Journal of Network and Systems Management (2021) 29:18
https://doi.org/10.1007/s10922-020-09583-4

Machine Learning-Based Multipath Routing for Software Defined Networks

Mohamad Khattar Awad[1] · **Marwa Hassan Hafez Ahmed**[1] · **Ali F. Almutairi**[2] · **Imtiaz Ahmad**[1]

Received: 27 April 2020 / Revised: 12 December 2020 / Accepted: 14 December 2020 /
Published online: 20 January 2021

Abstract

Network softwarization has recently been enabled via the software-defined networking (SDN) paradigm, which separates the data plane from control plane allowing for a flexible and centralized control of networks. This separation facilitates implementation of machine learning techniques for network management and optimization. In this work, a machine learning-based multipath routing (MLMR) framework is proposed for software-defined networks with quality-of-service (QoS) constraints and flow rules space constraints. The QoS-aware multipath routing problem in SDN is modeled as multicommodity network flow problem with side constraints, that is known to be NP-hard. The proposed framework utilizes network status estimates, and their corresponding routing configurations available at the network central controller to learn a mapping function between them. Once the mapping function is learned, it is applied on live-inputs of network status and routing requests to predict a multipath routing solutions in real-time. Performance evaluations of the MLMR framework on real traces of network traffic verify its accuracy and resilience to noise in training data. Furthermore, the MLMR framework demonstrates more than 98.99% improvement in computational efficiency.

Keywords Machine learning · Software defined networks · Software defined networking · Routing

✉ Mohamad Khattar Awad
mohamad@ieee.org

[1] Department of Computer Engineering, College of Engineering and Petroleum, Kuwait University, Kuwait, Kuwait

[2] Department of Electrical Engineering, College of Engineering and Petroleum, Kuwait University, Kuwait, Kuwait

1 Introduction

The network traffic has exponentially been growing due to rapid growth of information technology applications and communication industry such as cloud services, social media, smart phones, Internet of Things (IoT) and Internet applications. Cisco predicts a three-fold increase in IP traffic from 2017 to 2022 [1]. The traditional traffic routing and traffic engineering solutions should be revisited to accommodate, in real-time, this explosive increase in traffic. However, the rigid architecture of legacy networks does not permit for innovative network performance optimization due to the tight coupling between the data plane and control plane. Alternatively, software defined networking (SDN) has recently emerged as network architecture that enables dynamic network management and programmability, and hence performance optimization. This is achieved through decoupling the data forwarding plane and network control plane and shifting the control plane to a centralized controller, which oversees a network of simple data forwarding elements [2]. SDN facilitates building more innovative programmable network control and routing solutions based on a global view of the network status, and fine granular control of network traffic and network resources.

The recent advances in deep learning algorithms and recent emergence of SDN, have bridged the gap between machine learning (ML) and networking. Particularly, the centralized logical control shifts intelligence from the infrastructure layer to the control layer where efficient computing technologies like graphics processing unit (GPU) can be employed, and hence, provide the necessary computation power for machine intelligence. In addition, the packets inspection capabilities of forwarding elements and global view of the network status available at the central controller (CC) facilitate real-time training of data-driven ML algorithms. Moreover, the network programmability and flexibility in installing new packets forwarding rules enables real-time execution of routing results computed by ML algorithms [3]. Latah et al. [4] and Xie et al. [3] provide a comprehensive survey on the literature concerning ML algorithms and AI algorithms applied to SDN, and highlight how such algorithms can revolutionize traffic engineering and routing in SDN networks.

Several recent efforts have focused on applying ML techniques towards network traffic routing. Yanjun et al. [5] developed a framework that consists of a meta-layer of multiple ML modules for traffic routing in SDN. Each ML module implements a supervised learning algorithm to solve the routing problem between a pair of nodes in the network. The modules are trained based on a labeled data set of the network topology, QoS requirements, and optimal integral paths generated by a heuristic algorithm. Unlike multipath routing, integral routing does not allow splitting traffic on multiple paths and routes each flow on a single path. The trained modules can give heuristic-like routing solutions in real-time for every pair of nodes. The work in [6] proposed a dynamic routing framework for SDN called NeuroRoute. The framework predicts the future network traffic matrix and calculates an integral routing solution by implementing a heuristic algorithm. The predicted traffic matrices and obtained solutions constitute a training data set, that

is used to train a DNN. The trained neural network generates a heuristic-like integral routing solutions in real-time given a predicted traffic matrix. Unlike works in [5] and [6] which implement supervised learning based on a labeled data set, works [7–9] employ recursive learning to discover the routing solution that leads to the optimal network performance. In [7], a cognitive routing engine is proposed to find the optimal integral routing solution for overlay network of data centers. The work in [8] presented a QoS-aware routing method for multi-layer SDN networks. The method computes integral routing paths that maximize the network QoS performance for a given traffic type and user's applications. In [9], a deep reinforcement learning model is applied to compute integral routes that minimize the network delay for a given traffic matrix. These works foster ML-based routing; however, they overlook the negative impact of integral routing on traffic congestion [10].

The fine granular control of network traffic and network resources in SDN enables multipath routing, which is a key strategy for alleviating traffic congestion [11]. Several value-added services can be realized via flexible multipath routing in SDN [12]. First, end-to-end reliability assurance. This refers to switching traffic to alternate path in case of link failure or security threat. Second, service customization to applications QoS requirements. For instance, online gaming traffic can be routed on low-delay paths; whereas, file-sharing traffic can be routed on high throughput paths. Third, congestion avoidance and load balancing. This is maintained by routing traffic on multiple less congested paths instead of the shortest path. Furthermore, multipath routing has been shown effective in network energy conservation and congestion minimization [13]. Despite the importance of these services, multipath routing is limited by the size of the ternary content addressable memory (TCAM) at SDN forwarding elements. The routing of flows in SDN requires storing rules in TCAMs at forwarding elements, i.e., switches, along the routing path [14]. Then, multipath routing exacerbates the size limitation of TCAMs because the number of rules stored increases with the increase in the number of routing paths. Therefore, considering the routing rules space availability is critical in SDN.

The computation of optimal multipath routing decisions while considering multiple QoS constraints and TCAMs capacity constraints under stringent computation time requirements is not trivial. It involves solving an NP-hard optimization problem that does not scale well even for small networks [5, 15]. Although the problem can be relaxed and an approximate solution can be obtained by applying a heuristic extension of the column-generation approach as in [16], or fully polynomial time approximation schemes (FPTASs) [17], such approaches do not converge in real time and are not applicable in real networks [5, 6]. In this work, we close this research gap by introducing a computationally efficient routing framework that responds to the global network changes, and realizes QoS-constrained and TCAM size-constrained multipath routing in real-time. Leveraging recent advances in machine learning (ML), we develop a ML-based routing framework that predicts a heuristic-like routing solution based on estimates of the network status. We utilize the traffic matrices available at the CC and their corresponding sub-optimal heuristic routing solutions to train a feed-forward deep neural network (DNN). Given a global view of the network status, the trained DNN predicts a heuristic-like routing

solution on the fly. The contributions presented in this paper can be summarized as follows:

- The centralized multipath routing problem for SDN is modeled as a multi-commodity network flow problem (MCNF) with side constraints and capacitated links. The model captures traffic flows routing requirements in addition to traffic delay, link utilization, and forwarding rules space availability.
- A ML-based multipath routing (MLMR) framework is designed to realize heuristic-like and real-time multipath routing in software-defined networks with capacitated links, QoS performance constraints and TCAMs capacity constraints. The proposed framework comprises a ML-based routing module and other modules that support it. Adopting an empirical approach, the design and precision analysis of the routing module based on real traffic matrices are provided.
- Detailed performance evaluations are presented to demonstrate efficacy of proposed framework in comparison to a heuristic algorithm. The performance evaluations are comprehensive and implemented on real traffic matrices sampled at 5 min for nine weeks. The performance over the nine weeks is analyzed in relation to the traffic characteristics.

The remainder of this paper is organized as follows: In Sect. 2, the literature of related works is reviewed. Sect. 3 provides the problem formulation of the multipath routing problem in SDN as minimum cost multi-commodity network flow optimization problem with side constraints. The proposed MLMR framework architectural design is presented in Sect. 4. The implementation details of the proposed framework and performance evaluation results are presented and analyzed in Sect. 5. Finally, conclusions are drawn in Sect. 6.

2 Literature Review

The application of ML techniques in network traffic routing dates back to late 1980s. The use of neural network computational algorithm to determine the optimal traffic routes was first introduced by Rauch et al. in [18]. An extension of the neural network traveling salesman algorithm was implemented to minimize the network delay. A year later, Ouyang et al. [19] developed a shortest path algorithm based on the classical Hopfield type of neural networks, which is a parallel distributed processing system. The algorithm determines the optimal path from the source to a single destination, while considering link capacities. Ali et al. [20] proposed an improved version of the Hopfield neural network-based algorithm to support routing in packet switched networks. The same algorithm was also extended by Park et al. [21] to determine the optimal shortest path from a source to multiple destinations in large scale networks. In [22], Xia et al. presented a discrete time recurrent neural network for solving the shortest path problem. The lack of computational resources, appropriate learning algorithms, and training data in the ’80s challenged the efficiency and scalability of conventional neural networks in traffic routing [23].

The shortcomings of conventional neural networks have discouraged researchers from further employment of conventional neural networks in networking; however, the recent development of deep learning algorithms, availability of faster hardware, accessibility to a computationally powerful open-source cloud services, have enabled a resurgence of interest in applying ML techniques for networks management and traffic control. A summary of the basic workflow of applying ML techniques in networking was presented in [24]. Fadlullah et al. [25] provided an overview of deep learning applications for various network traffic control aspects. Similarly, Boutaba et al. [26] presented a comprehensive survey on evolution, applications and research opportunities of ML for networking. The literature works on unsupervised learning techniques for networking, which is mainly used for classification and prediction to support and enhance the network traffic control were surveyed in [27]. Choudhury et al. [28] describe two applications of ML in not only Internet protocol but also optical networks. In the first application, ML is applied for short-term and long-term prediction of traffic flows. However, in the second application, ML is applied for predicting optical path performance in multi-vendor network. ML is also applied in [29] and [30] for traffic prediction in elastic optical networks and mobile metro-core network, respectively. Mao et al. [31] discussed new opportunities in processing traffic packets with deep learning. In particular they envisioned a supervised deep learning system to construct routing tables of programmable routers. Similarly, a supervised deep learning neural network to improve traffic routing was proposed for programmable routers in [23]. Performance evaluations demonstrated a promising improvements in comparison to the open shortest path first (OSPF) protocol in terms of signaling overhead, network delay and throughput. A supervised learning-based routing framework for sensor networks was proposed in [32]. The framework gathers data, extracts features for offline learning, and performs online classification for routing decisions. Martin et al. [33] investigated the feasibility of applying supervised learning to solve the well-known routing and wavelength assignment (RWA) problem in optical networks. Tang et al. [34] proposed an unsupervised learning-based routing method that employs a deep convolutional neural network (CNN) for wireless mesh networks. The CNN is trained based on real-time data collected from the network, and hence does not require existing data set. The method exhibits intelligent learning for small-scale wireless networks with fixed CNN architecture; however, the convergence rate and scalability of the method remain open challenge for dynamic CNN.

Supervised learning-based schemes learn the mapping between network configurations and routing solutions computed by heuristics or FPTASs. Therefore, the optimality of the routing solution computed by supervised learning-based approaches is limited to the optimality of routing solutions used for training the DNN. Several recent efforts have been focusing on leveraging reinforcement learning to exploit a network performance beyond the level achieved by heuristics or FPTASs. Yu et al. [35] apply the Deep Deterministic Policy Gradients (DDPG) mechanism to optimize routing in SDN. DDPG is an actor-critic algorithm that utilizes the benefits of both Q-learning and policy gradients. In [36], Fang et al. apply a combination of reinforcement learning and neural networks towards routing via replacing Q-tables by neural networks to avoid manual extraction of

network features. In [37], Rischke et al. develop a tabular reinforcement learning-based scheme that represents the routing paths of flows in its state-action space. The scheme implements multipath routing; however, the state-action space grows exponentially with the number of forwarding nodes in the network. The reinforcement learning-based method have demonstrated performance superiority over traditional methods; however, they are computationally expensive and slow to converge. Furthermore, none of the above mentioned schemes consider multipath routing while guaranteeing QoS and forwarding rules space constraints.

3 Problem Formulation

The centralized control of the SDN architecture and availability of global view the network status at the CC facilitate development of centralized routing mechanisms. Furthermore, the programmability of network devices that is enabled by the SDN architecture allows the CC to install the routing solutions as rules, i.e., instructions, in the forwarding tables of the network forwarding elements. Therefore, the centralized routing problem can be formulated as a combinatorial multi-commodity network flow (MCNF) with side constraints [16].

We consider a software-defined network of N SDN forwarding elements and L directed links connecting them. The forwarding elements form the set $\mathcal{N} = \{1, \dots, n, \dots, N\}$ and links form the set $\mathcal{L} = \{1, \dots, l, \dots, L\}$. Each link has a traffic rate capacity denoted by g_l. A set of F traffic flows denoted by $\mathcal{F} = \{1, \dots, f, \dots, F\}$ are to be routed in the network. Each flow $f \in \mathcal{F}$ has a single source $s(f) \in \mathcal{N}$, destination node $e(f) \in \mathcal{N}$, and traffic demand d_f, $s(f) \neq e(f)$. The set of flows from all sources to all destinations is represented by traffic matrix $\mathbf{D}$ of size $N \times N$, where rows are N sources and columns are N destinations. For instance, element $[\mathbf{D}]_{i,j} = d_f$ denotes the traffic demand of flow f from node i to node j. A flow can be routed on a single path or multiple paths out of the set of possible paths $\mathcal{P}_f = \{p : \text{ chain of links connecting } s(f) \text{ to } e(f)\}$, where p denotes a specific path. The cost of transmitting one unit of flow f on link l is c_{fl}. Let δ_{pl} be a binary variable that takes a value 1 if $p \in \mathcal{P}_f$ traverse link $l \in \mathcal{L}$ and zero otherwise, i.e.,

$$\delta_{pl} = \begin{cases} 1, & \text{if } p \in \mathcal{P}_f \text{ traverses link } l \\ 0, & \text{otherwise} \end{cases}, \qquad \forall l \in \mathcal{L}, \forall p \in \mathcal{P}_f, \forall f \in \mathcal{F}. \tag{1}$$

Therefore, the cost of routing traffic through path $p \in \mathcal{P}_f$ is given by

$$\eta_p = \sum_{l \in \mathcal{L}} \delta_{pl} c_{fl}. \tag{2}$$

Let θ_p be the fraction of flow f's traffic demand d_f routed on path p, $\forall p \in \mathcal{P}_f$, satisfying the conditions $0 \leq \theta_p \leq 1$ and $\sum_{p \in \mathcal{P}_f} \theta_p = 1$, $\forall f \in \mathcal{F}$. Then, we can write the following traffic constraint,

$$\sum_{p \in \mathcal{P}_f} \theta_p d_f = d_f, \qquad \forall f \in \mathcal{F}, \tag{3}$$

where $\theta_p d_f$ is the amount of traffic routed on path p, $\forall p \in \mathcal{P}_f$. Furthermore, define a binary variable ξ_p that takes a value of 1 if path p is used to route flow f, $\forall p \in \mathcal{P}_f$ and $f \in \mathcal{F}$. This can be written as,

$$\xi_p = \begin{cases} 1, & \text{if } \theta_p > 0 \\ 0, & \text{otherwise} \end{cases}, \qquad \forall p \in \mathcal{P}_f, \forall f \in \mathcal{F}. \tag{4}$$

The traffic flows in current software-defined networks impose tight bounds on performance indicators in order to maintain stringent QoS requirements of emerging network services. We introduce a weight coefficient ω_{fl} to capture the number of flow rules installed in order to routing flow f on link l, and the maximum number of flow rules that can be used to route flow f on path p is given by u_f. Then, the following constraint has to be met,

$$\sum_{l \in \mathcal{L}} \delta_{pl} \omega_{fl} \xi_p \leq u_f, \qquad \forall p \in \mathcal{P}_f, \forall f \in \mathcal{F}. \tag{5}$$

Notice that ω_{fl} is independent of the amount of traffic demand d_f routed on a link l; whereas c_{fl} depends on the amount of traffic routed on a link l. The routing cost c_{fl} represents a rate-dependent routing performance metric like jitter, delay, load balance, congestion probability, packet, and reliability. However, the weight coefficient ω_{fl} models a rate-independent performance metric. In OpenFlow, which is one of the most commonly used SDN standards, forwarding rules are stored in ternary content-addressable memory (TCAM) at SDN switches [12, 38]. Therefore, the performance improvement gained by multipath routing must be well balanced with TCAM size in order to maintain scalability of the SDN network.

Given these definitions, the considered constrained routing problem (**CRP**) in SDN can be formulated as a MCNF problem with side constraints. The CRP is formulated as follows.

$$\text{CRP: } \min_{\theta_p, \delta_{pl}, \xi_p} \sum_{f \in \mathcal{F}} \sum_{p \in \mathcal{P}_f} \theta_p d_f \eta_p \tag{6}$$

$$\text{s.t.} \sum_{f \in \mathcal{F}} \sum_{p \in \mathcal{P}_f} \delta_{pl} \theta_p d_f \leq g_l, \ \forall l \in \mathcal{L} \tag{7}$$

$$\sum_{p \in \mathcal{P}_f} \theta_p d_f = d_f, \qquad \forall f \in \mathcal{F} \tag{8}$$

$$\sum_{l \in \mathcal{L}} \delta_{pl} \omega_{fl} \xi_p \leq u_f, \qquad \forall p \in \mathcal{P}_f, \forall f \in \mathcal{F} \tag{9}$$

$$\theta_p \leq \xi_p, \qquad \forall p \in \mathcal{P}_f, \forall f \in \mathcal{F} \tag{10}$$

$$0 \leq \theta_p \leq 1, \qquad \forall p \in \mathcal{P}_f, \forall f \in \mathcal{F} \tag{11}$$

$$\xi_p \in \{0, 1\}, \qquad \forall p \in \mathcal{P}_f, \forall f \in \mathcal{F} \tag{12}$$

$$c_{fl} \geq 0,\ \omega_{fl} \geq 0, \qquad \forall f \in \mathcal{F}, \forall l \in \mathcal{L}. \tag{13}$$

Constraint (7) guarantees that the total traffic rate routed on a link does not exceed link capacity. Constraint (8) ensures routing the whole traffic demand of all flows. Constraint (9) restricts the number of flow rules installed to route flow f on path p to bounded by u_f. Therefore, constraint (10) allows multi-path routing on only eligible paths. The problem formulation is flexible and allows variation in requirements from one flow to another; hence, different types of services with diverse QoS requirements can be supported.

It was shown in [16] that the **CRP** is NP-hard. We pose the MCNF with side constraints and capacitated links as a multi-output regression problem. We take advantage of the availability of large volume of the network states, i.e., traffic matrices, at the CC to construct a labeled training dataset. The corresponding solutions are computed off-line by a heuristic algorithm. The dataset is utilized by a supervised learning technique to train a feed-forward deep neural network (DNN). The trained model generates heuristic-like routing solutions in real-time.

4 Machine Learning-based Multipath Routing Framework

In this paper, we give a detailed description of the proposed MLMR framework. The main objective of the MLMR framework is to realize heuristic-like and real-time multipath routing in software-defined networks with capacitated links and side performance constraints. Although the MLMR framework replaces traditional algorithms that solve the MCNF problem with side constraints, it depends on such algorithms to build a labeled dataset for training the DNN. Therefore, the optimality of the solution predicted by the MLMR framework is comparable to the solution computed by the heuristic method. However, the MLMR framework is significantly more computationally efficient than traditional algorithms.

The MLMR framework can be implemented at the application layer of the SDN architecture. The northbound and southbound APIs are assumed to be in place to facilitate exchange of information between the proposed framework, CC, and forwarding elements. The forwarding elements at the infrastructure layer report traffic matrices to the controller, which depends on the proposed MLMR framework to compute the multipath routing solution. Then, the controller implements this routing solution in the forwarding tables of the forwarding elements. This process is repeated either when a packet of unrouted flow arrive at any of the forwarding elements, or a significant change in the network status is detected. Figure 1 depicts the

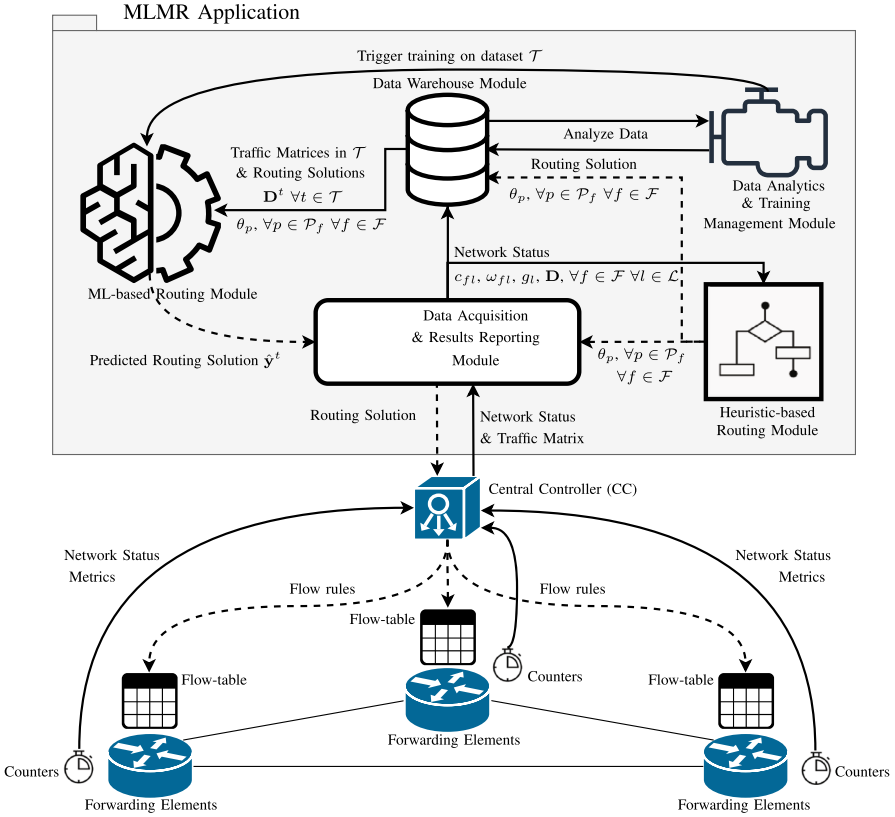

Fig. 1 Modular representation of the architecture of the proposed MLMR framework

architecture of the proposed MLMR framework, which consists of the following five modules:

- data acquisition and results reporting module;
- data analytics and training management module;
- data warehouse module;
- heuristic-based routing module;
- ML-based routing module.

The design of each of these modules is described below.

4.1 Data Acquisition and Results Reporting Module

The data acquisition and results reporting module collects the network states and traffic demands from all network forwarding elements through the southbound APIs. This module constructs an estimate of the network traffic matrix $\mathbf{D}$, links status (i.e. active or inactive), link capacity $g_l \ \forall l \in \mathcal{L}$, link weight coefficient ω_{fl},

$\forall f \in \mathcal{F}, \forall l \in \mathcal{L}$, and link cost c_{fl}, $\forall f \in \mathcal{F}, \forall l \in \mathcal{L}$. The statistics and counters available at the forwarding elements are major sources of these metrics. Some of them vary at a faster rate than other ones; therefore, the accuracy and validity of the estimates is limited to a short period of time. Hence, it is crucial to maintain time-stamped estimates of these metrics. These metrics are saved in data structures in the data warehouse module and forwarded to the heuristic-based routing module. The design of a network measurement schemes, which constructs a global view of the network status is beyond the scope of this work; however, several recent works have addressed this issue [39, 40]. Furthermore, this module reports routing solutions received from either the ML-based routing module or the heuristic-based routing module to the CC. During training of the DNN or during dataset generation, the updated ML-based routing solution would not be available; therefore, the framework returns the existing ML-based routing solution. Once the DNN is trained, the data acquisition and results reporting modules starts reporting the new ML-based solution to the CC.

4.2 Heuristic-Based Routing Module

Given a global view of the network and estimates of the traffic matrices, a heuristic algorithm can be employed to solve the MCNF problem with side constraints. In this work, we implement the column-generation-based algorithm presented in [16] to solve the problem. The algorithm computes a set of paths $\mathcal{P}_f$ $\forall f \in \mathcal{F}$ and fractions θ_p $\forall p \in \mathcal{P}_f$ of traffic demands d_f routed through these paths. In order to make the current paper self-contained, we include brief description of the heuristic algorithm in Appendix 1. The results are reported to the data warehouse and saved in the same data structure corresponding to the input data based on which the routing solution was computed. Moreover, these results are returned to the data acquisition and results reporting module, which forwards it to the CC. By the time these results are generated, the network status might have been changed which deteriorates the quality of such solutions. However, these results are applied temporarily only during initialization and training of the DNN.

4.3 Data Warehouse Module

This module maintains time stamped sequence of data structures holding the network status metrics, which are the traffic matrices, the set of routing paths of each flow, and the distribution of traffic on these paths. The data structures are periodically generated based on metrics reported from the infrastructure layer. The warehouse covers almost all possible network states under different network loads and conditions. The dataset is used to train the DNN in the ML-based routing module.

The dataset is implemented in a non-relational structured query language (NoSQL) database. Particularly, the collected traffic metrics are added to a JSON document that can be accessed by other modules in the framework. A model-driven service adaptation layer (MD-SAL) can be used in real implementations to provide

messaging and data storage functionality to various applications running on the CC [41].

4.4 Data Analytics and Training Management Module

Different data analytics techniques can be used in this module for traffic classification and clustering [42]. Two tasks are performed at this module in order to improve the quality and accuracy of computed routing solutions. First, the identification of the appropriate training dataset from the data warehouse to train the DNN of the routing module. Because the proposed framework stores traffic metrics in NoSQL database, it facilitates implementation of Big Data analytics to identify the training dataset for training the DNN [43]. The design of such Big Data analytics is beyond the scope of this work. It has been well studied in [43] and reference therein. Second, triggering the training process of the DNN when a significant change in the network status or traffic demands is detected. The module analyzes, online, the information being added to the data warehouse to detect dissimilarity in traffic matrices by utilizing statistical techniques like the principle component analysis (PCA), and the Hurst exponent estimation methods [44, 45]. The significance of change in network status is defined by the decrease of a moving average of Hurst exponents for a window of traffic measurements, below a predefined threshold of [46, 47]. In addition, the training process of the DNN can be triggered if a link failure is reported.

4.5 ML-Based Routing Module

The dataset stored in the data warehouse module holds a sequence of time stamped traffic matrices $\mathbf{D}$. We introduce a superscript t to denote the time index at which this traffic matrix is received. The size of the traffic matrix is $N \times N$. It can be reshaped into a vector of the size $1 \times N^2$ by stacking the transposes of the rows as follows,

$$\mathbf{x}^t = \begin{bmatrix} [\mathbf{D}]'_{1,\ 1\ldots N} \\ [\mathbf{D}]'_{2,\ 1\ldots N} \\ \vdots \\ [\mathbf{D}]'_{N,\ 1\ldots N} \end{bmatrix}, \tag{14}$$

where $[\cdot]'$ denotes a vector transpose.

The dataset also holds the routing solution computed by the hurestic-based routing module corresponding a specific traffic matrix. A routing solution comprises routing paths of each flow $\mathcal{P}_f$, and distribution of the traffic demand among these paths θ_p, $\forall p \in \mathcal{P}_f\ \forall f \in \mathcal{F}$. The maximum number of paths considered in routing each flow is denoted by K, i.e., the cardinality of a set of paths $|\mathcal{P}_f| \leq K$, $\forall f \in \mathcal{F}$. Recall from Sect. 3 that unused paths are assigned a fraction $\theta_p = 0$. Therefore, the sets of paths for all flows can be combined following the same order of flows $1, \ldots f, \ldots F$ and indexed by path ID $1, \ldots, F \times K$ [5]. Then, a vector of size $F \times K$ holding the traffic distribution fractions can be defined as follows,

$$\mathbf{y}^t = \begin{bmatrix} \theta_p \ \forall p \in \mathcal{P}_1 \\ \theta_p \ \forall p \in \mathcal{P}_2 \\ \vdots \\ \theta_p \ \forall p \in \mathcal{P}_F \end{bmatrix}. \tag{15}$$

Notice that the solution of the **CRP** consists of the flows distribution variables θ_p on paths $p \in \mathcal{P}_f, \forall f \in \mathcal{F}$. Because $\theta_p < \xi_p$, the flows are routed on only paths that satisfy the flow rules constraint. Therefore, listing the flows distribution factors on considered paths for each flow in the vector $\mathbf{y}^t$ gives a full representation of the **CRP** solution.

The ML-based routing module constructs a sequence of tuples of the vectors representing the traffic matrix $\mathbf{x}^t$ and vectors representing the routing solution $\mathbf{y}^t$, denoted by $\{(\mathbf{x}^t, \mathbf{y}^t)\}_{\forall t \in \mathcal{T}}$. Here $\mathcal{T}$ refers to the set of tuples in the data warehouse identified by data analytics and training management module to train the ML-based routing module. In supervised learning, the feed-forward DNN learns a function M that best maps input $\mathbf{x}^t$ to output $\mathbf{y}^t$ in such a way $M : \mathbf{x}^t \rightarrow \mathbf{y}^t$ based on a labeled training dataset $\{(\mathbf{x}^t, \mathbf{y}^t)\}_{\forall t \in \mathcal{T}}$ [48]. Once the mapping function M is learned, the ML-based routing module computes in real-time estimates of the routing solution denoted by $\hat{\mathbf{y}}^t$ for a new traffic matrix $\mathbf{x}^t$, $t \notin \mathcal{T}$.

The proposed ML-based routing module operates based on a feed-forward DNN shown in Fig. 2. The fractional values of the multi-path routing output $\hat{\mathbf{y}}^t$ make the deep learning structure a multi-output regression model. The DNN consists of fully connected layers out of which two are hidden, one is input layer representing $\mathbf{x}^t$, and one is output layer representing $\mathbf{y}^t$. A dropout out technique is applied in both hidden layers to enhance the model robustness against overfitting [49]. Furthermore, the adaptive moment estimation (Adam) optimization algorithm is adopted to train the deep learning network and minimize the prediction mean square error (MSE) [50]. We arrived at this DNN design after several trial and error experiments; details of these experiments

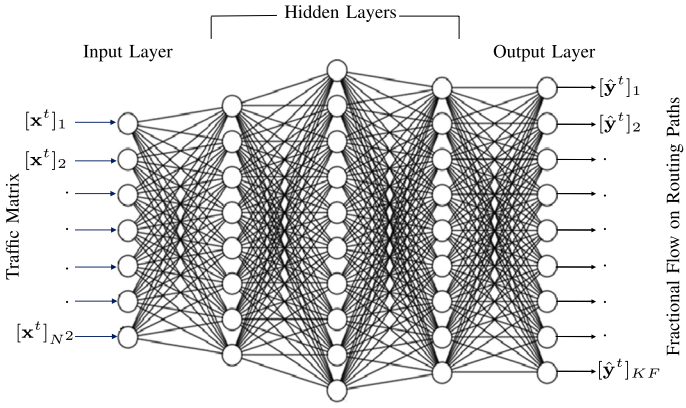

Fig. 2 Proposed feed-forward DNN. The number of input layers, hidden layers, and output layers depends on the number of forwarding elements in the network and number of paths considered for each flow

are presented in the following section. Various design versions were considered until the best performance is achieved in both training and testing evaluations.

Based on the above detailed description of the framework modules, we give an overview of the data flow and interaction among the different modules illustrated in Fig. 1. It is assumed that the SDN forwarding elements periodically report the network status and traffic matrices to the CC. During framework initialization, the data acquisition and results reporting module collects the network status and traffic matrices from the CC and forwards it to both the heuristic-based routing module and the data warehouse module. The heuristic-based module computes a heuristic-based routing solution and returns it to both data-acquisition-and-results-reporting module and the dataset warehouse. The solution is reported to the CC, which implements it as forwarding rules in forwarding elements. The flows are routed based on the heuristic-based solutions until sufficient number of networks traffic matrices and corresponding routing solutions are stored in the data warehouse. The records stored in the data warehouse are instantaneously analyzed by the data analytics and training management module, which decides when to trigger training of the DNN and which set, $\mathcal{T}$, of records is used for training. Once the DNN is trained, the ML-based routing module starts generating routing solutions and reports them to the data-acquisition-and-results-reporting module, which forwards them to CC. At this stage, the routing solutions received from the heuristic-based routing module are only used to update the data warehouse, and only ones received from the ML-based routing module are forwarded to the CC in real-time.

5 Implementation and Network Performance Evaluations

This section presents the implementation details of the proposed MLMR framework and evaluates its performance on real traffic matrices. Several design parameters, and settings of the training and testing environment have significant impact on the performance of ML-based designs. Particularly, the performance of the proposed MLMR framework highly depends on the performance of its constituting modules, and characteristics of the network in which it trains and operates. Therefore, before we present the network performance results of the MLMR framework, we describe the dataset resembling the the network settings considered in our performance evaluations, and precision analysis of the DNN implemented in the ML-based routing module.

The performance evaluations are conducted on a personal computer with a single-core i7-2.4 Ghz CPU and 8 GB RAM. The DNN of the ML-based routing module is implemented in Keras [51], which runs on top of TensorFlow platform for machine learning [52].

5.1 Dataset Generation

Despite the significant interest from both academia and industry in applying ML techniques towards optimization of networks performance, a standardized high-quality and widely accepted dataset remains unavailable for networking scenarios [24]. Therefore, we rely on a publicly available real traffic matrices to generate our

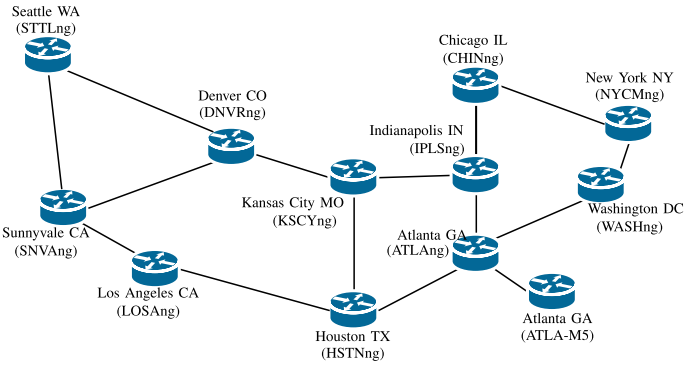

Fig. 3 Abeline topology

training and testing dataset [53]. The traffic matrices are available through the Toolbox for Traffic Engineering Methods (TOTEM) project web-page [54, 55]. The realistic traffic matrices are sufficiently large and captured at the well known Internet2 research backbone network, Abeline, shown in Fig. 3 [56]. The network consists of

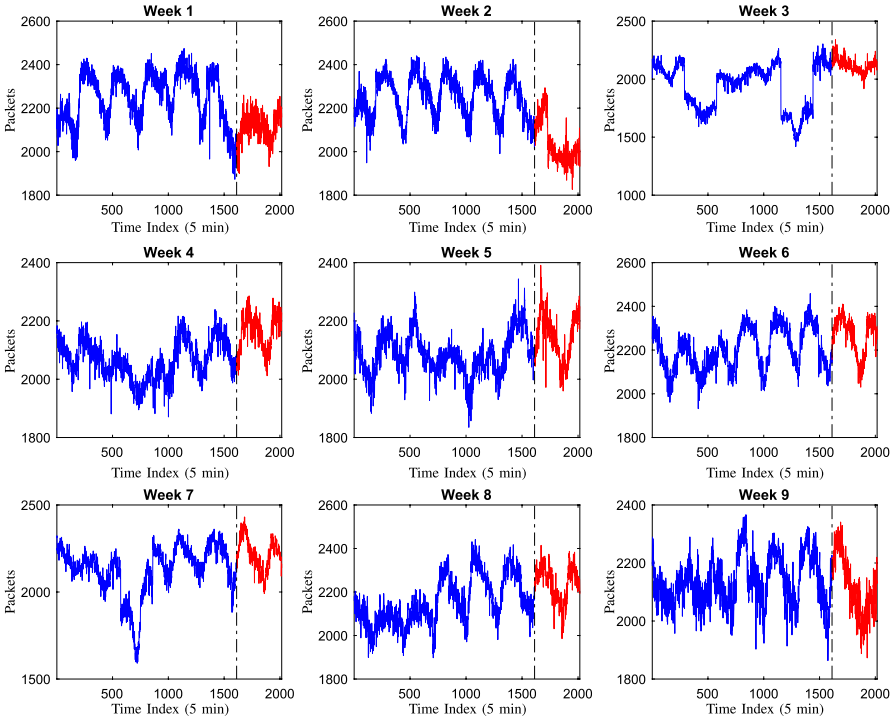

Fig. 4 Sum of network traffic on all links (100 Bytes Packets) versus indices of 5 min time samples captured over nine weeks. The samples used for training are marked blue whereas samples used for testing are marked red

Table 1 A sample of the routing paths the for Abilene network

Path ID	Source	Destination	Nodes on path
0	ATLA-M5	ATLAng	ATLA-M5-ATLAng
1	ATLA-M5	CHINng	ATLA-M5-ATLAng-IPLSng-CHINng
2	ATLA-M5	CHINng	ATLA-M5-ATLAng-WASHng-NYCMng-CHINng
3	ATLA-M5	CHINng	ATLA-M5-ATLAng-HSTNng-KSCYng-IPLSng-CHINng
4	ATLA-M5	DNVRng	ATLA-M5-ATLAng-HSTNng-KSCYng-DNVRng
⋮	⋮	⋮	⋮

Table 2 A sample routing solution for one of the traffic matrices

Traffic demand nodes			Routing solution	
Source	Destination	$\mathbf{x}^t$	Path ID	$\mathbf{y}^t$
ATLA-M5	ATLAng	125	0	1
ATLA-M5	CHINng	160	1	0.11
			2	0.89
⋮	⋮	⋮	⋮	⋮

12 nodes and 15 bidirectional links. All links are 10 Gbps; however, the Atlanta (ATLAng) to Indianpolis (IPLSng) is 2.5 Gbps. The traffic matrices contain samples of the network traffic captured at 1/(5 min) sampling rate for a period of 9 weeks. There are 2017 samples per week. Each sample contains the sources and the destinations of all active flows and their traffic demands. The sum of network traffic, i.e., $\sum_{i=1}^{N}\sum_{j=1}^{N}[\mathbf{D}]_{i,j}$, over the considered 9 weeks is shown in Fig. 4. We use the widely accepted ratio of splitting the dataset into training and testing, that is 80% for training (blue) and 20% for testing (red) [57]. The network load evolves in different patterns over the nine weeks; therefore, facilitates evaluation of the proposed framework under various network conditions.

Given the network topology, the set of possible paths connecting each source to each destination node is computed offline using Yen's K-shortest path algorithm, where K is set to three paths [58]. Assuming a maximum number of flows $F = N(N-1)$ from every source to all other destinations; then, the maximum number of paths is computed to be $KN(N-1)$. The paths are given IDs that is fixed for the entire network operation. A sample of the computed paths for the considered Abilene topology is tabulated in Table 1.

The traffic matrices are reshaped to match the input vector $\mathbf{x}^t$ of the ML-based routing module presented in Sect. 4. Recall that the superscript $(\cdot)^t$ refers to the time index of the data sample. The size of the traffic matrix $N \times N$ is reshaped to a vector of size $N(N-1)$ to construct the input vector $\mathbf{x}^t$. Notice that diagonal elements of the traffic matrix are eliminated since self-looping is not possible. Therefore, for the Abilene network which consists of 12 nodes, the size of the ML-based routing module input vector is $12 \times 11 = 132$. The multipath routing solutions of each of these traffic matrices is computed using the column-generation algorithm presented

[16]. The solution consists of the fractions θ_p of each flow $f \in \mathcal{F}$ that are distributed on the available paths $\forall p \in \mathcal{P}_f$, which are labeled with path IDs. The fractions are assigned to paths to construct the output vector $\mathbf{y}^t$. A sample of the input vector, output vector and data used to compute them is shown in Table 2. There are 2017 traffic samples per week; therefore, we have 18,153 tuples of $\mathbf{x}^t$ and $\mathbf{y}^t$, $\forall t = 1, \ldots, 18,153$. These tuples $\{(\mathbf{x}^t, \mathbf{y}^t)\}$ constitute our dataset. In the following sections, this dataset is utilized in evaluating the precision performance of the DNN and network performance of the proposed MLMR framework.

5.2 DNN Precision Analysis

The empirical approach has widely be adopted in designing the DNN architecture, which constitutes major part of ML-based routing frameworks [6, 31, 48]. We adopt the empirical approach towards the design of the DNN architecture. The architectural design includes setting the following parameters: number of nodes, number of layers, optimization algorithm, activation function, and dropout layers. Recall from Sect. 4 that we are interested in training the DNN to learn the mapping function between the inputs $\mathbf{x}^t$ and outputs $\mathbf{y}^t$ in the training set, i.e., $\forall t \in \mathcal{T}$, in order to estimate $\hat{\mathbf{y}}^t$ for a new $\mathbf{x}^t$ where $t \notin \mathcal{T}$. The prediction error, i.e., the difference between $\mathbf{y}^t$ and $\hat{\mathbf{y}}^t$, of the proposed design is analyzed in terms of MSE; whereas, the computational efficiency is evaluated by run time.

We have manually inspected the performance of various combinations of the design parameters when applied on the 9 weeks dataset. The 2017 tuples, i.e., $\{(\mathbf{x}^t, \mathbf{y}^t)\}$, of each week are divided into 1613 training samples (80% of the number of tuples per week) and 404 testing samples (20% of the number of tuples per week). The performance results varies sightly from one week to another; therefore, we summarize the results in box and whisker plots.

The DNN architecture consists of one input layer, one output layer and multiple hidden layers, and a number of hidden layers. The input layer is comprised of $N(N-1) = 132$ nodes and the output layer is comprised of 394 nodes. Notice that $KN(N-1) = 396$; however, the largest number of paths between a pair of adjacent nodes (ATLA-M5 & ATLAng) is limited to one. This reduces the number of nodes in the output layer to 394. Figure 5 depicts the MSE performance of different hidden layers for both training and testing samples. While the number of hidden layers is varied, the number of nodes in the first layer is 264 nodes, the number of nodes in second hidden layer is 300 nodes, which demonstrates the best performance as will be shown later. In this particular evaluation, the activation functions for the input and hidden layers are designed to be the rectified linear unit (ReLU) function; however, for the output layer, the activation function is chosen to be the Sigmoid function. We limit the number of epochs to 100 and compare the training time of the three DNN configurations as shown in Fig. 6. Notice that increasing epoch size may result in improving the prediction performance, but increases the training time significantly. We chose configuration with 2 hidden layer, which demonstrates the least MSE values and shortest training time. The median of the training time for the

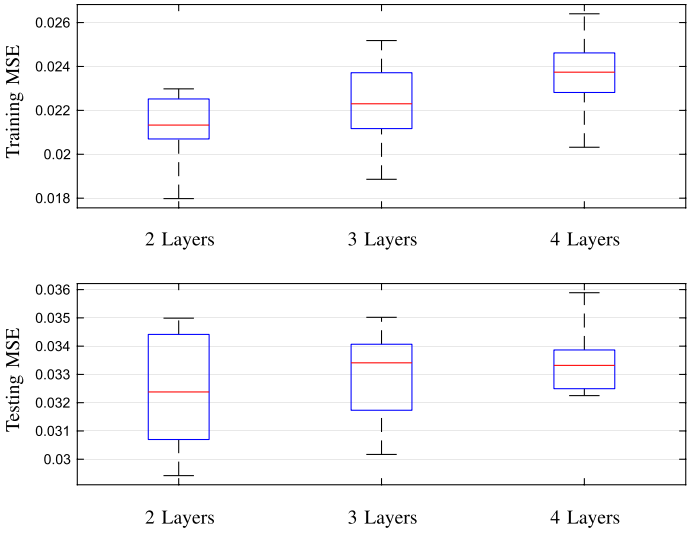

Fig. 5 MSE of DNN training (top) and testing (bottom) versus number of hidden layers

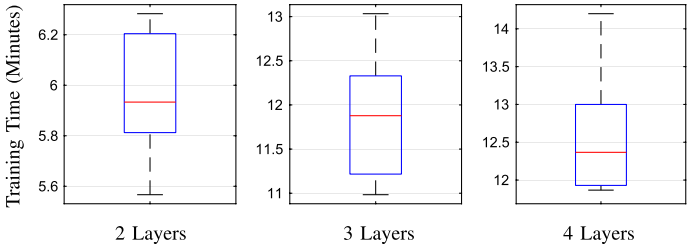

Fig. 6 Training time versus number of hidden layers

DNN with 2, 3, and 4 hidden layers is 5.9333 min, 11.8780 min and 12.3667 min, respectively.

The number of nodes of the hidden layers is designed based on the MSE results we obtained by trying different combinations of number of nodes [59]. All combinations fall between the number of nodes in the input layer and the output layer, which are 132 and 394, respectively. In fact, one of the numbers is 264, which is twice the number of nodes in the input layer. Figure 7 shows the MSE of the DNN testing for combinations of number of nodes in hidden layers. It is clear that the combination 264 in the first hidden layer and 300 in the second hidden layer demonstrates the best performance.

We evaluate the prediction accuracy of the four layers DNN when three different optimization algorithms are used to update the DNN parameters, i.e., neurons weights and bias, in order to minimize the MSE loss function during the training process. The three algorithms are variants of the gradient descent optimization algorithm. Theses optimization algorithms are adaptive moment estimation

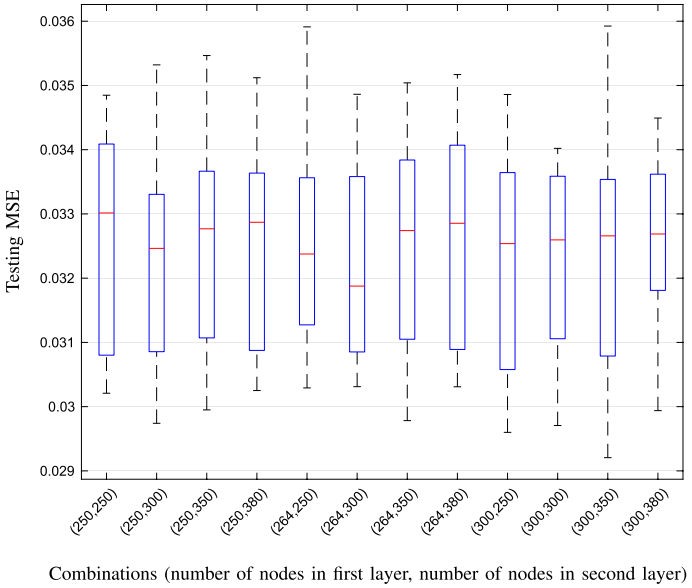

Fig. 7 MSE of DNN testing for combinations of number of nodes in the first and second hidden layer

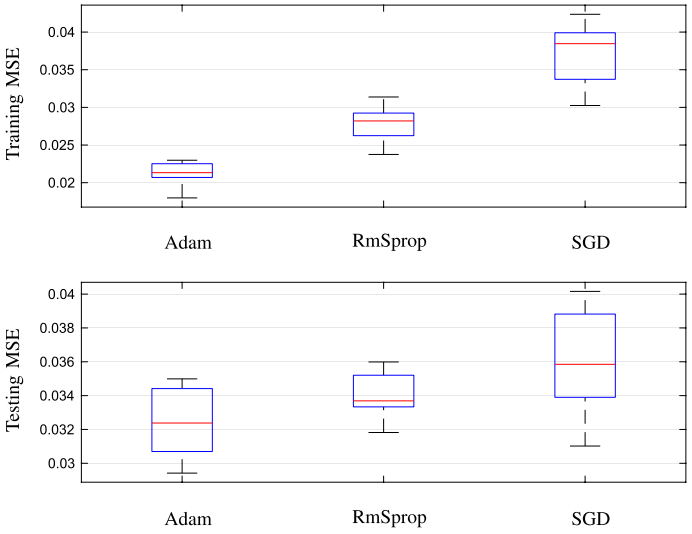

Fig. 8 MSE of DNN training and testing for three optimization algorithms

(Adam) [50], root mean square prop (RmSprop) [60] and the stochastic gradient descent (SGD) [59]. Figure 8 depicts the MSE of the DNN prediction for the three algorithms. It is clear that Adam outperform both RmSprop and SGD in

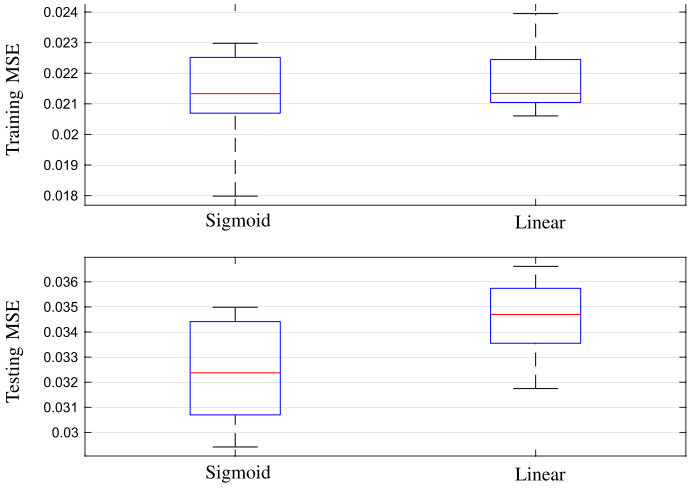

Fig. 9 A comparison of the performance of the DNN when one of two activation functions is used, Sigmoid or linear activation function

both training and testing. Thus, the DNN is configured to implement the Adam algorithm for all remaining evaluations.

In DNNs, the activation function defines the output of a node based on an input or multiple inputs from lower layers. The choice of activation function is problem dependent. Figure 9 shows the prediction accuracy of the DNN when the Sigmoid activation function is used in comparison to using a liner activation function. Although, both functions result in comparable prediction accuracy, the Sigmoid function demonstrates better training performance.

The online generated small set of real traffic matrices used for training the DNN might be subject to statistical noise. In such scenarios, the DNN learns the characteristics of the noise as part of mapping model and performs poorly when evaluated on new data. This problem is referred to as the "over fitting problem". The dropout technique is one of the commonly used regularization techniques for alleviating the over fitting problem. It drops out some randomly selected nodes to make layers look differently in each training epoch; hence, nodes parameters are updated differently. Figure 10 compares the prediction accuracy of the DNN when the dropout technique is used for both hidden layers and when it is not used. A clear drop in MSE of testing prediction accuracy can be observed; hence, the dropout technique is adopted for both hidden layers in the proposed architecture.

5.3 Network Performance

The precision analysis presented in the previous sections verifies the accuracy of the predicted multipath routing solutions computed by the ML-based routing module. However, it crucial to compare and contrast the performance of the proposed framework on the nine weeks datasets in order to gain deeper insights into the impact of various network traffic characteristics on the performance of the MLMR framework.

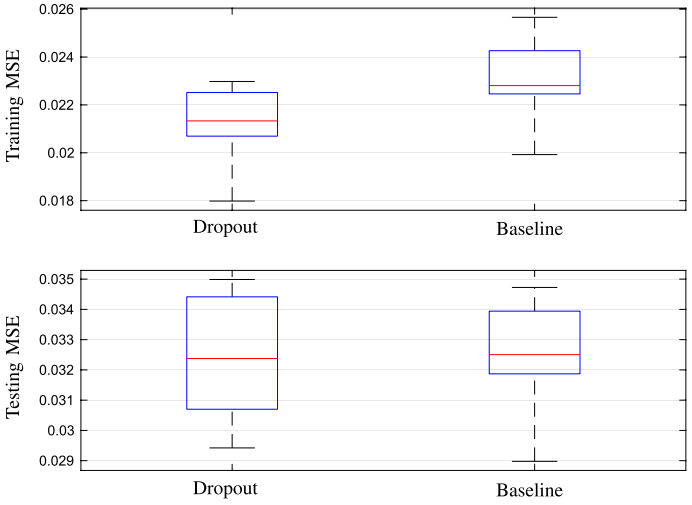

Fig. 10 The advantage of using dropout technique in hidden layers. The top subplot shows MSE of training and bottom shows MSE of testing

In this section, we identify major features of network traffic characteristics that drive the accuracy of the MLMR framework. Moreover, we evaluate the performance of the proposed MLMR framework in comparison to the performance of a heuristic column-generation-based algorithm (CGbA) presented in [16]. The cost of transmitting one unit of flow f on link l is set to $c_{fl} = 1$. The routing algorithms install a single forwarding rule per flow at each forwarding element on each of the routing paths; therefore, the $\omega_{fl} = 1$. Given the size of the network, it is reasonable to set $u_f = 5$ in our performance analysis.

The sum of the network traffic presented in Fig. 4 shows the general evolution behavior of the network traffic load. However, we are interested in a more detailed view of the network traffic behavior in order to analyze its impact on the precision of the DNN and the network performance of the MLMR framework. The traffic load on the 15 bidirectional links, i.e., 30 directional links, gives a more detailed view of the traffic distribution between the nodes. We apply the principle component analysis (PCA) in order to reduce the dimensionality of the matrix holding the links traffic for one week from a 2017 by 30 to a 2017 by 2 matrix [44]. This allows us to present the testing and training datasets in two dimensional plots. The first and second principle components with largest eigenvalues capture most features of the network traffic. The first two components of the data account for large percentage of the variance of each of the weeks dataset. It is important to mention that these components are used for data visualization and results presentation only. However, the training and testing is done on the full dimensional dataset.

The MSE of the objective function of the **CRP** computed based on predicted $\hat{\mathbf{y}}^t$ versus the objective function computed based on true $\mathbf{y}^t$ is given in Fig. 11. Due to randomness introduced by the dropout technique and dataset shuffling, the MLMR framework might perform differently in different runs. In order to get an accurate

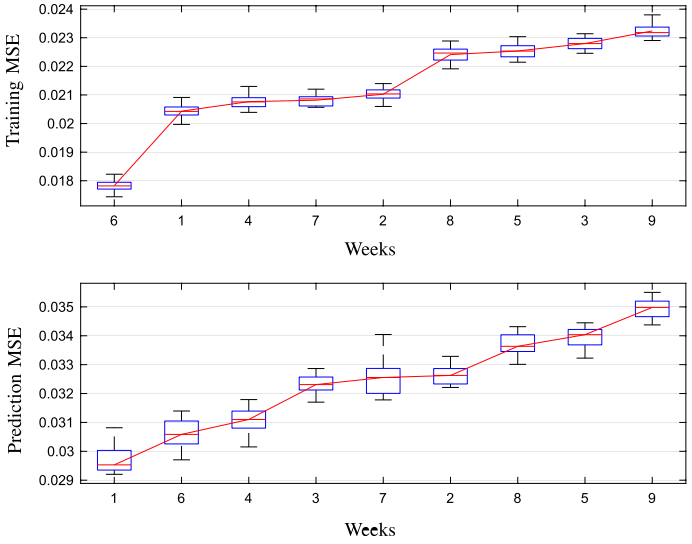

Fig. 11 The training (top), and prediction (bottom) MSE performance of the proposed MLMR framework versus CGbA over the nine weeks datasets. The box-plots summarize results obtained from ten runs. Results are sorted in ascending order of their median MSE values

representation of the framework performance, we deploy the framework ten times and summarize the results in box-plots presented in Fig. 11. The change in performance over the nine weeks can be explained by the change in characteristics of the network traffic over these weeks. Figure 12 depicts a scatter plot of PCA component-1 versus PCA component-2 of the network traffic on thirty directional links. A near circular scatter plot implies that the PCA components are uncorrelated; whereas, highly elliptical scatter plot implies high correlation [61]. As secondary measure of correlation, a counter clockwise tilt in the scatter plot indicates positive correlation and clockwise tilt indicates negative correlation. Also, both horizontal and vertical scatter plots indicate uncorrelation. It is important to mention here that simple measures of correlation, e.g., Pearson correlation coefficient, can not be applied to evaluate the correlation of the network traffic because the statistical distribution of the traffic is not known to be Gaussian. The 95% confidence ellipse is denoted by the green contour in Fig. 12, which covers 95% of the scatter plot points. It is graphically clear that the datasets for week 3, week 5 and week 9 are highly correlated; whereas, the contrary is true for week 1, week 4 and week 6. A moderate correlation can be observed for week 2, week 7 and week 8. In Fig. 11 (top), the MLMR framework shows a training MSE proportional to the correlation of the training dataset. For instance, the best training performance (relatively small MSE) is shown for dataset of week 6 that displays a circular scatter of the PCA points. Moreover, the training points (blue) show a uniform spatial spread within the contour, which implies balanced training of the DNN, thereby better performance. The worst performance is shown for week 9 dataset with highly elliptical scatter plot and positive 45° tilt. Notice that the extent of elliptical spread and tilt increase in an

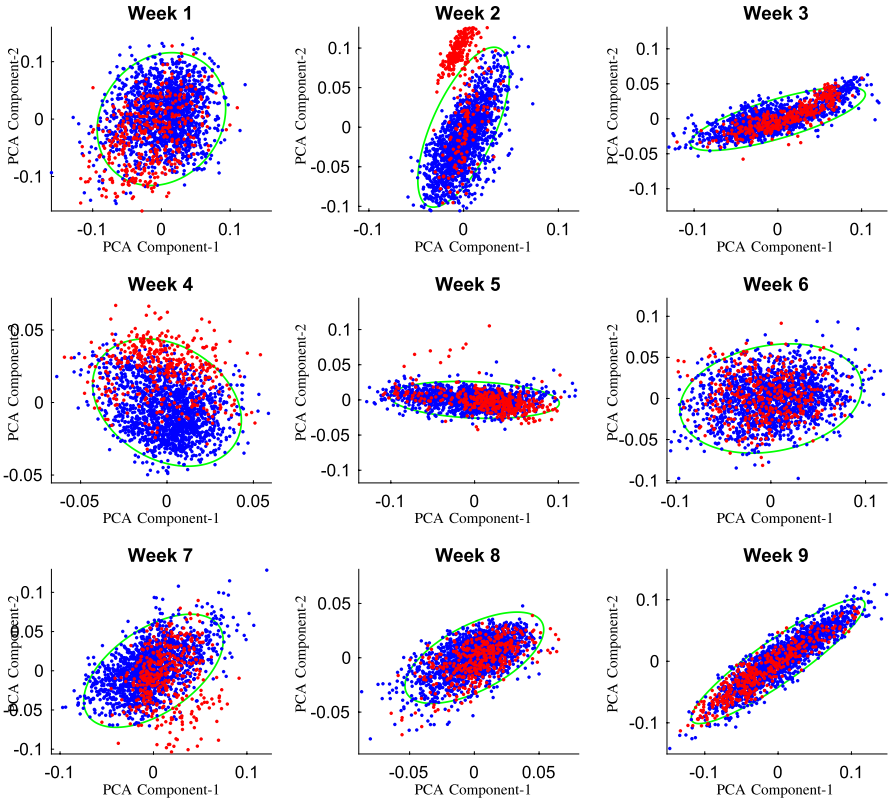

Fig. 12 Scatter plot of PCA component-1 versus PCA component-2 of the nine weeks datasets. The blue dots denote the learning dataset whereas the red dots denote the live-input. The green contour represents the 95% confidence ellipse

ascending order for week 5, week 3, and week 9, which is the same order for third-largest, second-largest and largest MSE as shown in Fig. 11 (top).

The prediction performance of the MLMR framework over the nine weeks is comparable to its training performance. However, its performance over week 1 and week 3 change rank position to be first and fourth in prediction rather than second and eighth in training, respectively. This is due to two features that week 1 and week 3 exhibit in Fig. 12. First, the live-input points (red) are uniformly spread over the training points (blue). Second, the direction of maximum variation of execution points and training points align well. Both features indicate similarity between the live-input and training dataset. It is interesting to see that the proposed framework performs well despite noise in live-input, i.e., the red points scattered out of the 95% confidence ellipse. This performance indicates that the DNN of the ML-based routing module captures a well balanced mapping between the input and output. In other words, the DNN learns the training dataset well but can still provide accurate predictions for live-inputs that uncorrelated with training dataset. This is particularly

demonstrated in the performance of the MLMR framework over week 2 dataset despite the presence of a cluster of live-points out of the 95% confidence ellipse of the training points. In week 2 dataset, large number of samples are not covered by the training dataset. This can also be observed in the sum of networks traffic for week 2 shown in Fig. 4. It is important to mention that these prediction scatter points direction of maximum variation aligns well with that of the training points, which indicates that the live-inputs enjoy the same characteristics of the training ones but at has a smaller magnitude.

The computation time of the CGbA and prediction time of the MLMR framework are illustrated in Table 3, along with the time improvement achieved by the MLMR framework. In addition, the training time of the MLMR framework is tabulated in Table 3. It is worth mentioning that the computational performance of both solutions can significantly be improved with the use of Graphical Processing Unit (GPU). As shown in the table, the proposed MLMR framework prediction time significantly improves the computation time of the CGbA by more than 98.99% for all evaluations. It can be observed that the prediction time of the MLMR framework is less than 430 milliseconds; therefore, it enables real-time routing in real networks. The learning time of the MLMR framework is less than 6.28 min and is triggered only once a week. The MLMR framework continues to apply the DNN trained model to predict routing solution in real-time, while a new DNN model is being trained. Hence, the long training time does not impact the real-time responsiveness of the of the MLMR framework. Furthermore, results confirm that the heuristic algorithm require more than 3.9 seconds to compute a routing solution, which is practically unacceptable.

6 Conclusions

In this paper, the application of machine learning techniques towards the multipath routing problem in software-defined network has been analyzed. The link capcitated and flow rules space constrained multipath routing problem is modeled as multicommodity network flow optimization problem. A column generations-based

Table 3 Time performance of the MLMR Framework and CGbA over the nine weeks (wk.s) of training data and live-inputs

Metric	wk. 1	wk. 2	wk. 3	wk. 4	wk. 5	wk. 6	wk. 7	wk. 8	wk. 9
CGbA computation Time (s)	5.38	5.39	3.89	4.59	3.90	5.68	5.79	5.43	5.60
MLMR prediction Time (s)	0.043	0.039	0.039	0.039	0.039	0.039	0.039	0.043	0.039
MLMR training Time (min)	5.56	6.13	5.90	6.28	6.26	5.93	5.80	6.18	5.81
`MLMR vs. CGbA Improvement (%)	99.20	99.27	98.99	99.14	98.99	99.31	99.32	99.20	99.30

heuristic algorithm is used to construct a dataset of network configurations and their routing solutions. A MLMR framework is developed to learn the mapping function between network configurations and their routing solutions in order predict a heuristic-like routing solution on the fly. Our precision analysis of the DNN constituting one of the salient modules of the framework shows that correlation among the learning dataset traffic characteristics is a key driver of the MLMR framework prediction accuracy. Application of the MLMR framework on real traffic data captured at real network show that the proposed MLMR framework demonstrates very high prediction accuracy of the heuristic routing solution represented by a prediction MSE less than 0.037; however, in less than 1% of the heuristic algorithm computation time.

Acknowledgements This work was supported and funded by Kuwait University Research Grant No. EO-07/18.

Appendix 1: Column Generation-Based Approach

We give a brief description of the column generation-based approach presented in [16] to solve the MCNF with side constraints. The **CRP** is expressed in arc-flows rather than path-flows. Then, a subset of paths that satisfy the side constraints, i.e., feasible paths, is considered for each flow. Denote the set of feasible paths $\bar{\mathcal{P}}_f = \{p | p \in \mathcal{P}_f, \sum_{l \in \mathcal{L}} \delta_{pl} \omega_{fl} \leq u_f\}$, $\forall f \in \mathcal{F}$. Incorporating the side constraints into the set of feasible paths, allows formulating the problem as a linear program which is given by [16],

$$\text{CRP-LP:} \quad \min_{\theta_p, \delta_{pl}} \sum_{f \in \mathcal{F}} \sum_{p \in \bar{\mathcal{P}}_f} \theta_p d_f \eta_p + \sum_{f \in \mathcal{F}} \mathfrak{M}\, \mathfrak{s}_f \tag{16}$$

$$\text{s.t.} \sum_{f \in \mathcal{F}} \sum_{p \in \bar{\mathcal{P}}_f} \delta_{pl} \theta_p d_f \leq g_l, \ \forall l \in \mathcal{L} \tag{17}$$

$$\sum_{p \in \bar{\mathcal{P}}_f} \theta_p d_f + \mathfrak{s}_f = d_f, \qquad \forall f \in \mathcal{F} \tag{18}$$

$$\theta_p d_f \geq 0 \qquad \forall p \in \bar{\mathcal{P}}_f, \forall f \in \mathcal{F} \tag{19}$$

$$\mathfrak{s}_f \geq 0 \ \ \forall f \in \mathcal{F}. \tag{20}$$

Here, $\mathfrak{s}_f$ is an artificial variable associated with demand f constraint. Moreover, each variable has a cost coefficient $\mathfrak{M}$. The theoretical difficulty of **CRP** is now hidden in $\bar{\mathcal{P}}_f$, i.e., the generation of paths.

Let $\longmapsto_l$ and κ_f, be dual variables associated with (17) and (18), respectively. Then, the **CRP-LP** problem can be decomposed into F *subproblems*, i.e., a problem for each flow, and the following column-generation-based approach is applied. In

each iteration, the Procedure 1 provides upper and lower bounds. Therefore, it can be stopped when they are sufficiently close to each other.

Procedure 1 Column-generation Approach

Step 1 Solve an initial *subproblem* and identify the shortest feasible path for every $f \in \mathcal{F}$. If no feasible shortest path exists for any f, terminate.

Step 2 Construct the first **CRP-LP** using the shortest paths identified in Step 1 and $\mathfrak{s}_f \ \forall f \in \mathcal{F}$.

Step 3 Get upper-bound by solving the constructed **CRP-LP** problem in Step 2. Obtain the value of the dual variables $\pi_l \leq 0$ and $\kappa_f \geq 0$. Also, compute the links reduced costs $\bar{c}_{fl} = c_{fl} - \pi_l$.

Step 4 Given the reduced costs, solve the *subproblem* for each flow f. If there exist a path p' such that $\bar{\eta}_p = \eta_p - \kappa_f - \sum_{l \in \mathcal{L}} \delta_{p'l} \pi_l \leq 0$, chose $\theta_{p'} d_f$ as entering variable for flow f, add p' to $\bar{\mathcal{P}}$, i.e., add column, and go to Step 3. Otherwise, go to Step 5.

Step 5 If $\mathfrak{s}_f < 0, \ \forall f \in \mathcal{F}$, the optimal solution is found. Otherwise, the problem is infeasible.

A label correcting algorithm is used to solve each of the *subproblems* [16]. The algorithm allows creation of multiple labels, each with a cost and weight at each node. The labels are sorted in a priority queue in accordance with their cost value. In each iteration, a label is chosen and updated. Let *s*, *t*, *lim*, *Q* and *n*(*i*) be the source, the destination, the weight limit, the priority queue, and the number of labels at node *i*, respectively. Then, a formal description of the multi-labeling algorithm is given in Algorithm 1. Let ω_{ij} be a weight coefficient of the link connecting nodes *i* and *j* for a given flow *f*; hence, it is equivalent to the a weight coefficient ω_{fl}. Similarly, let $\bar{c}_{ij}$ be the link reduced costs corresponding to $\bar{c}_{fl}$ of the link *l* connecting nodes *i* and *j*.

Algorithm 1 Label-correcting Algorithm

Initialization

$C_s^1 = 0$, $W_s^1 = 0$, $Q = \{(C_s^1, W_s^1)\}$, $n(s) = 1$ and $n(i) = 0, \ \forall i \neq s$.

Label Selection

if $Q = \{\}$ **then**

  terminate

else

  delete the first label $\{(C_i^h, W_i^h)\}$ at the head of the Q.

end if

Treatment of Labels

for $j \in \mathcal{L}_i^+$ **do** ▷ $\mathcal{L}_i^+$ is the set of nodes adjacent to node i.

  if $W_i^h + \omega_{ij} > lim$ **then**

    go to next j.

  else if there is a label $\{(C_i^{h'}, W_i^{h'})\}$ such that $C_i^h + \bar{c}_{ij} \geq C_i^{h'}$ and $W_i^h + \omega_{ij} \geq W_i^{h'}$ **then**

    go to next j.

  else

    Create a new label $\{(C_i^{n(i)+1} = C_i^h + \bar{c}_{ij}, W_i^{n(i)+1} = W_i^h + \omega_{ij})\}$

    if $j \neq t$ **then**

      insert label into Q.

    else if There is a label $\{(C_j^{h'}, W_j^{h'})\}$ such that $C_j^{n(i)+1} \leq C_j^{h'}$ and $W_i^{n(i)+1} \leq W_j^{h'}$ **then**

      delete $\{(C_j^{h'}, W_j^{h'})\}$ from Q.

      update $n(i)$ and return to **Label Selection**.

    end if

  end if

end for

Appendix 2: Complexity Analysis

In order to be able to compare the computational time of both MLMR framework and CGbA, the number of flows F can be written in terms of number of forwarding elements N. Assume that there are flows from each forwarding element to all other elements; hence, the number of flows $F = N^2$. Let h be the number of layers in the proposed DNN. Then, the maximum number of neurons in any of the layers of the proposed DNN is KN^2. Therefore, the runtime of h matrix multiplications is $\mathcal{O}\big(h(KN^2)^3\big)$, i.e., $\mathcal{O}(K^3N^6)$, where $K << N$ and h is constant.

The CGbA consists of five major steps outlined in Procedure 1. The Floyd-Warshall algorithm can be used to identify the shortest feasible paths between every pair of nodes in first step. Then, the runtime of the first and second step is $\mathcal{O}\big(N^3\big)$. The **CRP-LP** is solved in step three for each flow $f \in \mathcal{F}$. Given that the runtime of the label correcting algorithm is $\mathcal{O}\big(lim^2N^2\big)$, where lim is bounded by N and $F = N^2$, the third step runtime is $\mathcal{O}(N^6)$ [16]. In step four and five, the maximum number of columns that can be added for all flows is $N^2(K-1)$ and the runtime of solving the subproblem is $\mathcal{O}(N^4)$. Thus, the runtime of steps three, four and five is $\mathcal{O}\big(N^2(K-1)[N^6+N^4]\big)$. Hence, the overall time of CGbA is $\mathcal{O}\big(N^8(K-1)\big)$.

The empirical and asymptotic computational analysis demonstrate superiority of the proposed MLMR framework to the CGbA.

References

1. Cisco, V.: Cisco Visual Networking Index: Forecast and Trends, 2017–2022, vol. 1. White Paper, (2018)
2. Nunes, B.A.A., Mendonca, M., Nguyen, X.-N., Obraczka, K., Turletti, T.: A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. **16**(3), 1617–1634 (2014)
3. Xie, J., Yu, F.R., Huang, T., Xie, R., Liu, J., Wang, C., Liu, Y.: A survey of machine learning techniques applied to software defined networking (sdn): research issues and challenges. IEEE Commun. Surv. Tutor. **21**(1), 393–430 (2019). First quarter
4. Latah, M., Toker, L.: Artificial intelligence enabled software defined networking: a comprehensive overview. IET Netw. 11 (2018)
5. Yanjun, L., Xiaobo, L., Osamu, Y.: Traffic engineering framework with machine learning based meta-layer in software-defined networks. In Proceedings of the Network Infrastructure and Digital Content (IC-NIDC), 2014 4th IEEE International Conference on. IEEE, pp. 121–125 (2014)
6. Azzouni, A., Boutaba, R., Pujolle, G.: Neuroute: Predictive dynamic routing for software-defined networks. *arXiv preprint*. arXiv:1709.06002 (2017)
7. Francois, F., Gelenbe, E.: Optimizing secure sdn-enabled inter-data centre overlay networks through cognitive routing. In Proceedings of the 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 283–288 (2016)
8. Lin, S., Akyildiz, I. F., Wang, P., Luo, M.: Qos-aware adaptive routing in multi-layer hierarchical software defined networks: a reinforcement learning approach. In Proceedings of the 2016 IEEE International Conference on Services Computing (SCC), pp. 25–33 (2016)
9. Stampa, G., Arias, M., Sánchez-Charles, D., Muntés-Mulero, V., Cabellos, A.: A deep-reinforcement learning approach for software-defined networking routing optimization. *arXiv preprint*. arXiv :1709.07080 (2017)
10. Mendiola, A., Astorga, J., Jacob, E., Higuero, M.: A survey on the contributions of software-defined networking to traffic engineering. IEEE Commun. Surv. Tutor. **19**(2), 918–953 (2017)
11. Wang, Y.-C., Lin, Y.-D., Chang, G.-Y.: Sdn-based dynamic multipath forwarding for inter–data center networking. Int. J. Commun. Syst. 32(1), e3843 (2019). https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.3843
12. He, J., Rexford, J.: Toward internet-wide multipath routing. IEEE Netw. **22**(2), 16–21 (2008)
13. Doshi, M., Kamdar, A.: Multi-constraint qos disjoint multipath routing in sdn. In Moscow Workshop on Electronic and Networking Technologies (MWENT), pp. 1–5 (2018)
14. Zhang, P., Gang, Y., Huang, X., Zeng, S., Xie, K.: Bandwidth allocation with utility maximization in the hybrid segment routing network. IEEE Access **7**, 85 253–85 261 (2019)
15. Gao, C., Wang, H., Zhai, L., Yi, S., Yao, X.: Optimizing routing rules space through traffic engineering based on ant colony algorithm in software defined network. In Proceedings of the 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 106–112 (2016)
16. Holmberg, K., Yuan, D.: A multicommodity network-flow problem with side constraints on paths solved by column generation. INFORMS J. Comput. 15(1), pp. 42–57 (2003). https://pubsonline.informs.org/doi/abs/10.1287/ijoc.15.1.42.15151
17. Misra, S., Xue, G., Yang, D.: Polynomial time approximations for multi-path routing with bandwidth and delay constraints. In *Proceedings of the IEEE INFOCOM 2009*, pp. 558–566 (2009)
18. Rauch, H.E., Winarske, T.: Neural networks for routing communication traffic. IEEE Control Syst. Mag. **8**(2), 26–31 (1988)
19. Ouyang, Y.C., Bhatti, A.A.: Neural network-based routing in an intelligent computer communication network. In Proceedings of the System Theory, Twenty-Second Southeastern Symposium on. IEEE, pp. 444–448 (1990)

20. Ali, M.K.M., Kamoun, F.: Neural networks for shortest path computation and routing in computer networks. IEEE Trans. Neural Netw. **4**(6), 941–954 (1993)
21. Park, D.-C., Choi, S.-E.: A neural network based multi-destination routing algorithm for communication network. In 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227), vol. 2. IEEE, pp. 1673–1678 (1998)
22. Xia, Y., Wang, J.: A discrete-time recurrent neural network for shortest-path routing. IEEE Trans. Autom. Control **45**(11), 2129–2134 (2000)
23. Kato, N., Fadlullah, Z.M., Mao, B., Tang, F., Akashi, O., Inoue, T., Mizutani, K.: The deep learning vision for heterogeneous network traffic control: proposal, challenges, and future perspective. IEEE Wirel. Commun. **24**(3), 146–153 (2017)
24. Wang, M., Cui, Y., Wang, X., Xiao, S., Jiang, J.: Machine learning for networking: workflow, advances and opportunities. IEEE Netw. **32**(2), 92–99 (2018)
25. Fadlullah, Z., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., Mizutani, K.: State-of-the-art deep learning: evolving machine intelligence toward tomorrow's intelligent network traffic control systems. IEEE Commun. Surv. Tutor. (2017)
26. Boutaba, R., Salahuddin, M.A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., Caicedo, O.M.: A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. J. Internet Serv. Appl. **9**(1), 16 (2018). https://doi.org/10.1186/s13174-018-0087-2
27. Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K.-L.A., Elkhatib, Y., Hussain, A., Al-Fuqaha, A.: Unsupervised machine learning for networking: techniques, applications and research challenges. IEEE Access **7**, 65 579–65 615 (2019)
28. Choudhury, G., Lynch, D., Thakur, G., Tse, S.: Two use cases of machine learning for sdn-enabled ip/optical networks: traffic matrix prediction and optical path performance prediction [invited]. IEEE/OSA J. Opt. Commun. Netw. **10**(10), D52–D62 (2018)
29. Aibin, M.: Traffic prediction based on machine learning for elastic optical networks. Opt. Switch. Netw. 30, 33–39 (2018). http://www.sciencedirect.com/science/article/pii/S157342771730190X
30. Alvizu, R., Troia, S., Maier, G., Pattavina, A.: Matheuristic with machine-learning-based prediction for software-defined mobile metro-core networks. IEEE/OSA J. Opt. Commun. Netw. **9**(9), D19–D30 (2017)
31. Mao, B., Fadlullah, Z.M., Tang, F., Kato, N., Akashi, O., Inoue, T., Mizutani, K.: Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning. IEEE Trans. Comput. (2017)
32. Wang, Y., Martonosi, M., Peh, L.-S.: Supervised learning in sensor networks: New approaches with routing, reliability optimizations. In Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, vol. 1. IEEE, pp. 256–265 (2006)
33. Martín, I., Hernández, J. A., Troia, S., Musumeci, F., Maier, G., de Dios, O. G.: Is machine learning suitable for solving rwa problems in optical networks? In: Proceedings of the 2018 European Conference on Optical Communication (ECOC). IEEE, pp. 1–3 (2018)
34. Tang, F., Mao, B., Fadlullah, Z.M., Kato, N., Akashi, O., Inoue, T., Mizutani, K.: On removing routing protocol from future wireless networks: a real-time deep learning approach for intelligent traffic control. IEEE Wirel. Commun. **25**(1), 154–160 (2018)
35. Yu, C., Lan, J., Guo, Z., Hu, Y.: Drom: optimizing the routing in software-defined networks with deep reinforcement learning. IEEE Access **6**, 64 533–64 539 (2018)
36. Fang, C., Cheng, C., Tang, Z., Li, C.: Research on routing algorithm based on reinforcement learning in SDN. J. Phys. Conf. Ser. *1284*, 012053 (2019). https://doi.org/10.1088%2F1742-6596%2F1284%2F1%2F012053
37. Rischke, J., Sossalla, P., Salah, H., Fitzek, F.H.P., Reisslein, M.: Qr-sdn: towards reinforcement learning states, actions, and rewards for direct flow routing in software-defined networks. IEEE Access **8**, 174 773–174 791 (2020)
38. Göransson, P., Black, C., Culver, T.: Chapter 5—the openflow specification. Software Defined Networks (Second Edition) In P. Göransson, C. Black, and T. Culver (eds.) Morgan Kaufmann, Boston, pp. 89 – 136. (2017). http://www.sciencedirect.com/science/article/pii/B9780128045558000053
39. Liu, C., Malboubi, A., Chuah, C.: Openmeasure: Adaptive flow measurement inference with online learning in sdn. In: Proceedings of the 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 47–52 (2016)

40. Suh, J., Kwon, T. T., Dixon, C., Felter, W., Carter, J.: Opensample: a low-latency, sampling-based measurement platform for commodity sdn. In: Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems. IEEE, pp. 228–237 (2014)
41. OpenDylight: Model-Driven Service Adaptation Layer (MD-SAL) (2018). https://docs.opendaylight.org/projects/mdsal
42. Yang, J., Huang, X., Jiang, S.: An accurate approach for traffic matrix estimation in large-scale backbone networks. In: Proceedings of the 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 425–431 (2016)
43. Queiroz, W., Capretz, M. A., Dantas, M.: An approach for sdn traffic monitoring based on big data techniques. J. Netw. Comput. Appl. 131, 28–39 (2019). http://www.sciencedirect.com/science/article/pii/S1084804519300244
44. Lakhina, A., Papagiannaki, K., Crovella, M., Diot, C., Kolaczyk, E. D., Taft, N.: Structural analysis of network traffic flows. In: Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, ser. SIGMETRICS '04/Performance '04. Association for Computing Machinery, New York, pp. 61–72 (2004). https://doi.org/10.1145/1005686.1005697
45. Stoev, S., Taqqu, M. S., Park, C., Michailidis, G., Marron, J.: Lass: a tool for the local analysis of self-similarity. Comput. Stat. Data Anal., 50(9), 2447–2471 (2006), statistical signal extraction and filtering. http://www.sciencedirect.com/science/article/pii/S0167947304004062
46. Kirichenko, L., Radivilova, T., Deineko, Z.: Comparative analysis for estimating of the hurst exponent for stationary and nonstationary time series. Inf. Technol. Knowl. **5**(1), 371–388 (2011)
47. Chen, C.: hurst parameter estimate (2020) https://www.mathworks.com/matlabcentral/fileexchange/19148-hurst-parameter-estimate
48. Martín, I., Troia, S., Hernández, J.A., Rodríguez, A., Musumeci, F., Maier, G., Alvizu, R., González de Dios, Ó.: Machine learning-based routing and wavelength assignment in software-defined optical networks. IEEE Trans. Netw. Serv. Manage. **16**(3), 871–883 (2019)
49. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
50. Kingma, D. P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint. arXiv:1412.6980 (2014)
51. (2019) Keras documentation. [Online]. Available: https://keras.io
52. (2019) Tensorflow. [Online]. Available: https://www.tensorflow.org
53. Zhang, Y.: (2014) 6 months of abilene traffic matrices. [Online]. Available: http://www.cs.utexas.edu/~yzhang/
54. Leduc, G., Abrahamsson, H., Balon, S., Bessler, S., D'Arienzo, M., Delcourt, O., Domingo-Pascual, J., Cerav-Erbas, S., Gojmerac, I., Masip, X., Pescapè, A., Quoitin, B., Romano, S., Salvadori, E., Skivée, F., Tran, H., Uhlig, S., Ümit, H.: An open source traffic engineering toolbox. Comput. Commun. **29**(5), 593–610 (2006) networks of Excellence. http://www.sciencedirect.com/science/article/pii/S0140366405002124
55. Leduc, G.: (2020) Toolbox for traffic engineering methods project (totem). [Online]. Available: https://totem.info.ucl.ac.be/index.html
56. Howell, D.: (2004) Internet2. [Online]. Available: https://www.internet2.edu/news/detail/1978/
57. Géron, A.: Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc. (2017)
58. Yen, J .Y.: Finding the k shortest loopless paths in a network. Manage. Sci. **17**(11), 712–716 (1971)
59. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org
60. Hinton, G., Srivastava, N., Swersky, K.: Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on **14**, 8 (2012)
61. Campbell, N.A., Atchley, W.R.: The geometry of canonical variate analysis. Syst. Zool. **30**(3), 268–280 (1981). http://www.jstor.org/stable/2413249

Mohamad Khattar Awad (Senior Member, IEEE) received the B.A.Sc. degree in electrical and computer

engineering (communications option) from the University of Windsor, Windsor, ON, Canada, in 2004, and the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2006 and 2009, respectively. From 2004 to 2009, he was a Research Assistant with the Broadband Communications Research Group, University of Waterloo. From 2009 to 2012, he was an Assistant Professor of electrical and computer engineering with the American University of Kuwait. Since 2012, he has been with Kuwait University, where he is currently an Associate Professor of computer engineering. His research interests include wireless and wired communications, software-defined networks resource allocation, wireless networks resource allocation, and acoustic vector-sensor signal processing. Dr. Awad received the Ontario Research and Development Challenge Fund Bell Scholarship in 2008 and 2009, the University of Waterloo Graduate Scholarship in 2009, the Fellowship Award from the Dartmouth College, Hanover, NH, in 2011, the Kuwait University Teaching Excellence Award in 2015, and the Best Young Researcher Award in 2017. He serves on the editorial board for the IEEE Transactions on Green Communications and Networking.

Marwa Hassan Hafez Ahmed received B.Sc. degree in Computer Science and Automatic Control from Faculty of Engineering of Alexandria University, Egypt in 2003. She received the M.Sc. degree in Computer Engineering from Kuwait University, Kuwait in 2020. She has more than 17 years of experience in the field Information Technology and more than 8 years of experience in project management. She is currently a Senior Project Manager with the Technical Enterprise Project Management Office of Kuwait International Bank (KIB), Kuwait. Her research interests include Machine Learning, Distributed Systems, Software Defined Networks, and Digital Transformation.

Ali F. Almutairi received the B.S. degree in electrical engineering from the University of South Florida, Tampa, FL, USA, in 1993, and the M.S. and Ph.D. degrees in electrical engineering from the University of Florida, Gainesville, FL, USA, in 1995 and 2000, respectively. He served as the Vice Dean for academic affairs with the College of Engineering and Petroleum, Kuwait University, from 2016 to 2018. He served as the Chairperson for the Electrical Engineering Department, Kuwait University, from 2007 to 2011, and served as the Graduate Program Director of the Electrical Engineering Department, Kuwait University, from 2015 to 2016. He is currently a Professor with the Electrical Engineering Department, and the Dean of admission and registration with Kuwait University. His current research interests include multiuser detection, wireless networks, antenna design, and current and future cellular networks performance issues. He is a member of other professional societies. He served/serving as an Associate Editor and a Reviewer for many technical publications. In 1993, he received a full scholarship from Kuwait University to pursue his graduate studies.

Imtiaz Ahmad received the B.Sc. degree in Electrical Engineering from the University of Engineering and Technology at Lahore, Pakistan, the M.Sc. degree in Electrical Engineering from the King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, and the Ph.D. degree in Computer Engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1988, and 1992, respectively. Since 1992, he has been with the Department of Computer Engineering, Kuwait University, Kuwait, where he is currently a Professor. His research interests include the design automation of digital systems, parallel and distributed computing, machine learning and software-defined networks.