate the outage and congestion. Rerouting Engine decides policies based on the estimation results to modify network state through the SDN controller.

want to simulate the performance of ML-based intrusion detection methods, it is better to use the NSL-KDD dataset.

- To enhance network security, real-time intrusion detection is required. Feature reduction is an effective way to optimize system performance and speed up attack detection. As mentioned above, feature selection and feature extraction are two well-known methods to reduce flow features. Most of the related works select flow features according to researchers' experience. Decision tree is another feature selection method by analyzing raw features and choosing the most qualified features [208]. Deep learning is a widely-used feature extraction method due to its strong feature representation capability [217].
- The performance of supervised learning algorithms depends on training datasets. Thus, in Table VII, we provide a detailed comparison of the related works discussed above, from the perspectives of objective, feature reduction method, intrusion detection method, training dataset, training dataset input and output. In Table VIII, we summarize the performance of the ML-based intrusion detection solutions, from the perspectives of complexity, lower bound, upper bound, and average detection accuracy.
- As the intrusion detection problem can be considered as a classification task, supervised learning algorithms are often applied to detect abnormal activities. Ref. [200] points out that random forest has better performance than decision tree. The performance of four machine learning algorithms (i.e., decision tree, BayesNet, decision table, and Naive Bayes) is compared and evaluated in [204]. The results indicate that BayesNet has better performance than the other three algorithms. The experiments in [211] show that Baysian algorithm has better performance than k-NN. Compared with conventional machine learning algorithms, deep learning is better for intrusion detection due to its multiple-level feature representation capability. Ref. [205] compares the deep learning algorithm with three other ML algorithms (i.e., Bayes' theory, SVM and decision tree). The experimental results show that the deep learning algorithm has higher intrusion detection accuracy. However, the complexity of the deep learning algorithm is higher than the other three ML algorithms (i.e., Bayes' theory, SVM and decision tree). The experiments in [98] indicate that increasing the volume of training datasets not only improves the performance of deep learning model significantly, but also increases the training time and complexity of deep learning model. Therefore, there is a tradeoff between detection performance and model complexity.

Finally, the ML-based solutions discussed in the section and their pros and cons are compared in Table IX and Table X.

VI. Challenges and Future Research Directions

Despite current works being done in SDN, with the requirements of robustness and maturity in the area, many significant research challenges remain to be addressed prior to widespread implementation of a fully intelligent SDN in the near future. In this section, we discuss challenges and present future research directions. Furthermore, the role of ML algorithms to address these challenges is also discussed.

A. High-quality Training Datasets

In order to improve the estimation or classification accuracy of the models trained by machine learning techniques, enough training datasets are needed [2]. What is sufficient training data in networking? Therefore, it is a research direction to study the relationship between the training dataset size, the network characteristics, and the performance of the machine learning models. On the other hand, progress in machine learning algorithms heavily depends on the availability of high-quality standardized training datasets. However, it is difficult to obtain high-quality annotated network flow samples across a broad range of applications [139]. To solve this problem, one possible approach is to public datasets, which is a common solution in several popular machine learning applications, such as image recognition [220]. In this respect, it is necessary to initiate similar activities for the publication of datasets in networking AI field.

B. Distributed Multi-controller Platform

Network scalability is a critical issue in SDN. When a single controller is deployed in the control plane, with the increasing of network size and the number of flows, the controller usually faces the scalability issue due to the computation limitation of a controller [221]. Distributed multi-controller platforms [40],

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE VII
ML-BASED INTRUSION DETECTION SOLUTIONS IN SDN.

Ref.	Objective	Feature reduction method	Intrusion detection method	Training dataset	Dataset input	Dataset output
[200]	Coarse-grained intrusion detection	A forward feature selection strategy	DT, RF	KDD99	Ten features	2 classes: normal and anomaly
[201]	Coarse-grained intrusion detection	-	HMM	Collected network traffic	Five features: the length of the packet, source port, destination port, source IP address and destination IP address	2 classes: normal and anomaly
[202]	Coarse-grained intrusion detection	-	SVM	Collected network traffic	IP address, transport port	2 classes: normal and anomaly
[203]	Coarse-grained intrusion detection	-	SVM	Collected network traffic	Three features such as inter-packet time interval	2 classes: normal and attack
[204]	Coarse-grained intrusion detection	-	DT, BayesNet, decision table, Naive Bayes	LongTail [218]	Attacker IP, attacked host, number of attempts in an attack, and timestamp	2 classes: normal and attack
[205]	Coarse-grained intrusion detection	-	Deep NN	NSL-KDD	Six features: duration, protocol type, source bytes, destination bytes, count and service count	2 classes: normal and anomaly
[207]	Coarse-grained intrusion detection	-	Recurrent NN	NSL-KDD	Six features such as duration and protocol type	2 classes: normal and anomaly
[208]	Fine-grained intrusion detection	DT	SVM	KDD99	23 features	5 classes: normal and four types of network attacks (i.e., DoS, U2R, R2L and Probe)
[209]	Fine-grained intrusion detection	Deep NN	RF	KDD99, NSL-KDD	41 features	5 classes: normal and four types of network attacks (i.e., DoS, U2R, R2L and Probe)
[210]	DDoS attack detection	-	SOM	Collected network traffic	Six features such as average of packets per flow and average of bytes per flow	2 classes: normal and DDoS
[211]	DDoS attack detection	-	Naive Bayes, k-NN, k-means, k-medoids	Collected network traffic	-	2 classes: normal and DDoS
[98]	DDoS attack detection	Deep NN	Deep NN	ISCX dataset [219]	20 features	2 classes: normal and DDoS
[212]	DDoS attack detection	Deep NN	Deep NN	Collected network traffic	68 features: 34 from TCP flows, 20 from UDP flows and 14 from ICMP flows	8 classes: normal and seven types of DDoS attacks
[213]	Application fault identification	-	ML approaches	Using ML approaches to detect application software faults in SDN		
[214]	Firewall performance optimization	-	Neural network, HMM	A ML-based software defined firewall to process packets quickly		
[215]	Feeder-link outage prediction	-	Neural network, Bayes	Developing a feeder-link outage prediction algorithm in the SDN-based HTS system		

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE VIII
PERFORMANCE OF THE ML-BASED SOLUTIONS IN SDN.

	Ref.	Learning model	Complexity	Classification accuracy		
				Lower bound	Upper bound	Average
Traffic Classification	[144]	Decision tree	Low	85%	98%	>90%
	[139]	Random forest	Fair	73.6%	96%	86.4%
	[145]	ML classifier	Low	-	-	>85%
	[146]	SVM	Fair	78%	99.9%	>90%
	[140]	Decision tree	Low	82%	100%	94%
	[147]	Deep NN	High	85%	100%	93.5%
	[148]	Decision tree	Low	85%	100%	95.5%
		k-NN	Low	70%	99.1%	>90%
	[141]	Semi-supervised learning	Fair	81%	92%	>90%
Routing Optimization	[150]	Neural network	Fair	83.2%	88.2%	84.81%
	[151]	Neural network	Fair	95.5%	97.85%	97%
	[159]	Neural network	Fair	87%	96%	93%
QoS/QoE Prediction	[163]	Neural network	Fair	-	-	Over 98%
	[164]	Decision tree	Low	-	-	Not mentioned
	[165]	Random forest	Fair	71%	98%	Over 90%
		Regression tree	Low	68%	98%	Over 90%
	[166]	ML algorithm	Low	-	-	Not mentioned
	[167]	Decision tree, neural network, k-NN, random forest	Fair	-	-	The Pearson correlation coefficient of four ML algorithms are 0.79, 0.68, 0.75 and 0.8 respectively
Resource Management	[189]	Naive Bayes	Low	85.1%	99.3%	92%
		Linear SVM	Fair	86.1%	99.8%	92.8%
		Radial SVM	Fair	87.1%	99.1%	93.1%
		k-NN	Low	87.2%	99.6%	93.4%
	[192]	Decision tree	Low	82.2%	90.9%	87.4%
Security	[200]	Decision tree	Low	45.02%	91.18%	82.48%
		Random forest	Fair	45.02%	99.41%	98.75%
	[201]	HMM	Fair	-	-	88%
	[202]	SVM	Fair	-	-	88.7%
	[203]	SVM	Fair	-	-	96.2%
	[204]	Decision tree	Low	78.62%	91.4%	86.19%
		BayesNet	Low	73.56%	99.89%	91.68%
		Decision table	Low	70.36%	99.99%	88.52%
		Naive Bayes	Low	67.7%	99.52%	87.78%
	[205]	Deep NN	High	72.05%	91.7%	Deep NN: 75.75%, decision tree: 74%, SVM: 70.9%, Bayes' theory: 45%
	[207]	Recurrent NN	High	-	-	RNN: 89%, SVM: 65.67%
	[208]	DT + SVM	Fair	97.52%	99.79%	97.55%
	[209]	Deep NN + RF	High	-	97.73%/99.79%	NSL-KDD dataset: 85.42%, KDD99 dataset: 97.85%
	[210]	SOM	Low	-	-	98.61%
	[211]	Naive Bayes, k-NN, k-means, k-medoids	Low	-	-	Naive Bayes: 94%, k-NN: 90%, k-means: 86%, k-medoids: 88%
	[98]	Deep NN	High	94.39%	99.79%	97.56%
	[212]	Deep NN	High	-	-	95.65%
	[213]	ML approaches	Low	-	-	-
	[214]	Neural network, HMM	Fair	-	-	-
	[215]	Neural network, Bayes	Fair	-	-	-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE IX
ADVANTAGES AND SHORTCOMINGS OF THE ML-BASED SOLUTIONS.

	Ref.	Advantages	Shortcomings
Traffic Classification	[144]	Both accuracy and cost-sensitive are considered when detecting elephant flows.	The classification of elephant flows with different QoS requirements has not been considered.
	[139]	A single OpenFlow switch is deployed in an enterprise network to collect data.	Only TCP traffic is taken into account.
	[145]	MultiClassifier is proposed for application classification.	Calculation consumption of the SDN controller is high and the scheme cannot scale to a large network.
	[146]	SVM algorithm is used to classify UDP traffic according to Netflow records.	Only UDP traffic is taken into account.
	[140]	A crowd sourcing approach is used to collect ground truth data from end devices.	Only 40 Android applications are tested.
	[147]	Deep NN is used to identify mobile applications.	Flow features are selected manually to train the deep NN model.
	[148]	Both application name and flow types are identified.	Users' mobile devices need run an "agent" software to communicate with the SDN controller.
	[141]	Network traffic is classified into different QoS classes.	Applications with time-dynamic QoS requirements are not considered.
Routing Optimization	[150]	The heuristic algorithm's input and its corresponding output are the training dataset of the ML-based meta-layer.	The dynamic network topology is not considered.
	[151]	Neural network model is applied to obtain real-time heuristic-like routing solutions.	Only network delay is considered while ignoring the load balance of the total network traffic.
	[152]	RL method is applied to improve the performance of routing protocols.	The proposed routing protocol is tested in a virtual SDN.
	[153]	Random NN and RL are used to find the optimal network paths.	Only one cloud tenant is considered.
	[155]	A multi-layer hierarchical SDN scenario is investigated.	Only the QoS requirements of widely-used applications are taken into account.
	[156]	DRL model is applied for routing optimization.	Only network delay is considered, while other performance like throughput is ignored.
	[158]	Neural network model is applied to predict network traffic load in advance.	The route change and link congestion are not considered.
	[159]	Neural network model is used to predict the load of each path.	The dynamic network topology is not considered.
	[162]	K-means method is used to cluster network traffic into several clusters of risk ratios in an off-line mode.	The controller's load is high.
QoS/QoE Prediction	[164]	A linear regression algorithm is applied to perform root cause analysis and discover each KPI's quantitative impact.	The comparison between different machine learning algorithms is not considered.
	[165]	Two learning methods are used to estimate application-aware QoS metrics.	The overhead of the used feature selection method is high.
	[166]	A ML-based approach and adaptive video delivery service are combined to provide a better QoE for video streaming service.	Other services such as video conference and real-time applications are not analyzed.
	[167]	Four machine learning algorithms are used to predict QoE values.	QoS parameters such as response time and transmission delay are not considered.
Resource Management	[170]	An integrated framework is proposed to improve the performance of vehicular networks by allocating the networking, caching and computing resources jointly.	Energy efficiency issue in the integrated framework is not considered.
	[171]	An integrated framework is proposed to improve the performance of applications in smart cities by allocating the networking, caching and computing resources jointly.	Energy efficiency issue in the integrated framework is not considered.
	[172]	RL is utilized to minimize the long-term reconfiguration cost.	The overall complexity of the RSU cloud system is not discussed.
	[173]	RL method is used to select the optimal protocol (i.e., TCP/IP or CCN) to deliver different services.	Calculating time of the proposed algorithm is long.
	[174]	A cluster-based mechanism is presented to increase the scalability of the SDNHAS system.	Only three features (i.e., device resolution, content type and user expectation) are considered to optimize each HAS player's QoE.
	[177]	ESN is used to predict the content request distribution and mobility pattern of each user.	Only the downlink transmission is considered.
	[131]	A ML-based game theoretical approach is applied for efficient RRH assignment among MNOs.	Information exchange is required among RRHs.
	[132]	Game theory is applied by MEC servers to make activation decisions.	The offloaded computing tasks are assumed to be equally divided among the active servers.
	[133]	Decision tree algorithm is used to predict the opponent's network status.	The energy efficiency issue is not considered.

Continued on next page

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

TABLE X
ADVANTAGES AND SHORTCOMINGS OF THE ML-BASED SOLUTIONS (CONTINUED).

Continued from previous page

	Ref.	Advantages	Shortcomings
Resource Management	[134]	RL algorithm is applied to converge to the equilibrium of Stackelberg game.	The price charged to users is constant.
	[185]	Three learning models are used to learn the mapping between the hypervisor's CPU consumption and the control message rate.	The available computing resource of hypervisors are constant.
	[186]	SVM is used to detect the change of hypervisors' available computing resource in dynamic scenarios.	Only the computing resource (i.e., CPU) is taken into account.
	[187]	ML algorithms are used to obtain the real-time controller placement solutions.	Features such as network connectivity are not considered.
	[189]	Only the information available at the receiving nodes is used to estimate the number of active UEs.	The selection of master nodes in a D2D WiFi-Direct network is not considered.
	[192]	LSTM and decision tree are used to predict the most likely SLA Violation.	Only two use cases are discussed.
	[193]	Statistical machine learning is used to predict interference patterns in WSN.	Comprehensive experiments have not been done.
Security	[200]	Reactive routing is used to install different flow rules for different flow types.	Relevant contextual information is not considered in the intrusion detection system.
	[201]	HMM is used to predict malicious activities and enhance network security.	Each of the selected features (e.g., the length of the packet and source port) is treated as an independent event.
	[202]	SVM is utilized to identify abnormal activities.	Unknown traffic flows are analyzed manually by a human administrator.
	[203]	Machine learning techniques are used to detect malicious activities of smart devices.	The intrusion detection and mitigation process of all smart devices in a home network are not feasible by using IoT-IDM.
	[204]	Four machine learning algorithms are used to predict the potential malicious connections and vulnerable hosts.	The entire subnet is blocked to prevent potential attacks.
	[205]	Deep NN model is applied to classify traffic flows into normal and anomaly classes.	Only six features are utilized to train the deep NN model.
	[207]	Deep recurrent NN is used in an anomaly-based IDS.	Only six features are considered to detect intrusion.
	[208]	Decision tree is used as a feature reduction approach to select the most qualified features.	Comparison between SVM and other ML algorithms is not given.
	[209]	Deep learning is applied for feature reduction, and random forest is used for intrusion detection.	The performance of the proposed NDAE based on real-world backbone network traffic has not been evaluated.
	[210]	SOM is used to detect DDoS attack in SDN.	Which hosts are launching attacks cannot be determined precisely.
	[211]	Different machine learning algorithms are used to classify network traffic as normal and abnormal.	Only the DDoS attack is considered.
	[212]	Deep learning is applied to detect DDoS attack.	The SDN controller has a high overhead when performing feature extraction and DDoS attack detection.
	[214]	HMM is applied to identify whether a network connection is legitimate or not.	The scalability of the proposed method is not discussed when the number of firewall filtering rules is large.
	[215]	ML approaches are used to predict feeder-link outage in HTS system.	A full SDN-based network architecture has not been designed.

[155], [222], [223] have been proposed to solve the scalability issue. In general, a distributed multi-controller platform is composed of a logically centralized root controller and several local controllers. The root controller has a global view of the entire network and has full accessibility to all of the switches. On the contrary, each local controller has network status information in a domain and can only control part of the switches.

In order to optimize the routing of intra-domain and inter-domain traffic flows, a multi-level RL scheme can be used, where the root controller is the higher-level learning agent and the local controllers are the lower-level learning agents. Each lower-level learning agent learns to make the optimal decisions to route intra-domain traffic flows based on its local network status information, while the higher-level learning agent aims to process inter-domain traffic flows based on the global view of the entire network. In order to reduce the system response time, the root controller can deploy the trained RL models on the local controllers periodically. The trained RL models will guide the local controllers to process inter-domain traffic flows directly. The multi-level RL scheme not only reduces the processing delay of traffic flows, but also improves the scalability of the SDN network.

Due to the single point failure of the controller, reliability is another critical issue in SDN. To solve the reliability issue of the SDN controller, a cluster of controllers can be deployed to control and manage an SDN network. When a flow table update request arrives, it can be processed by a cluster of controllers. It is interesting to study how to select an optimal controller to process the flow table update request.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

RL algorithm is a possible approach to solve the problem. In this case, the network hypervisor works as a learning agent. The agent aims to maximize network utility by selecting an optimal controller to process the flow table update request according to the health condition, resource usage, as well as other information of each controller.

C. Improving Network Security

The separation of data plane and control plane reduces the complexity of network devices and provides a flexible network management. Since switches in the data plane do not have any intelligence, they just send raw data packets to the controller. Unfortunately, this behavior introduces a serious vulnerability which can be used by attackers to overload the controller through a large number of flow requests. ML-based anomaly detection is often used by the SDN controller to detect and process network attacks. However, anomaly detection is an adversarial problem where malicious attackers are continually trying to create new attacks to avoid detection of the controller. In this case, using historical data to train ML models may not be an effective method to detect attacks due to the crafting of new attacks. Generative Adversarial Network (GAN) [224] is a possible approach to solve the problem by predicting new attacks.

GAN is comprised of two neural networks. One neural network, called the generator, generates new data, while the other neural network, called the discriminator, is responsible for evaluating the new data for authenticity according to the real training dataset. The generator and discriminator train themselves together to make the generated new data more realistic. GAN can be employed to generate possible new attacks' data based on the historical data. After training ML models by using the generated new data and the historical data, the trained ML models can detect the known attacks and possible new attacks. Based on the attack detection, the controller is able to modify flow tables in switches in advance to prevent network attacks and limit the communication between the control plane and the data plane.

D. Cross-layer Network Optimization

Traditionally, networking is divided into different layers, and a set of protocols are designed for communications between adjacent layers. In traditional networks, direct communications between non-adjacent layers are not allowed. Recent studies [225]–[227] show that sharing information between non-adjacent layers can improve network performance significantly. However, the cross-layer design breaks the principle of modularity and makes the network so complex that traditional approaches are inadequate to optimize such networks. Fortunately, ML algorithms can be utilized for the cross-layer network optimization.

In SDN, the controller has a global network view and can collect the cross-layer information from all different layers, such as channel state information at the physical layer, packet information at the data link/network layers, and application information at application layer. Then the collected information can be used by machine learning algorithms to do the network optimization, such as physical layer parameters adaptation, resource allocation, topology construction, routing mechanism and congestion control. Therefore, it is interesting to study the ML-based cross-layer optimization approaches in SDN.

E. Incrementally Deployed SDN

Despite the promising prospect of SDN, its deployment needs to update all network switches to be SDN-aware. In the circumstance, the widespread deployment of pure commercial SDN would not be done in the near future. The incremental deployment [228], [229] is a feasible solution. In such a network, SDN switches and controllers are incrementally deployed in the traditional network and only parts of the network traffic is controlled by the controller. In this scenario, how to perform effective traffic engineering and optimize resource allocation remains an active research direction. One possible solution is that the SDN controller communicates with the other traditional network nodes to exchange link weights, available bandwidth and topology information. In this way, the SDN controller gets the required network information. By analyzing these information, as well as collected flow statistics information from SDN-enabled switches, machine learning algorithms can be utilized to create models, with which resource allocation optimization and traffic engineering can be performed effectively.

VII. Some Broader Perspectives

Since SDN has attracted widespread attention and been studied widely, its development can be influenced by a lot of other technologies. In the mean time, the SDN architecture also impacts both wired and wireless networks, such as vehicle networks, cellular networks (e.g., 4G and 5G networks) and sensor networks. In this section, some broader perspectives of applications of ML-based SDN in different network scenarios are presented.

A. Software Defined Network Function Virtualization

Network Function Virtualization (NFV) [230] is a promising technology to enable a more flexible and open network architecture, by virtualizing network functions and decoupling network functions from the underlying specialized hardware. NFV makes network reconfiguration quick and adaptive. In addition, it can reduce ISPs' capital expenditures for scaling up the network. NFV and SDN are two closely related technologies to make the network easy to control and manage. The difference between them is that SDN is applied to control network resources, while NFV focuses on the softwarization of network functions by using virtualization technologies. Integrated with SDN, the software defined NFV architecture is able to jointly optimize network functions and resources [62], [231]. The goal of the software defined NFV architecture is to automate network configuration, provision and management.

ML can promote the dynamic service provision and the network resource utilization optimization in software defined NFV systems. Now many works have been done to reduce the service providing cost and improve the utilization of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

network resources by using ML techniques, especially RL algorithms. In [232], Markov Decision Process (MDP) and Bayesian learning method are applied to optimize the dynamic resource allocation for NFV components. Ref. [233] focuses on the Service Function Chaining (SFC) in NFV. RL is used to create service chains dynamically based on the resource usage condition to enable efficient service providing. The network function assignment issue has been studied in [134]. The network function assignment problem is formulated as a two-stage Stackelberg game, where servers act as the sellers of network functions and users act as the buyers of network functions. RL algorithm is applied to obtain the optimal network function assignment strategy.

B. Software Defined Edge Computing

With the popularity of smartphones and wearable gadgets, such as smart glass, smart watch and smart bracelet, Edge Computing as a novel paradigm has attracted widespread attention. In recent years, a few Edge Computing architectures have been presented, such as Cloudlet [234], Edge Computing [235], Fog Computing [236], Mobile Cloud Computing (MCC) [237], Mist Computing [238] and Mobile-Edge Computing (MEC) [239].

A critical role of Edge Computing is computation offloading of computation-intensive tasks from the less capable devices to the powerful edge servers, hence extending the battery life time of the devices as well as speeding up the process of computation-intensive tasks. Computation offloading is a challenging problem due to the dynamic mobile environment such as mobility, heterogeneity, varying network conditions and user requirements. The computation offloading problem is generally considered as a decision-making task. Supervised learning algorithms, especially neural networks, are effective to obtain the optimal heuristic-like offloading decisions. Ref. [240] focuses on the computation offloading problem from mobile users' perspective, and uses a deep learning approach to decide which tasks are offloaded to edge servers. An exhaustive-based optimal offloading algorithm is utilized to obtain the labeled training dataset, which is used to train the deep learning model. RL is another effective approach for decision-making tasks. In [241], the computation offloading problem is considered from edge servers' perspective. RL algorithm is used by edge servers to allocate their limited computing resources according to the dynamic resource requests from mobile users.

Although many works have been done for computation offloading, only local network information is used by mobile users and edge servers to make decisions. To use computing resources more efficiently, global network information, including available computing resources of edge servers, resource requirements from mobile users and dynamic network conditions, should be considered to make computation offloading decisions. In this case, the software defined Edge Computing is proposed to meet the requirements of mobile users and improve the utilization of computing resources [59].

C. Software Defined Optical Networks

Optical networks have high transmission capacities and are widely deployed in our modern information systems. On the other hand, the specific optical transmission and switching characteristics, such as burst, circuit and packet switching on wavelength channels, make it difficult to control optical networks. To solve the problem, Software Defined Optical Network (SDON) [55], [242] has been proposed to enable the networking applications to use underlying optical network infrastructure dynamically and efficiently, by leveraging the ability of logically centralized control in SDN.

The main difference between optical networks and other networks is the optical physical layer of optical networks where lightpaths are established to transmit data. The Quality of Transmission (QoT) of a lightpath can be affected by many network parameters such as baud rate, code rate, modulation format, and so on. In order to improve the design and planning of optical networks, QoT prediction prior to the deployment of a new lightpath is very important. Supervised learning algorithms are effective methods to perform QoT prediction. In [243], k-NN and random forest are used to predict QoT according to the lightpath total length, traffic volume, the longest link length, modulation format and the number of lightpath links. Ref. [244] uses neural network as the QoT prediction tool.

Fault management is another critical task in optical networks. Lightpaths have high data transmission capacities, so the failure of several lightpaths can result in huge data losses. Supervised learning algorithms are effective data-driven methods to manage network faults based on historical failure incidents [245]. In [246], statistical machine learning techniques are used for failure localization by predicting each link's failure probability. Ref. [247] uses neural network to assess the network performance by predicting the blocking probability of optical networks.

In addition, in order to provide flexible, reconfigurable and economical end-to-end services, RL is often utilized to allocate resources in the optical network infrastructure components efficiently. In [248], RL is applied for resource allocation optimization to meet the service requirements.

In summary, ML is an effective data-driven method to improve the performance of optical networks by facilitating the flexible optical transmission management and the cost-effective resource utilization.

D. Software Defined Internet of Things

In the future, nearly everything will be connected to the Internet, from traditional communication tools (e.g., laptops and smartphones) to home appliances (e.g., refrigerators and garage doors). Typically, an Internet of Things (IoT) architecture can provide different applications such as intelligent transportation systems [249], smart health-care [250], [251], and smart energy systems [252]–[254], by leveraging large-scale distributed embedded systems to connect billions of sensors and RFID nodes. In order to manage the rapidly increasing number of devices, an intelligent, efficient, secure, cost-effective and scalable IoT needs to be designed. In this

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

case, the software defined IoT architecture [255] is proposed to satisfy the requirements of IoT. In the software defined IoT architecture, the billions of connected devices are controlled by the network operators and users remotely, and the real-time information is collected to provide intelligent services [58].

There is no doubt that IoT devices will generate a large amount of data [256]. ML techniques are often applied to process these data. Most of the ML algorithms generally require many storage and computing resources, thus a natural way to train ML models is to use a centralized approach. However, transferring huge amounts of raw data to the centralized SDN controller consumes a lot of network bandwidth. In order to reduce network bandwidth consumption and improve the system response time, edge computing is often used to pre-process the raw data [257]–[259]. Then, the reduced intermediate data is transferred to the centralized SDN controller. This way not only reduces the amount of transferred data, but also speeds up the training of ML models in the SDN controller. By deploying the trained ML models on edge servers, the system can improve the response time of IoT services.

Network security is an important research direction in IoT. Supervised learning algorithms have been applied to enhance security of the edge computing-based IoT systems. In [260], deep NN is used to perform anomaly detection at edge servers. SVM is employed by [261] to analyze sensor data and detect anomaly activities. Ref. [262] uses deep learning to detect the IoT cyber-attacks in edge networks.

In summary, the combination of SDN, edge computing and ML techniques can promote the deployment of IoT systems and the development of IoT services.

E. Software Defined Vehicular Networks

With the advancement of information communication and technology, smart vehicles have attracted widespread attention in recent years. To enable the implementation of smart vehicles, it is essential for vehicles to access the Internet and communicate with each other through vehicular networks [263]. Vehicular networks aim to provide comfort and convenience for drivers and passengers, improve traffic efficiency and enhance vehicle road safety by sharing real-time traffic information among vehicles, such as traffic jams, accident prone areas, hurdles and other important traffic information [264]. In vehicular networks, a vehicle usually has multiple network interfaces (e.g., WiFi, DSRC, UMTS, WiMax and Bluetooth) to communicate with RSUs and ambient vehicles. However, due to the intrinsic characteristics of vehicular networks such as diverse vehicle densities, sparse distribution of RSUs, high mobility of vehicles, dynamic traffic conditions and application requirements [265], designing an efficient data forwarding system for vehicular networks is a challenging task.

Unicast, multicast and broadcast are three routing approaches. Multicast and broadcast are not energy efficient due to the redundant transmission of traffic information. Unicast routing approach transmits traffic information to the intended destination through a specific path. Unicast routing approach can be divided into two categories: topology-based and position-based routing. Topology-based routing approaches make data forwarding decisions based on link information among vehicles. In [266], [267], fuzzy logic is employed to evaluate wireless links by considering link quality, the available bandwidth and vehicles' mobility. Then, the evaluation results are used by Q-learning algorithm to select the optimal route. In [268], a collaborative Learning Automata (LA)-based routing strategy is proposed to select the optimal path with minimum delay adaptively according to vehicle-related information such as vehicle density and distance from RSUs. A ML-assisted Route Selection (MARS) system is proposed by [269] to optimize routing decisions. The k-means algorithm is applied to predict vehicles' movement. Based on the prediction results, routing paths with better transmission capacity will be selected to transmit information. Different from topology-based routing approaches, position-based routing approaches make data forwarding decisions based on vehicles' position information [270]. In [271], a city is divided into multiple grids. Vehicles in different grids select the optimal next-hop grid to transmit information by using Q-learning algorithm.

Topology-based routing approaches can select the neighbor vehicle with a better link condition to transmit information, but they may make the local optimal decisions due to the lack of destination vehicles' position information. On the contrary, position-based routing approaches can decrease the communication hops by selecting the neighbor vehicle that is closest to the destination vehicle, but their major shortcoming is the instability of the selected route caused by weak signal strength due to the lack of link information. Thus, to improve the routing decisions in vehicular networks, both link information and vehicles' position information should be considered.

The global network view of the SDN controller simplifies the collection and analysis of network information. In this case, the Software Defined Vehicular Network (SDVN) [272] has been presented to reduce the communication cost and improve the network performance. Based on the collected link and position information, ML techniques such as LSTM can be used to predict the movement of vehicles and the topology change of vehicular networks. Then, the prediction results can be used by the SDN controller to make data forwarding decisions by using RL algorithms. Thus, ML techniques can promote the implementation of SDVN and enable an intelligent, safe and efficient vehicular network.

F. Software Defined Mobile Networks

After several decades of development, mobile networks have evolved to 5G [273], which is expected to support an increasing number of connected devices, provide higher user data rate, enable higher mobile data traffic per geographical area, reduce transmission delay and network energy consumption. Besides the specific performance requirements, 5G also needs to support heterogeneous services, devices and access networks [274]. To meet challenges from the heterogeneous wireless environments, the complexity of network management, the increasing mobile traffic demand and diverse service requirements in 5G mobile networks, ML techniques have been utilized to deploy more intelligence in mobile

networks. Generally, ML algorithms are applied to achieve self-configuration, self-optimization and self-healing functions [12].

Self-configuration focuses on the configuration and deployment of mobile networks to make the network operable. ML algorithms are often used to configure network parameters automatically, such as operational parameters, radio related parameters and other network parameters. The self-optimization function aims to optimize network performance and update network parameters by continuously monitoring the network environment. ML algorithms can be used in many aspects, such as mobility management, handover parameter optimization, load balancing and resource optimization. The objectives of self-healing function are failure detection, failure analysis and fast failure recovery. ML algorithms are mainly used for fault detection, fault classification and cell outage management. Ref. [12] has surveyed the machine learning algorithms and their solutions to achieve the self-configuration, self-optimization and self-healing functions.

Data is very important for ML algorithms. The global network view of the SDN controller simplifies the collection and analysis of network information. In this case, the Software Defined Mobile Network (SDMN) has been presented to promote the implementation of intelligent mobile networks by utilizing software oriented design [57].

G. Software Defined Wireless Sensor Networks

In recent years, the development of smart sensors has promoted the advancements of Wireless Sensor Network (WSN). The WSN is composed of a large number of small, inexpensive and intelligent sensor nodes capable of monitoring and collecting environmental information such as temperature, pressure, humidity and movement. However, smart sensors generally have limited storage resource, energy, computational capability and communication bandwidth. These device constraints bring several challenges in heterogeneous nodes management and network configuration. SDN is an emerging network paradigm to simplify network management and configuration. Recently, there is an increasing trend of integrating WSN with SDN to foster efficiency and sustainability in WSN [275]. In this case, the Software Defined WSN (SDWSN) model [275], [276] is proposed. Machine learning algorithms play a critical role in SDWSN to manage a mass of heterogeneous sensor nodes, optimize the resource utilization in each node, as well as to schedule communication links flexibly and efficiently. Now ML techniques have been adopted to address many issues in WSN, such as routing optimization, node clustering and data aggregation, event detection and query processing, localization [277], intrusion detection [278], fault detection, and so on. Ref. [9] has surveyed the machine learning algorithms and their solutions to solve these issues in WSN.

VIII. Conclusion

This article provided a survey of current ML techniques applied to SDN. We began our discussion with related survey papers and background knowledge of SDN. Thereafter, an overview of ML algorithms was presented. Then, how ML algorithms are applied in the realm of SDN was discussed in detail, from the perspective of traffic classification, routing optimization, QoS/QoE prediction, resource management and security. We also discussed significant research challenges and future research directions in ML-based SDN, including high-quality training datasets, distributed multi-controller platform, improving network security, cross-layer network optimization, and incrementally deployed SDN. Finally, we explored some broader perspectives, such as software defined edge computing, software defined vehicular networks, software defined mobile networks, etc.

In summary, research on applying ML algorithms in SDN is quite broad and many challenges lay ahead. Nevertheless, it is favorable for the network community to address the challenges and go forward. This article attempts to briefly explore how ML algorithms work and when they should be used to solve problems in SDN. We hope that our discussion and exploration may open a new avenue for the development of SDN and the implementation of a more intelligent network.

References

[1] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the Internet," in *Proc. ACM SIGCOMM'03, Karlsruhe, Germany*, 2003, pp. 3–10.

[2] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, G. Estrada, K. Ma'ruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. Evans, F. Maino, J. Walrand, and A. Cabellos, "Knowledge-defined networking," *SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 2–10, sep. 2017.

[3] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, March 2018.

[4] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, "Unsupervised machine learning for networking: Techniques, applications and research challenges," *arXiv preprint arXiv:1709.06599*, 2017.

[5] G. Xu, Y. Mu, and J. Liu, "Inclusion of artificial intelligence in communication networks and services," *ITU Journal: ICT Discoveries*, no. 1, pp. 1–6, Oct. 2017.

[6] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[7] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, Fourth Quarter 2008.

[8] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A survey on machine-learning techniques in cognitive radios," *IEEE Commun. Surveys Tutorials*, vol. 15, no. 3, pp. 1136–1159, Third Quarter 2013.

[9] M. A. Alsheikh, S. Lin, D. Niyato, and H. P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tutorials*, vol. 16, no. 4, pp. 1996–2018, Fourth Quarter 2014.

[10] X. Wang, X. Li, and V. C. M. Leung, "Artificial intelligence-based techniques for emerging heterogeneous network: State of the arts, opportunities, and challenges," *IEEE Access*, vol. 3, pp. 1379–1391, 2015.

[11] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, Second Quarter 2016.

[12] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self organizing cellular networks," *IEEE Commun. Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

[13] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 4, pp. 2432–2455, Fourth Quarter 2017.

[14] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," *arXiv preprint arXiv:1701.02145*, 2017.

[15] X. Zhou, M. Sun, G. Y. Li, and B.-H. Juang, "Machine learning and cognitive technology for intelligent wireless networks," *arXiv preprint arXiv:1710.11240*, 2017.

[16] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *arXiv preprint arXiv:1710.02913*, 2017.

[17] "Open Networking Foundation," Jun. 2014. [Online]. Available: https://www.opennetworking.org/

[18] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? implementation challenges for software-defined networks," *IEEE Commun. Magazine*, vol. 51, no. 7, pp. 36–43, July. 2013.

[19] "Open vSwitch," May. 2018. [Online]. Available: https://www.openvswitch.org/

[20] "Indigo: Open Source OpenFlow Switches," May. 2018. [Online]. Available: http://www.projectfloodlight.org/indigo/

[21] "Pantou: OpenFlow 1.3 for OpenWRT," May. 2018. [Online]. Available: https://github.com/CPqD/ofsoftswitch13/wiki/OpenFlow-1.3-for-OpenWRT

[22] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "NetFPGA: An open platform for gigabit-rate network switching and routing," in *Proc. IEEE MSE'07, San Diego, CA, USA*, June 2007, pp. 160–161.

[23] M. B. Anwer, M. Motiwala, M. b. Tariq, and N. Feamster, "SwitchBlade: A platform for rapid deployment of network protocols on programmable hardware," in *Proc. ACM SIGCOMM'10, New Delhi, India*, 2010, pp. 183–194.

[24] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, H. Wu, Y. Xiong, R. Gao, and Y. Zhang, "ServerSwitch: A programmable and high performance platform for data center networks," in *NSDI*, vol. 11, 2011, pp. 2–2.

[25] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, July. 2008.

[26] M. McCauley, "About Pox," 2013. [Online]. Available: http://www.noxrepo.org/pox/about-pox/

[27] Floodlight, "Project Floodlight open source software for building softwaredefined networks," 2012. [Online]. Available: http://www.projectfloodlight.org/

[28] Ryu, "Ryu SDN Framework," 2013. [Online]. Available: http://osrg.github.io/ryu/

[29] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in *Proc. IEEE WoWMoM'14, Sydney, NSW*, June. 2014, pp. 1–6.

[30] D. Erickson, "The Beacon OpenFlow controller," in *Proc. ACM SIGCOMM Workshop HotSDN'13, Hong Kong, China*, 2013, pp. 13–18.

[31] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[32] B. Pfaff and B. Davie, "The open vSwitch database management protocol," Dec. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc7047.txt

[33] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and control element separation (ForCES) protocol specification," Tech. Rep., 2010. [Online]. Available: https://rfc-editor.org/rfc/rfc5810.txt

[34] H. Song, "Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane," in *Proc. ACM SIGCOMM Workshop HotSDN'13, Hong Kong, China*, 2013, pp. 127–132.

[35] R. Enns, M. Bjorklund, and J. Schoenwaelder, "Network configuration protocol (NETCONF)," Jun. 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6241.txt

[36] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller, "The locator/ID separation protocol (LISP)," Jan. 2013. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6830.txt

[37] M. Smith, M. Dvorkin, Y. Laribi, V. Pandey, P. Garg, and N. Weidenbacher, "OpFlex control protocol," *IETF*, Apr. 2014. [Online]. Available: http://www.ietf.org/internet-drafts/draft-smith-opflex-00.txt

[38] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "OpenState: Programming platform-independent stateful OpenFlow applications inside the switch," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 44–51, 2014.

[39] Open Networking Foundation, "Common Information Model Overview. V1.2," 2016. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/TR-513_CIM_Overview_1.2.pdf

[40] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A distributed control platform for large-scale production networks," in *OSDI*, vol. 10, 2010, pp. 1–6.

[41] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for OpenFlow," in *Proc. Enterprise Networking'10*, 2010, pp. 3–3.

[42] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi, "SDNi: A message exchange protocol for Software Defined Networks (SDNs) across multiple domains," *IETF Draft*, 2012.

[43] P. Lin, J. Bi, and Y. Wang, "East-west bridge for SDN network peering," in *Frontiers in Internet Technologies*. Springer, 2013, pp. 170–181.

[44] F. Benamrane, R. Benaini *et al.*, "An east-west interface for distributed SDN control plane: Implementation and evaluation," *Computers & Electrical Engineering*, vol. 57, pp. 162–175, 2017.

[45] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, "A survey on the contributions of software-defined networking to traffic engineering," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 2, pp. 918–953, Second Quarter 2017.

[46] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tutorials*, vol. 17, no. 4, pp. 2317–2346, Fourth Quarter 2015.

[47] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Commun. Surveys Tutorials*, vol. 18, no. 1, pp. 623–654, First Quarter 2016.

[48] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 1, pp. 325–346, First Quarter 2017.

[49] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Trans. Reliability*, vol. 64, no. 3, pp. 1086–1097, Sept. 2015.

[50] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tutorials*, vol. 18, no. 1, pp. 602–622, First Quarter 2016.

[51] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful SDN data planes," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 3, pp. 1701–1725, Third Quarter 2017.

[52] P. Fonseca and E. Mota, "A survey on fault management in software-defined networks," *IEEE Commun. Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

[53] J. W. Guck, A. V. Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Commun. Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

[54] R. Alvizu, G. Maier, N. Kukreja, A. Pattavina, R. Morro, A. Capello, and C. Cavazzoni, "Comprehensive survey on T-SDN: Software-defined networking for transport networks," *IEEE Commun. Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

[55] A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein, and W. Kellerer, "Software Defined Optical Networks (SDONs): A comprehensive survey," *IEEE Commun. Surveys Tutorials*, vol. 18, no. 4, pp. 2738–2786, Fourth Quarter 2016.

[56] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: A survey and taxonomy," *IEEE Commun. Surveys Tutorials*, vol. 18, no. 4, pp. 2713–2737, Fourth Quarter 2016.

[57] T. Chen, M. Matinmikko, X. Chen, X. Zhou, and P. Ahokangas, "Software defined mobile networks: Concept, survey, and research directions," *IEEE Commun. Magazine*, vol. 53, no. 11, pp. 126–133, Nov. 2015.

[58] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.

[59] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Commun. Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

[60] O. Michel and E. Keller, "SDN in wide-area networks: A survey," in *Proc. IEEE SDS'17, Valencia, Spain*, 2017, pp. 37–42.

[61] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: A survey," *IEEE Commun. Magazine*, vol. 51, no. 11, pp. 24–31, Nov. 2013.

[62] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[63] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tutorials*, vol. 17, no. 1, pp. 358–380, First Quarter 2015.

[64] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third Quarter 2014.

[65] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE Commun. Surveys Tutorials*, vol. 16, no. 4, pp. 1955–1980, Fourth Quarter 2014.

[66] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tutorials*, vol. 17, no. 1, pp. 27–51, First Quarter 2015.

[67] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: From concept to implementation," *IEEE Commun. Surveys Tutorials*, vol. 16, no. 4, pp. 2181–2206, Fourth Quarter 2014.

[68] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: A survey," *Computer Communications*, vol. 67, no. Supplement C, pp. 1 – 10, 2015.

[69] C. Trois, M. D. D. Fabro, L. C. E. de Bona, and M. Martinello, "A survey on SDN programming languages: Toward a taxonomy," *IEEE Commun. Surveys Tutorials*, vol. 18, no. 4, pp. 2687–2712, Fourth Quarter 2016.

[70] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and Y. Liu, "A survey on large-scale Software Defined Networking (SDN) testbeds: Approaches and challenges," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 2, pp. 891–917, Second Quarter 2017.

[71] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Commun. Surveys Tutorials*, vol. 18, no. 1, pp. 655–685, First Quarter 2016.

[72] OpenFlow Switch Consortium, "OpenFlow Switch Specification Version 1.0 of OpenFlow," Dec. 2009. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf

[73] M. Mohammed, M. B. Khan, and E. B. M. Bashier, *Machine Learning: Algorithms and Applications*. CRC Press, 2016.

[74] S. Marsland, *Machine Learning: An Algorithmic Perspective*. CRC Press, 2015.

[75] M. Kubat, *An Introduction to Machine Learning*. Springer, 2016.

[76] E. Alpaydin, *Introduction to Machine Learning*. MIT Press, 2014.

[77] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging Artificial Intelligence Applications in Computer Engineering*, vol. 160, pp. 3–24, 2007.

[78] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*. Springer Series in Statistics New York, 2001, vol. 1.

[79] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[80] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. CRC Press, 1984.

[81] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Elsevier, 2011.

[82] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[83] S. Karatsiolis and C. N. Schizas, "Region based support vector machine algorithm for medical diagnosis on Pima Indian Diabetes dataset," in *Proc. IEEE BIBE'12, Larnaca, Cyprus*, Nov. 2012, pp. 139–144.

[84] W. R. Burrows, M. Benjamin, S. Beauchamp, E. R. Lord, D. McCollor, and B. Thomson, "CART decision-tree statistical analysis and prediction of summer season maximum surface ozone for the Vancouver, Montreal, and Atlantic regions of Canada," *Journal of Applied Meteorology*, vol. 34, no. 8, pp. 1848–1862, 1995.

[85] A. Kumar, P. Bhatia, A. Goel, and S. Kole, "Implementation and comparison of decision tree based algorithms," *International Journal of Innovations & Advancement in Computer Science*, vol. 4, pp. 190–196, May. 2015.

[86] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[87] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1994.

[88] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks*, vol. 2, no. 2004, p. 41, 2004.

[89] K. Lee, D. Booth, and P. Alam, "A comparison of supervised and unsupervised neural networks in predicting bankruptcy of Korean firms," *Expert Systems with Applications*, vol. 29, no. 1, pp. 1–16, 2005.

[90] S. Timotheou, "The random neural network: A survey," *The Computer Journal*, vol. 53, no. 3, pp. 251–267, 2010.

[91] S. Basterrech and G. Rubino, "A tutorial about random neural networks in supervised learning," *arXiv preprint arXiv:1609.04846*, 2016.

[92] H. Bakircioglu and T. Koccak, "Survey of random neural network applications," *European Journal of Operational Research*, vol. 126, no. 2, pp. 319–330, 2000.

[93] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[94] J. Baker, "Artificial neural networks and deep learning," Feb. 2015. [Online]. Available: http://lancs.ac.uk/˜bakerj1/pdfs/ANNs/Artificial_neural_networks-report.pdf

[95] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[96] G. Pandey and A. Dukkipati, "Learning by stretching deep networks," in *International Conference on Machine Learning*, 2014, pp. 1719–1727.

[97] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[98] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of DDoS attack-based on deep learning in OpenFlow-based SDN," *International Journal of Communication Systems*, 2018.

[99] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[100] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[101] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[102] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *Proc. IEEE ICASSP'15, Brisbane, QLD, Australia*, April 2015, pp. 4520–4524.

[103] V. N. Vapnik and V. Vapnik, *Statistical Learning Theory*. Wiley New York, 1998, vol. 1.

[104] B. Yekkehkhany, A. Safari, S. Homayouni, and M. Hasanlou, "A comparison study of different kernel functions for SVM-based classification of multi-temporal polarimetry SAR data," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, no. 2, p. 281, 2014.

[105] A. Patle and D. S. Chouhan, "SVM kernel functions for classification," in *Proc. IEEE ICATE'13, Mumbai, India*, Jan 2013, pp. 1–9.

[106] I. Steinwart and A. Christmann, *Support Vector Machines*. Springer Science & Business Media, 2008.

[107] M. Martínez-Ramón and C. Christodoulou, "Support vector machines for antenna array processing and electromagnetics," *Synthesis Lectures on Computational Electromagnetics*, vol. 1, no. 1, pp. 1–120, 2005.

[108] H. Hu, Y. Wang, and J. Song, "Signal classification based on spectral correlation analysis and SVM in cognitive radio," in *Proc. IEEE AINA'08, Okinawa, Japan*, March. 2008, pp. 883–887.

[109] G. E. Box and G. C. Tiao, *Bayesian Inference in Statistical Analysis*. John Wiley & Sons, 2011, vol. 40.

[110] J. Bakker, "Intelligent traffic classification for detecting DDoS attacks using SDN/OpenFlow," *Victoria University of Wellington*, pp. 1–142, 2017.

[111] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[112] F. V. Jensen, *An Introduction to Bayesian Networks*. UCL Press London, 1996, vol. 210.

[113] D. Heckerman *et al.*, "A tutorial on learning with Bayesian networks," *Nato Asi Series D Behavioural And Social Sciences*, vol. 89, pp. 301–354, 1998.

[114] T. D. Nielsen and F. V. Jensen, *Bayesian Networks and Decision Graphs*. Springer Science & Business Media, 2009.

[115] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[116] P. Holgado, V. A. VILLAGRA, and L. Vazquez, "Real-time multistep attack prediction based on hidden markov models," *IEEE Trans. Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[117] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[118] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.

[119] M. M. Van Hulle, "Self-organizing maps," in *Handbook of Natural Computing*. Springer, 2012, pp. 585–622.

[120] X. Zhu, "Semi-supervised learning literature survey," *Citeseer*, pp. 1–59, 2005.

[121] X. Zhou and M. Belkin, "Semi-supervised learning," in *Academic Press Library in Signal Processing*. Elsevier, 2014, vol. 1, pp. 1239–1269.

[122] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on Challenges in Representation Learning, ICML*, vol. 3, 2013, p. 2.

[123] H. Wu and S. Prasad, "Semi-supervised deep learning using Pseudo labels for hyperspectral image classification," *IEEE Trans. Image Processing*, vol. 27, no. 3, pp. 1259–1270, March 2018.

[124] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Trans. Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[125] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press Cambridge, 1998, vol. 1, no. 1.

[126] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[127] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[128] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[129] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[130] Y. He, C. Liang, R. Yu, and Z. Han, "Trust-based social networks with computing, caching and communications: A deep reinforcement learning approach," *IEEE Trans. Network Science and Engineering*, pp. 1–1, 2018.

[131] O. Narmanlioglu and E. Zeydan, "Learning in SDN-based multi-tenant cellular networks: A game-theoretic perspective," in *Proc. IEEE INM'17, Lisbon, Portugal*, May. 2017, pp. 929–934.

[132] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," *arXiv preprint arXiv:1711.09012*, 2017.

[133] F. Cai, Y. Gao, L. Cheng, L. Sang, and D. Yang, "Spectrum sharing for LTE and WiFi coexistence using decision tree and game theory," in *Proc. IEEE WCNC'16, Doha, Qatar*, April 2016, pp. 1–6.

[134] S. DOro, L. Galluccio, S. Palazzo, and G. Schembra, "A game theoretic approach for distributed resource allocation and orchestration of softwarized networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 3, pp. 721–735, March 2017.

[135] H.-Y. Shi, W.-L. Wang, N.-M. Kwok, and S.-Y. Chen, "Game theory for wireless sensor networks: A survey," *Sensors*, vol. 12, no. 7, pp. 9055–9097, 2012.

[136] H. Zhang, J. Du, J. Cheng, K. Long, and V. C. M. Leung, "Incomplete CSI based resource optimization in SWIPT enabled heterogeneous networks: A non-cooperative game theoretic approach," *IEEE Trans. Wireless Communications*, vol. 17, no. 3, pp. 1882–1892, March 2018.

[137] J. Xie, R. Xie, T. Huang, J. Liu, F. R. Yu, and Y. Liu, "Caching resource sharing in radio access networks: A game theoretic approach," *Frontiers of Information Technology & Electronic Engineering*, vol. 17, no. 12, pp. 1253–1265, 2016.

[138] A. Agrawal and D. Jaiswal, "When machine learning meets AI and game theory," *Stanford University, Machine Learning Final Year Projects*, pp. 1–5, 1981.

[139] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: Data collection and traffic classification," in *Proc. IEEE ICNP'16, Singapore, Singapore*, Nov. 2016, pp. 1–5.

[140] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in SDN," in *Proc. ACM SIGCOMM'13, Hong Kong, China*, 2013, pp. 487–488.

[141] P. Wang, S. C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs," in *Proc. IEEE SCC'16, San Francisco, CA, USA*, June. 2016, pp. 760–765.

[142] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM IMC'10, Melbourne, Australia*, 2010, pp. 267–280.

[143] M. Glick and H. Rastegarfar, "Scheduling and control in hybrid data centers," in *Proc. IEEE PHOSST'17, San Juan, Puerto Rico*, July. 2017, pp. 115–116.

[144] P. Xiao, W. Qu, H. Qi, Y. Xu, and Z. Li, "An efficient elephant flow detection with cost-sensitive in SDN," in *Proc. IEEE INISCom'15, Tokyo, Japan*, March. 2015, pp. 24–28.

[145] Y. Li and J. Li, "MultiClassifier: A combination of DPI and ML for application-layer classification in SDN," in *Proc. IEEE ICSAI'14, Shanghai, China*, Nov. 2014, pp. 682–686.

[146] D. Rossi and S. Valenti, "Fine-grained traffic classification with Netflow data," in *Proc. ACM IWCMC'10, Caen, France*, 2010, pp. 479–483.

[147] A. NAKAO and P. DU, "Toward in-network deep machine learning for identifying mobile applications and enabling application specific network slicing," *IEICE Trans. Communications*, p. 2017CQI0002, 2014.

[148] M. Uddin and T. Nadeem, "TrafficVision: A case for pushing software defined networks to wireless edges," in *Proc. IEEE MASS'16, Brasilia, Brazil*, Oct. 2016, pp. 37–46.

[149] R. Hajlaoui, H. Guyennet, and T. Moulahi, "A survey on heuristic-based routing methods in vehicular ad-hoc network: Technical challenges and future trends," *IEEE Sensors Journal*, vol. 16, no. 17, pp. 6782–6792, Sept. 2016.

[150] L. Yanjun, L. Xiaobo, and Y. Osamu, "Traffic engineering framework with machine learning based meta-layer in software-defined networks," in *Proc. IEEE ICNIDC'14, Beijing, China*, Sept. 2014, pp. 121–125.

[151] A. Azzouni, R. Boutaba, and G. Pujolle, "NeuRoute: Predictive dynamic routing for software-defined networks," *arXiv preprint arXiv:1709.06002*, 2017.

[152] S. Sendra, A. Rego, J. Lloret, J. M. Jimenez, and O. Romero, "Including artificial intelligence in a routing protocol using software defined networks," in *Proc. IEEE ICC Workshops'17, Paris, France*, May. 2017, pp. 670–674.

[153] F. Francois and E. Gelenbe, "Optimizing secure SDN-enabled inter-data centre overlay networks through cognitive routing," in *Proc. IEEE MASCOTS'16, London, UK*, Sept. 2016, pp. 283–288.

[154] ——, "Towards a cognitive routing engine for software defined networks," in *Proc. IEEE ICC'16, Kuala Lumpur, Malaysia*, May. 2016, pp. 1–6.

[155] S. C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "QoS-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *Proc. IEEE SCC'16, San Francisco, CA, USA*, June. 2016, pp. 25–33.

[156] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntes-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," *arXiv preprint arXiv:1709.07080*, 2017.

[157] Á. López-Raventós, F. Wilhelmi, S. Barrachina-Muñoz, and B. Bellalta, "Machine learning and software defined networks for high-density WLANs," *arXiv preprint arXiv:1804.05534*, 2018.

[158] R. Alvizu, S. Troia, G. Maier, and A. Pattavina, "Matheuristic with machine-learning-based prediction for software-defined mobile metro-core networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 9, pp. D19–D30, Sept. 2017.

[159] C. Chen-Xiao and X. Ya-Bin, "Research on load balance method in SDN," *International Journal of Grid and Distributed Computing*, vol. 9, no. 1, pp. 25–36, 2016.

[160] A. Azzouni and G. Pujolle, "NeuTM: A neural network-based framework for traffic matrix prediction in SDN," *arXiv preprint arXiv:1710.06799*, 2017.

[161] "GEANT Network," May. 2018. [Online]. Available: https://www.geant.org/Projects/GEANT_Project_GN4

[162] K. K. Budhraja, A. Malvankar, M. Bahrami, C. Kundu, A. Kundu, and M. Singhal, "Risk-based packet routing for privacy and compliance-preserving SDN," in *Proc. IEEE CLOUD'17, Honolulu, CA, USA*, June. 2017, pp. 761–765.

[163] J. Carner, A. Mestres, E. Alarcn, and A. Cabellos, "Machine learning-based network modeling: An artificial neural network model vs a theoretical inspired model," in *Proc. IEEE ICUFN'17, Milan, Italy*, 2017, pp. 522–524.

[164] S. Jain, M. Khandelwal, A. Katkar, and J. Nygate, "Applying big data technologies to manage QoS in an SDN," in *Proc. IEEE CNSM'16, Montreal, QC, Canada*, Oct. 2016, pp. 302–306.

[165] R. Pasquini and R. Stadler, "Learning end-to-end application QoS from OpenFlow switch statistics," in *Proc. IEEE NETSOFT'17, Bologna, Italy*, 2017, pp. 1–9.

[166] A. B. Letaifa, "Adaptive QoE monitoring architecture in SDN networks: Video streaming services case," in *Proc. IEEE IWCMC'17, Valencia, Spain*, 2017, pp. 1383–1388.

[167] T. Abar, A. B. Letaifa, and S. E. Asmi, "Machine learning based QoE prediction in SDN networks," in *Proc. IEEE IWCMC'17, Valencia, Spain*, 2017, pp. 1395–1400.

[168] R. Huo, F. R. Yu, T. Huang, R. Xie, J. Liu, V. C. M. Leung, and Y. Liu, "Software defined networking, caching, and computing for green wireless networks," *IEEE Commun. Magazine*, vol. 54, no. 11, pp. 185–193, November 2016.

[169] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching, and computing in wireless systems: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tutorials*, vol. 20, no. 1, pp. 7–38, First Quarter 2018.

[170] Y. He, F. R. Yu, and A. Boukerche, "Deep reinforcement learning based resource management in software-defined and virtualized vehicular ad hoc networks," in *Proc. ACM DIVANet'17, Miami Beach, FL*, Nov. 2017.

[171] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Magazine*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[172] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Software-defined networking for RSU clouds in support of the internet of vehicles," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 133–144, April.2015.

[173] R. Haw, M. G. R. Alam, and C. S. Hong, "A context-aware content delivery framework for QoS in mobile cloud," in *Proc. IEEE APNOMS'14, Hsinchu, Taiwan*, Sept. 2014, pp. 1–6.

[174] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous, "SDNHAS: An SDN-enabled architecture to optimize QoE in HTTP adaptive streaming," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2136–2151, Oct. 2017.

[175] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. ACM SIGCOMM'14, Chicago, Illinois, USA*, 2014, pp. 187–198.

[176] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, April. 2014.

[177] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1046–1061, May 2017.

[178] S. Paris, J. Leguay, L. Maggi, M. Draief, and S. Chouvardas, "Online experts for admission control in SDN," in *Proc. IEEE NOMS'16, Istanbul, Turkey*, April. 2016, pp. 1003–1004.

[179] J. Leguay, L. Maggi, M. Draief, S. Paris, and S. Chouvardas, "Admission control with online algorithms in SDN," in *Proc. IEEE NOMS'16, Istanbul, Turkey*, April. 2016, pp. 718–721.

[180] S. Agrawal and N. R. Devanur, "Fast algorithms for online stochastic convex programming," in *Proc. ACM SODA'15, San Diego, California*, 2015, pp. 1405–1424.

[181] N. Buchbinder and J. Naor, "Improved bounds for online routing and packing via a primal-dual approach," in *Proc. IEEE FOCS'06, Berkeley, CA, USA*, Oct. 2006.

[182] N. Buchbinder and J. S. Naor, "Online primal-dual algorithms for covering and packing," *Mathematics of Operations Research*, vol. 34, no. 2, pp. 270–286, 2009.

[183] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar, "Can the production network be the testbed?" in *OSDI*, vol. 10, 2010, pp. 1–6.

[184] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, "OpenVirteX: Make your virtual SDNs programmable," in *Proc. ACM HotSDN'14, Chicago, Illinois, USA*, 2014, pp. 25–30.

[185] C. Sieber, A. Basta, A. Blenk, and W. Kellerer, "Online resource mapping for SDN network hypervisors using machine learning," in *Proc. IEEE NETSOFT'16, Seoul, South Korea*, June. 2016, pp. 78–82.

[186] C. Sieber, A. Obermair, and W. Kellerer, "Online learning and adaptation of network hypervisor performance models," in *Proc. IEEE INM'17, Lisbon, Portugal*, May. 2017, pp. 1204–1212.

[187] M. He, P. Kalmbach, A. Blenk, W. Kellerer, and S. Schmid, "Algorithm-data driven optimization of adaptive communication networks," in *Proc. IEEE ICNP'17, Toronto, ON, Canada*, Oct. 2017, pp. 1–6.

[188] A. Blenk, P. Kalmbach, W. Kellerer, and S. Schmid, "O'Zapft is: Tap your network algorithm's big data!" in *Proc. ACM Big-DAMA'17, Los Angeles, CA, USA*, 2017, pp. 19–24.

[189] D. D. Testa, M. Danieletto, G. M. D. Nunzio, and M. Zorzi, "Estimating the number of receiving nodes in 802.11 networks via machine learning techniques," in *Proc. IEEE GLOBECOM'16, Washington, DC, USA*, Dec. 2016, pp. 1–7.

[190] D. D. Testa, M. Danieletto, and M. Zorzi, "A machine learning based ETA estimator for WiFi transmissions," *IEEE Trans. Wireless Communications*, vol. PP, no. 99, pp. 1–1, 2017.

[191] W. Jiang, M. Strufe, and H. Schotten, "Autonomic network management for software-defined and virtualized 5G systems," in *Proc. VDE European Wireless'17, Dresden, Germany*, May. 2017, pp. 1–6.

[192] J. Bendriss, I. G. B. Yahia, and D. Zeghlache, "Forecasting and anticipating SLO breaches in programmable networks," in *Proc. IEEE ICIN'17, Paris, France*, March. 2017, pp. 127–134.

[193] C. Orfanidis, "Ph.D. forum abstract: Increasing robustness in WSN using software defined network architecture," in *Proc. ACM/IEEE IPSN'16, Vienna, Austria*, April. 2016, pp. 1–2.

[194] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," *NIST Special Publication*, vol. 800, no. 2007, p. 94, 2007.

[195] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, pp. 1–13, 2017.

[196] J. Ashraf and S. Latif, "Handling intrusion and DDoS attacks in software defined networks using machine learning techniques," in *Proc. IEEE NSEC'14, Rawalpindi, Pakistan*, Nov. 2014, pp. 55–60.

[197] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, pp. 1–9, 2018.

[198] J. Ibrahim and S. Gajin, "SDN-based intrusion detection system," *Infoteh Jahorina*, vol. 16, pp. 621–624, March. 2017.

[199] M. Khairi, S. Ariffin, N. A. Latiff, A. Abdullah, and M. Hassan, "A review of anomaly detection techniques and Distributed Denial of Service (DDoS) on Software Defined Network (SDN)," *Engineering, Technology & Applied Science Research*, vol. 8, no. 2, pp. 2724–2730, 2018.

[200] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, "Machine-learning based threat-aware system in software defined networks," in *Proc. IEEE ICCCN'17, Vancouver, BC, Canada*, July. 2017, pp. 1–9.

[201] T. Hurley, J. E. Perdomo, and A. Perez-Pons, "HMM-based intrusion detection system for software defined networking," in *Proc. IEEE ICMLA'16, Anaheim, CA, USA*, Dec. 2016, pp. 617–621.

[202] A. S. da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, "ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN," in *Proc. IEEE NOMS'16, Istanbul, Turkey*, April. 2016, pp. 27–35.

[203] M. Nobakht, V. Sivaraman, and R. Boreli, "A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow," in *Proc. IEEE ARES'16, Salzburg, Austria*, Aug. 2016, pp. 147–156.

[204] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in SDN using machine learning approach," in *Proc. IEEE NFV-SDN'16, Palo Alto, CA, USA*, Nov. 2016, pp. 167–172.

[205] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. IEEE WINCOM'16, Fez, Morocco*, Oct. 2016, pp. 258–263.

[206] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *International Journal of Engineering Research and Technology. ESRSA Publications*, 2013.

[207] T. Tang, S. A. R. Zaidi, D. McLernon, L. Mhamdi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in SDN-based networks," in *Proc. IEEE NetSoft'18, Montreal, Canada*, 2018.

[208] P. Wang, K. M. Chao, H. C. Lin, W. H. Lin, and C. C. Lo, "An efficient flow control approach for SDN-based network threat detection and migration using support vector machine," in *Proc. IEEE ICEBE'16, Macau, China*, Nov. 2016, pp. 56–63.

[209] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, Feb 2018.

[210] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. IEEE LCN'10, Denver, CO, USA*, Oct. 2010, pp. 408–415.

[211] L. Barki, A. Shidling, N. Meti, D. G. Narayan, and M. M. Mulla, "Detection of distributed denial of service attacks in software defined networks," in *Proc. IEEE ICACCI'16, Jaipur, India*, Sept. 2016, pp. 2576–2581.

[212] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in Software Defined Networking (SDN)," *arXiv preprint arXiv:1611.07400*, 2016.

[213] L. J. Jagadeesan and V. Mendiratta, "Programming the network: Application software faults in software-defined networks," in *Proc. IEEE ISSREW'16, Ottawa, ON, Canada*, Oct. 2016, pp. 125–131.

[214] Z. Din and J. de Oliveira, "Anomaly free on demand stateful software defined firewalling," in *Proc. IEEE ICCCN'17, Vancouver, BC, Canada*, July. 2017, pp. 1–9.

[215] M. Mongelli, T. D. Cola, M. Cello, M. Marchese, and F. Davoli, "Feeder-link outage prediction algorithms for SDN-based high-throughput satellite systems," in *Proc. IEEE ICC'16, Kuala Lumpur, Malaysia*, May. 2016, pp. 1–6.

[216] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE CISDA'09, Ottawa, ON, Canada*, July. 2009, pp. 1–6.

[217] L. F. Maim, . L. P. Gmez, F. J. G. Clemente, M. G. Prez, and G. M. Prez, "A self-adaptive deep learning-based system for anomaly detection in 5G networks," *IEEE Access*, vol. 6, pp. 7700–7712, 2018.

[218] LongTail, "LongTail Log Analysis Dashboard," May. 2018. [Online]. Available: http://longtail.it.marist.edu/honey/dashboard.shtml

[219] "UNB ISCX intrusion detection evaluation dataSet," May. 2018. [Online]. Available: http://www.unb.ca/cic/datasets/ids.html

[220] "Imagenet database," May. 2018. [Online]. Available: http://www.image-net.org/

[221] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[222] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. ACM HotSDN'12, Helsinki, Finland*, 2012, pp. 19–24.

[223] J. McCauley, A. Panda, M. Casado, T. Koponen, and S. Shenker, "Extending SDN to large-scale networks," *Open Networking Summit*, pp. 1–2, 2013.

[224] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[225] C. She, C. Yang, and T. Q. S. Quek, "Cross-layer optimization for ultra-reliable and low-latency radio access networks," *IEEE Trans. Wireless Communications*, vol. PP, no. 99, pp. 1–1, 2017.

[226] G. Liu, Z. Ma, X. Chen, Z. Ding, R. Yu, and P. Fan, "Cross-layer power allocation in non-orthogonal multiple access systems for statistical QoS provisioning," *IEEE Trans. Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2017.

[227] A. Y. Al-Zahrani, F. R. Yu, and M. Huang, "A joint cross-layer and colayer interference management scheme in hyperdense heterogeneous networks using mean-field game theory," *IEEE Trans. Vehicular Technology*, vol. 65, no. 3, pp. 1522–1535, March. 2016.

[228] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. N. A. Corrêa, S. Cunha de Lucena, and R. Raszuk, "Revisiting routing control platforms with the eyes and muscles of software-defined networking," in *Proc. ACM HotSDN'12, Helsinki, Finland*, 2012, pp. 13–18.

[229] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE INFOCOM'13, Turin, Italy*, April. 2013, pp. 2211–2219.

[230] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: State of the art, challenges, and implementation in next generation mobile networks (vEPC)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, Nov. 2014.

[231] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, "Toward an SDN-enabled NFV architecture," *IEEE Commun. Magazine*, vol. 53, no. 4, pp. 187–193, April. 2015.

[232] R. Shi, J. Zhang, W. Chu, Q. Bao, X. Jin, C. Gong, Q. Zhu, C. Yu, and S. Rosenberg, "MDP and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization," in *Proc. IEEE SCC'15, New York, NY, USA*, June. 2015, pp. 65–73.

[233] S. I. Kim and H. S. Kim, "A research on dynamic service function chaining based on reinforcement learning using resource usage," in *Proc. IEEE ICUFN'17, Milan, Italy*, July. 2017, pp. 582–586.

[234] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[235] H. H. Pang and K. L. Tan, "Authenticating query results in edge computing," in *Proc. IEEE ICDE'04, Boston, MA, USA, USA*, March. 2004, pp. 560–571.

[236] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. ACM MCC'12, Helsinki, Finland*, 2012, pp. 13–16.

[237] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[238] J. S. Preden, K. Tammeme, A. Jantsch, M. Leier, A. Riid, and E. Calis, "The benefits of self-awareness and attention in Fog and Mist computing," *Computer*, vol. 48, no. 7, pp. 37–45, July. 2015.

[239] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, Third Quarter 2017.

[240] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: A deep learning approach," in *Proc. IEEE PIMRC'17, Montreal, QC, Canada*, Oct. 2017, pp. 1–6.

[241] M. G. R. Alam, Y. K. Tun, and C. S. Hong, "Multi-agent and reinforcement learning based code offloading in mobile fog," in *Proc. IEEE ICOIN'16, Kota Kinabalu, Malaysia*, Jan. 2016, pp. 285–290.

[242] P. N. Ji, "Software defined optical network," in *Proc. IEEE ICOCN'12, Chonburi, Thailand*, Nov. 2012, pp. 1–4.

[243] C. Rottondi, L. Barletta, A. Giusti, and M. Tornatore, "Machine-learning method for quality of transmission prediction of unestablished lightpaths," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 2, pp. A286–A297, Feb. 2018.

[244] M. Bouda, S. Oda, O. Vassilieva, M. Miyabe, S. Yoshida, T. Katagiri, Y. Aoki, T. Hoshida, and T. Ikeuchi, "Accurate prediction of quality of transmission based on a dynamically configurable optical impairment model," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 1, pp. A102–A109, Jan. 2018.

[245] D. Rafique, T. Szyrkowiec, H. Grieer, A. Autenrieth, and J. P. Elbers, "Cognitive assurance architecture for optical network fault management," *Journal of Lightwave Technology*, vol. 36, no. 7, pp. 1443–1450, April. 2018.

[246] T. Panayiotou, S. P. Chatzis, and G. Ellinas, "Leveraging statistical machine learning to address failure localization in optical networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 3, pp. 162–173, March. 2018.

[247] D. R. B. de Araujo, C. J. A. Bastos-filho, and J. F. Martins-filho, "Methodology to obtain a fast and accurate estimator for blocking probability of optical networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 7, no. 5, pp. 380–391, May. 2015.

[248] G. Zervas, K. Banias, B. R. Rofoee, N. Amaya, and D. Simeonidou, "Multi-core, multi-band and multi-dimensional cognitive optical networks: An architecture on demand approach," in *Proc. IEEE ICTON'12, Coventry, UK*, July. 2012, pp. 1–4.

[249] J. A. Guerrero-ibanez, S. Zeadally, and J. Contreras-Castillo, "Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and internet of things technologies," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 122–128, Dec. 2015.

[250] L. Hu, M. Qiu, J. Song, M. S. Hossain, and A. Ghoneim, "Software defined healthcare networks," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 67–75, Dec. 2015.

[251] A. Samanta, S. Bera, and S. Misra, "Link-quality-aware resource allocation with load balance in wireless body area networks," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–8, 2015.

[252] Y. Kim and Y. Lee, "Automatic generation of social relationships between internet of things in smart home using SDN-based home cloud," in *Proc. IEEE WAINA'15, Gwangiu, South Korea*, March. 2015, pp. 662–667.

[253] S. Bera, S. Misra, and J. J. P. C. Rodrigues, "Cloud computing applications for smart grid: A survey," *IEEE Trans. Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1477–1494, May. 2015.

[254] J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, and M. Sha, "An internet of things framework for smart energy in buildings: Designs, prototype, and experiments," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 527–537, Dec. 2015.

[255] N. Bizanis and F. A. Kuipers, "SDN and virtualization solutions for the internet of things: A survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[256] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[257] Y. Huang, X. Ma, X. Fan, J. Liu, and W. Gong, "When deep learning meets edge computing," in *Proc. IEEE ICNP'17, Toronto, ON, Canada*, Oct. 2017, pp. 1–2.

[258] S. B. Calo, M. Touna, D. C. Verma, and A. Cullen, "Edge computing architecture for applying AI to IoT," in *Proc. IEEE BigData'17, Boston, MA, USA*, Dec. 2017, pp. 3012–3016.

[259] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, Jan. 2018.

[260] J. Schneible and A. Lu, "Anomaly detection on the edge," in *Proc. IEEE MILCOM'17, Baltimore, MD, USA*, Oct. 2017, pp. 678–682.

[261] D. Zissis, "Intelligent security on the edge of the cloud," in *Proc. IEEE ICE/ITMC'17, Funchal, Portugal*, June. 2017, pp. 1066–1070.

[262] A. Abeshu and N. Chilamkurti, "Deep learning: The frontier for distributed attack detection in fog-to-things computing," *IEEE Commun. Magazine*, vol. 56, no. 2, pp. 169–175, Feb. 2018.

[263] Z. He, D. Zhang, and J. Liang, "Cost-efficient heterogeneous data transmission in software defined vehicular networks," in *Proc. IEEE HPCC-CSS-ICESS'15, New York, NY, USA*, Aug. 2015, pp. 666–671.

[264] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, "Towards software-defined VANET: Architecture and services," in *Proc. IEEE MedHocNet'14, Piran, Slovenia*, June. 2014, pp. 103–110.

[265] K. Liu, L. Feng, P. Dai, V. C. S. Lee, S. H. Son, and J. Cao, "Coding-assisted broadcast scheduling via memetic computing in SDN-based vehicular networks," *IEEE Trans. Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–12, 2017.

[266] C. Wu, S. Ohzahata, and T. Kato, "Flexible, portable, and practicable solution for routing in VANETs: A fuzzy constraint Q-learning approach," *IEEE Trans. Vehicular Technology*, vol. 62, no. 9, pp. 4251–4263, Nov. 2013.

[267] ——, "Routing in VANETs: A fuzzy constraint Q-learning approach," in *Proc. IEEE GLOBECOM'12, Anaheim, CA, USA*, Dec. 2012, pp. 195–200.

[268] N. Kumar, S. Misra, and M. S. Obaidat, "Collaborative learning automata-based routing for rescue operations in dense urban regions using vehicular sensor networks," *IEEE Systems Journal*, vol. 9, no. 3, pp. 1081–1090, Sept. 2015.

[269] W. K. Lai, M.-T. Lin, and Y.-H. Yang, "A machine learning system for routing decision-making in urban vehicular ad hoc networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 3, pp. 374–391, 2015.

[270] F. Teymoori, H. Nabizadeh, and F. Teymoori, "A new approach in position-based routing protocol using learning automata for VANETs in city scenario," *arXiv preprint arXiv:1308.0099*, 2013.

[271] R. Li, F. Li, X. Li, and Y. Wang, "QGrid: Q-learning based routing protocol for vehicular ad hoc networks," in *Proc. IEEE IPCCC'14, Austin, TX, USA*, Dec. 2014, pp. 1–8.

[272] Z. He, J. Cao, and X. Liu, "SDVN: Enabling rapid network innovation for heterogeneous vehicular communication," *IEEE Network*, vol. 30, no. 4, pp. 10–15, July. 2016.

[273] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June. 2014.

[274] R. Trivisonno, R. Guerzoni, I. Vaishnavi, and D. Soldani, "SDN-based 5G mobile networks: Architecture, functions, procedures and backward compatibility," *Trans. Emerging Telecommunications Technologies*, vol. 26, no. 1, pp. 82–92, 2015.

[275] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[276] B. T. de Oliveira and C. B. Margi, "Distributed control plane architecture for software-defined wireless sensor networks," in *Proc. IEEE ISCE'16, Sao Paulo, Brazil*, Sept. 2016, pp. 85–86.

[277] H. Ahmadi and R. Bouallegue, "Exploiting machine learning strategies and RSSI for localization in wireless sensor networks: A survey," in *Proc. IEEE IWCMC'17, Valencia, Spain*, June. 2017, pp. 1150–1154.

[278] Z. Yu and J. J. P. Tsai, "A framework of machine learning based intrusion detection for wireless sensor networks," in *Proc. IEEE SUTC'08, Taichung, Taiwan*, June. 2008, pp. 272–279.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2866942, IEEE Communications Surveys & Tutorials

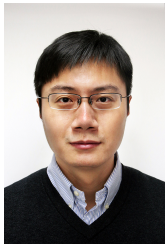
Junfeng Xie received his B.S. degree in communication engineering from University of Science and Technology Beijing, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. From September 2017 to September 2018, he visited Carleton University, Ottawa, ON, Canada, as a visiting Ph.D. student. His current research interests include machine learning, content delivery network, software defined networking, and blockchain.

F. Richard Yu (S'00-M'04-SM'08-F'18) received the PhD degree in electrical engineering from the University of British Columbia (UBC) in 2003. From 2002 to 2006, he was with Ericsson (in Lund, Sweden) and a start-up in California, USA. He joined Carleton University in 2007, where he is currently a Professor. He received the IEEE Outstanding Service Award in 2016, IEEE Outstanding Leadership Award in 2013, Carleton Research Achievement Award in 2012, the Ontario Early Researcher Award (formerly Premiers Research Excellence Award) in 2011, the Excellent Contribution Award at IEEE/IFIP TrustCom 2010, the Leadership Opportunity Fund Award from Canada Foundation of Innovation in 2009 and the Best Paper Awards at IEEE ICNC 2018, VTC 2017 Spring, ICC 2014, Globecom 2012, IEEE/IFIP TrustCom 2009 and Int'l Conference on Networking 2005. His research interests include wireless cyber-physical systems, connected/autonomous vehicles, security, distributed ledger technology, and deep learning.

He serves on the editorial boards of several journals, including Co-Editor-in-Chief for Ad Hoc & Sensor Wireless Networks, Lead Series Editor for IEEE Transactions on Vehicular Technology, IEEE Transactions on Green Communications and Networking, and IEEE Communications Surveys & Tutorials. He has served as the Technical Program Committee (TPC) Co-Chair of numerous conferences. Dr. Yu is a registered Professional Engineer in the province of Ontario, Canada, a Fellow of the Institution of Engineering and Technology (IET), and a Fellow of the IEEE. He is a Distinguished Lecturer, the Vice President (Membership), and an elected member of the Board of Governors (BoG) of the IEEE Vehicular Technology Society.

Tao Huang received his B.S. degree in communication engineering from Nankai University, Tianjin, China, in 2002, and his M.S. and Ph.D. degrees in communication and information systems from Beijing University of Posts and Telecommunications in 2004 and 2007, respectively. He is currently a professor at Beijing University of Posts and Telecommunications. His current research interests include network architecture, machine learning and software defined networking.

Renchao Xie received his Ph.D. degree from the School of Information and Communication Engineering, BUPT, in 2012. From July 2012 to September 2014, he worked as a postdoctoral researcher at China Unicom. From November 2010 to November 2011, he visited Carleton University as a visiting scholar. He is an associate professor at BUPT. His current research interests include content delivery network, machine learning, software defined networking, and 5G networks. He has published more than 30 journal and conference papers. He has served on the Technical Program Committees (TPCs) of Chinacom 2016 and the 2012 IEEE Vehicular Technology Conference (VTC)-Spring. He has also served for several journals and conferences as a reviewer, including IEEE Transactions on Communications, ACM/Springer Wireless Networks, the EURASIP Journal on Wireless Communications and Networking, (Wiley) Wireless Communications and Mobile Computing, IEEE Communications Letters, 2011 IEEE GLOBECOM, and so on.



Jiang Liu received his B.S. degree in electronics engineering from Beijing Institute of Technology, China, in 2005, his M.S. degree in communication and information systems from Zhengzhou University, China, in 2009, and his Ph.D. degree from BUPT in 2012. He is currently an associate professor at BUPT. His current research interests include network architecture, network virtualization, machine learning, software defined networking, and tools and platforms for networking research and teaching.

Chenmeng Wang received the M.S. degree in information and telecommunication engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2014, where he is currently pursuing the Ph.D. degree with the School of Communication and Information Engineering. From 2015 to 2017, he was a visiting student with Carleton University, Ottawa, ON, Canada. His current research interests include small cell networks, mobile edge computing systems, resource allocation, and applications of convex optimization in mobile networks.

Yunjie Liu received his B.S. degree in technical physics from Peking University, Beijing, China, in 1968. He is currently the Academician of the China Academy of Engineering, Chief of the Science and Technology Committee of China Unicom, and Dean of the School of Information and Communication Engineering, BUPT. His research interests include next generation networks, and network architecture and management.