Deep Reinforcement Learning Based QoS-Aware Routing in Knowledge-Defined Networking

Tran Anh Quang Pham[1](✉), Yassine Hadjadj-Aoul[1], and Abdelkader Outtagarts[2]

[1] Inria, Univ Rennes, CNRS, IRISA, Rennes, France
quang.pham-tran-anh@inria.fr, yassine.hadjadj-aoul@irisa.fr
[2] Nokia Bell Labs, Paris, France
abdelkader.outtagarts@nokia-bell-labs.com

Abstract. Knowledge-Defined networking (KDN) is a concept that relies on Software-Defined networking (SDN) and Machine Learning (ML) in order to operate and optimize data networks. Thanks to SDN, a centralized path calculation can be deployed, thus enhancing the network utilization as well as Quality of Services (QoS). QoS-aware routing problem is a high complexity problem, especially when there are multiple flows coexisting in the same network. Deep Reinforcement Learning (DRL) is an emerging technique that is able to cope with such complex problem. Recent studies confirm the ability of DRL in solving complex routing problems; however, its performance in the network with QoS-sensitive flows has not been addressed. In this paper, we exploit a DRL agent with convolutional neural networks in the context of KDN in order to enhance the performance of QoS-aware routing. The obtained results demonstrate that the proposed approach is able to improve the performance of routing configurations significantly even in complex networks.

Keywords: Knowledge-Defined networking · Software-Defined networking · Routing · Deep Reinforcement Learning

1 Introduction

Routing optimization and particularly traffic engineering are the most fundamental networking task and, therefore, has been extensively studied in a number of contexts. The emergence of Software-Defined networking (SDN) paradigm unveiled new capabilities in routing. In SDN, the control plane and the forwarding plane are separated. A centralized controller is responsible for computing routing decisions, thus reducing the complexity of network elements. Moreover, it is able to monitor the demands and availability of resources globally; therefore, it is capable of matching the resource needs optimally. However, the size of the solution space as well as the complexity of the optimization problem are

T. Q. Duong et al. (Eds.): Qshine 2018, LNICST 272, pp. 14–26, 2019.
https://doi.org/10.1007/978-3-030-14413-5_2

increased since SDN paradigm adds degree of freedoms in routing (flow-based forwarding vs destination-based forwarding) and the centralized controller has to solve the problem with a global view. As a result, new solutions of routing for SDN have been proposed [1,2]. In [3], the author proposed a Knowledge plane (KP), which is based on machine learning and cognitive techniques, to control the network. The KP is able to offer many advantages to networking, such as automation and recommendation and it may lead to a paradigm shift on the way we operate, manage, and optimize the data networks.

Machine learning (ML) techniques has been adopted and made breakthroughs in a number of application areas. ML algorithms can be classified into three categories: supervised learning (SL), unsupervised learning (USL), and reinforcement learning (RL). While SL and USL focuses on classification or regression tasks, RL algorithms learn to identify the best action series in order to maximize a given objective function (i.e reward). The most important advantages of ML is its capability of dealing with complicated problems; thus it is intuitive to exploit ML in the network domain where the complex problems are common [4]. In the context of routing, RL has been confirmed that it outperforms other ML techniques [5]; therefore, this paper focuses only on RL techniques.

RL techniques have been exploited to solve routing optimization [6]. They are also used in QoS routing [7]; however, table-based RL agents cannot provide efficient solutions for unseen network states. Deep Reinforcement Learning (DRL) is able to provide solutions for unseen network states, which cannot be achieved by traditional table-based RL agents. Moreover, it overcomes the iterative improvement process of optimization and heuristics by having a DRL agent providing a near-optimal solution in one single step. Recent breakthroughs in deep neural networks [8,9] has improved the performance of DRL; thus paving a way for adopting DRL in the context of networking. Deep convolutional neural networks [10] has proved its efficiency in various applications, e.g. image processing; therefore, the adoption of deep convolutional neural networks in the context of networking may offer some advantages.

In [5], the authors studied on the impacts of inputs and action spaces to the performance of machine learning in the context of routing problems. The results in the paper confirmed learning the link weights with traffic matrix outperforms other approaches. The link weights may have real positive values which cannot be solved by the well-known Deep Q Network algorithm [11] since it can handle only discrete and low-dimensional action space [12]. The authors in [12] addressed the needs of continuous control by proposing a Deep Deterministic Policy Gradient (DDPG) algorithm. The capability of DDPG in solving routing problem has been confirmed in [13]. However, the author in [13] focused on minimize the mean latency in the network and did not address the QoS routing problem with multiple metrics (e.g. latency and packet loss rate).

In this paper, we study on a DDPG agent which learns to make the QoS-aware routing decisions. Unlike to the DDPG agent presented in [13] which used fully connected layers, we take advantages of convolutional neural networks in order to extract the mutual impacts between flows in the networks; thus being able to provide better routing configurations. The rest of this paper is structured as follows. A brief review of KDN can be found in Sect. 2. A problem formulation followed by the proposed convolutional DRL network, which can improve the routing performance, is presented in Sect. 3. The performance of the proposed DRL network is verified by simulations, presented in Sect. 4.

2 Knowledge-Defined Networking

In the context of SDN architecture, a concept of Knowledge plane (KP) has been introduced in [3]. The addition of a KP to the conventional SDN paradigm unveils a new paradigm, what is called Knowledge-Defined Networking (KDN). In KDN, the data plane is responsible for storing, forwarding and processing data packets. Practically, it comprises line-rate programmable forwarding hardware which operate without being aware of the rest of the network and depend on the other planes to populate forwarding tables and update configurations.

The control plane exchanges operational states in order to inform the data plane about forwarding and processing rules. In SDN, a logically centralized SDN controller is responsible for this task. It programs the data plane via a southbound interface. The role of the management plane is to guarantee the proper operation and performance of the network in the long term. Its main tasks are to monitor the network and to provide network analytic. In fact, this task is could also be handled by the SDN controller. The KP exploits the control and the management planes to provide a comprehensive view and control over the network. Based on ML approaches, it is capable of learning the behavior of the network and operating the network appropriately.

The KDN paradigm uses a control loop to provide automation, optimization, validation, and estimation. Figure 1 describes the basic steps of the KDN control. The role of the analytics platform is to collect information to provide a completely global view of the network. It monitors the data plane elements in real-time and retrieves the control and management states from the SDN controller.

ML algorithms are able to learn the network behavior; thus, they play an important role in KP. The current and historical data offered by the analytics platform are fed to learning algorithms so as to generate knowledge. The decision maker (e.g. human decision or automatic decision) will give a decision based on knowledge and execute it via the northbound SDN controller API.

3 Convolutional Deep Learning for QoS-Routing

3.1 Network Model

We consider a framework in which a decision maker repeatedly selects routing configurations. Each flow is identified by the source and the destination. Traffic and QoS demands of each flow may be different. For instance, the video streaming may require a low latency and packet loss while web surfing is tolerant to packet

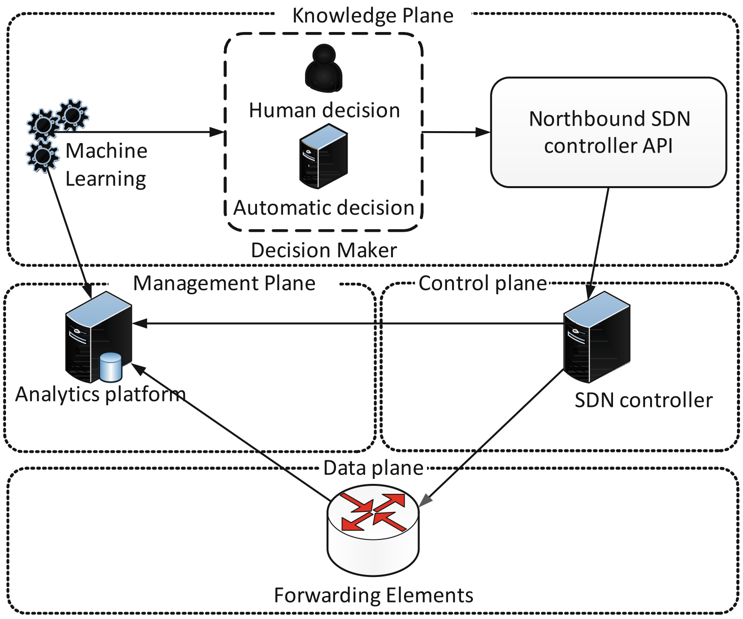

Fig. 1. KDN operation

loss and latency. Thus, we introduce a metric $\mathtt{QF}(\mathcal{R})$, named the number of qualified flows of routing configuration $\mathcal{R}$, which is the number of flows that meet their QoS requirements when applying the routing configuration $\mathcal{R}$.

We model the network as a capacitated indirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$, where $\mathcal{V}$ and $\mathcal{E}$ are the vertex and edge sets, respectively, and $c : \mathcal{E} \leftarrow \mathbb{R}^+$ assigns a capacity to each edge. The number of nodes is $N = |\mathcal{V}|$ and the number of links is $L = |\mathcal{E}|$.

Traffic demands of flows are presented in a $N \times N$ matrix, named Traffic Matrix (TM), in which the entry (i, j) is the traffic demand between source i and destination j. Based on this matrix, the decision maker computes a L-vector, named link-weight vector, which is composed of weights of links in order to describe the routing configurations. This vector is exploited to calculate a forwarding table using the Dijkstra algorithm.

3.2 Deep RL Agent Model

The proposed RL agent is an off-policy, actor-critic, deterministic policy gradient algorithm [14] that interacts with the data network through states, actions, and rewards [15]. The state is the TM while the action is the link-weight vector. We define two reward functions: (1) the mean of QoS metrics (e.g. latency and end-to-end packet loss) and (2) the mean of qualified flows $\mathtt{QF}(\mathcal{R})$.

The objective of the agent is to identify the optimal policy π mapping from the state space S to the action space A, $\pi : S \rightarrow A$, that maximizes the expected reward (minimize the mean of QoS metrics or maximize the number of qualified flows). It is done by repeatedly ameliorating its knowledge of the connections between the state, action, and reward over the means of deep neural networks of the actor and the critic.

A vanilla approach adopting the Deep Deterministic Policy Gradient (DDPG) to solve routing problem has been introduced in [13]. In that paper, the actor and critic networks consists of dense layers connected serially. Consequently, the input is a vector in which each entry is the amount of a flow. This approach may not be able to explore the mutual impacts between flows. Consequently, we propose an extra module comprising multiple convolutional layers before the dense layers in order to allow the DRL network to learn the mutual impacts between flows more efficiently. Moreover, it is able to feed the proposed DRL network with multiple channels in order to provide a more comprehensive view of demands (e.g. latency and packet loss rate requirements). The details of this proposal is presented in the next section.

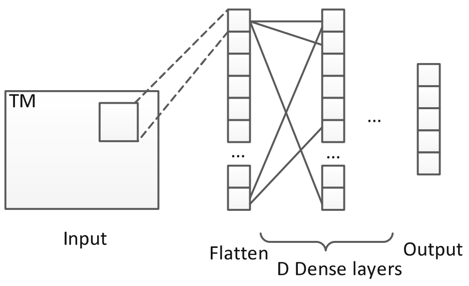

Fig. 2. Neural network architecture used in [13]

3.3 Convolutional DRL for QoS-Routing

Conventional neural network includes dense neural network layers as shown in Fig. 2. Unlike to fully connected layer in which each neuron connects to every neuron in the previous layer, a convolutional layer is only connected to a few local neurons in the previous layers; therefore it is relatively simpler than the fully connected layer. Moreover, it is able extract and learn mutual impacts of adjacent flows, thus enhancing the efficiency of routing configurations. Figure 3 presents the proposed neural network architecture.

TM is the input of the convolutional layers. There are M convolutional layers in critic network as well as in actor network. In fact, the value of M as well as the hyper-parameter of convolutional layers (e.g. stride size, filter size, etc.) may impact the DRL's performance. In the proposed architecture we may, also, have several max pooling layers. Having several layers of max pooling has the disadvantage of removing too much information, which significantly degrades the performance. However, a thorough study of these parameters is beyond the scope of our paper.

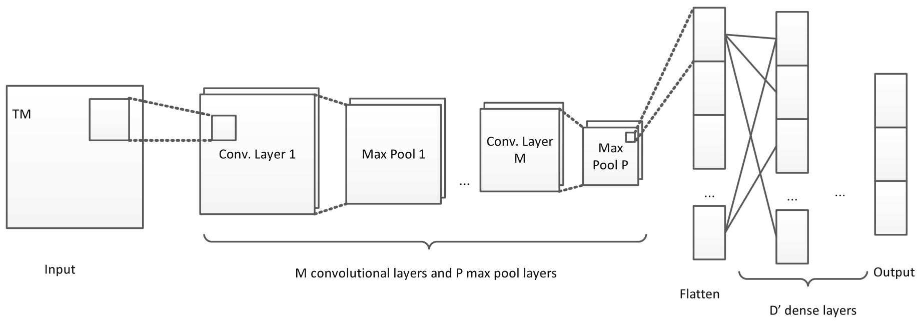

Fig. 3. Convolutional neural network architecture

We use the same value of M and hyper-parameters for both critic network and actor networks in all configurations in simulations. The output of M convolutional layers is flattned and fed into D' dense layer as the vanilla approach in order to compute the action (i.e. link-weight vector).

4 Simulation

To assess the performance of proposed approach we use a network topology BtEurope [16] of 24 nodes and 37 full-duplex links, with uniform link capacities (10 Mbps). The OMNeT++ discrete event simulator [17] (v5.4.1) was used to obtain the latency and packet loss rate under given traffic conditions (TM) and routing configurations ($\mathcal{R}$). We generate non-spare TM using a gravity model [18]. The total traffic entering (exiting) the network is generated by an exponential distribution with the mean 1000. The packet intervals of each flows follows an exponential distribution with the rate given in TM. The packet size is fixed at 1000 bits. We consider two QoS metrics: latency and packet loss rate. The QoS requirements of each flow is generated uniformly in range 1 ms to 100 ms for latency and 0 to 30% for packet loss rate.

Adam [19] is used for learning the neural network parameters. The learning rates for actor and critic networks are 10^{-4} and 10^{-3}, respectively. The discount factor γ is 0.99. The soft target is updated with the coefficient of $\tau = 0.001$. The batch size is 32. For the dense neural network learning, we reuse the configurations in [13], in which the number of dense layers is $D = 2$ with a numbers of units equal to 91 and 42, respectively. For the proposed approach, we used only one max pooling layer and 3 convolutional layers with 24 filters. The number of dense layers in the proposed approach is $D' = 1$, with 42 units.

We runs 10 simulations with different random seeds. The results are the mean of these runs. Each run is composed of 100 episodes, where one episode includes 50 different TMs.

4.1 Homogeneous Capacity Networks

The DDPG with dense neural networks is the approach used in [13] and the DDPG with convolutional neural networks is the proposed DRL network proposed in Sect. 3. For each type of DRL network, the objective could be maximizing the mean of QoS metrics or maximizing the mean of QFs. The combination of dense neural networks with the maximizing tmean QoS objective and maximizing mean of QFs objective are denoted as `Dense-QoS` and `Dense-QF`, while `Conv-QF` and `Conv-QoS` are the combination of convolutional neural networks with the QF objective and the mean QoS objective.

Figure 4 shows the mean latency under difference configurations of neural networks and objectives. The mean latency of `Dense-QoS` is slightly worse than of `Dense-QF`. It also applies to the packet loss rate and the mean number of QFs as shown in Figs. 5 and 6. It is because the QF objective aims to maximize the number of flows satisfying the QoS requirements and thus being able to distribute traffic better. As a result, the lower congestion level can be obtained and it leads to a better performance in the mean of QFs as well as the mean of QoS. Consequently, we focus on the mean QF objective in the following simulations.

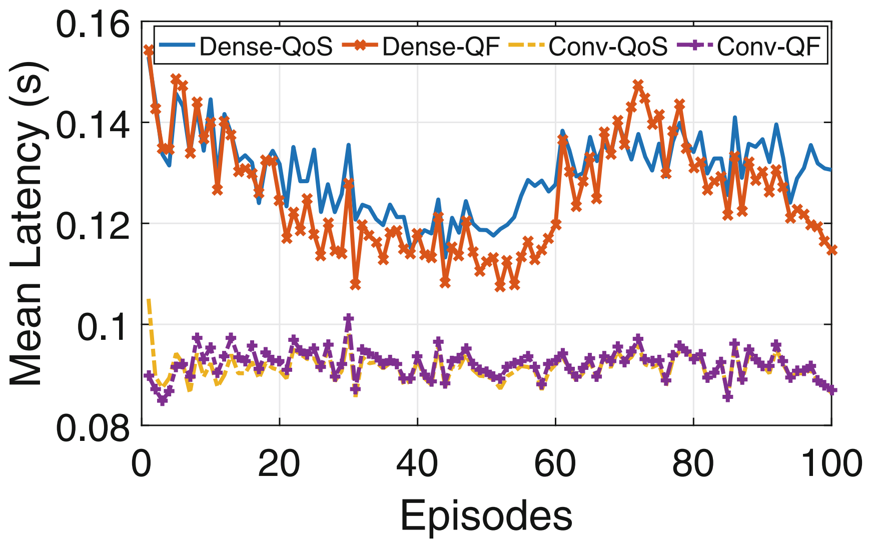

Fig. 4. Mean latency of dense vs convolutional networks

Generally, the performances of `Conv-QF` and `Conv-QoS` are better than of `Dense-QF` and of `Dense-QoS` in every QoS metric and the mean of QFs. It is because convolutional neural networks are able to extract the mutual impacts of flows; thus avoiding congestion better. A remarkable lower latency can be obtained by convolutional neural networks; however, the gaps in the mean of QFs are not always remarkable. It may be caused by the loose QoS requirements,

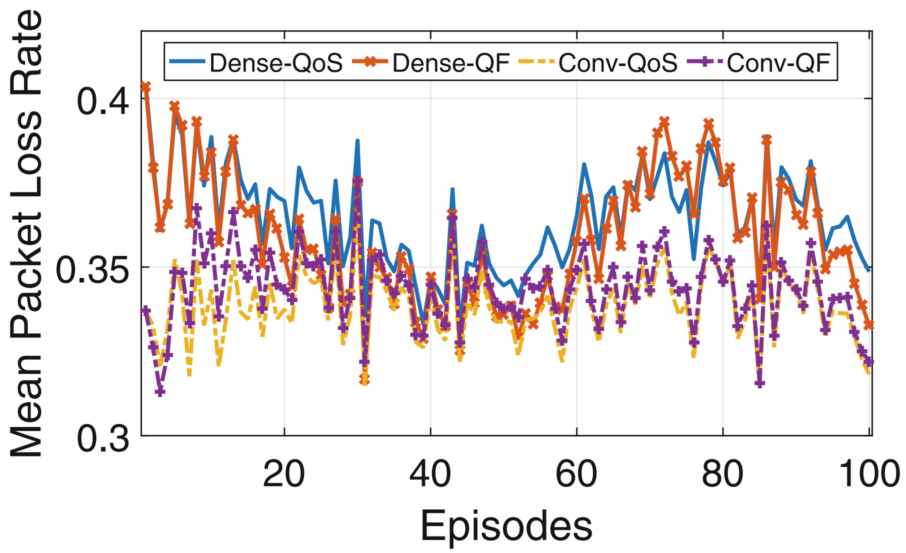

Fig. 5. Mean packet loss rate of dense vs convolutional networks

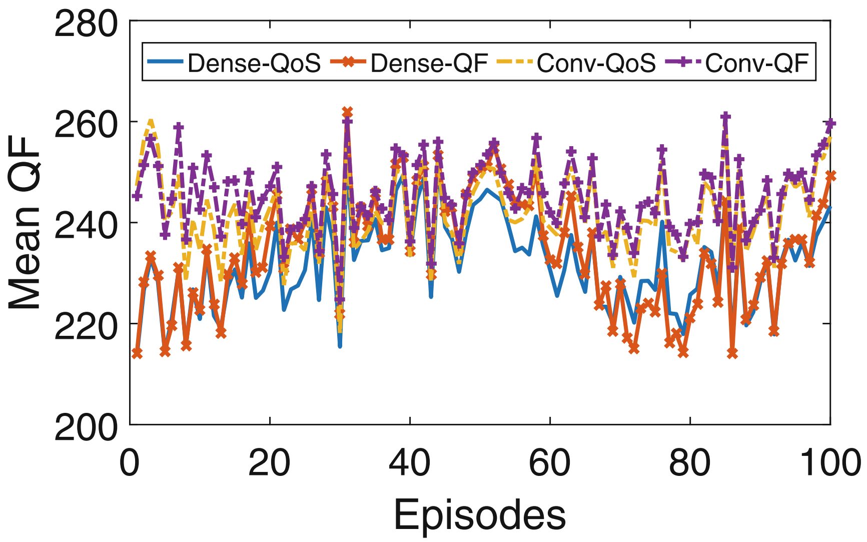

Fig. 6. Mean number of qualified streams of dense vs convolutional networks

especially the latency (up to 100 ms). To verify this hypothesis, we conduct the simulation with the latency in range of 1 ms to 10 ms and the loss rate in range of 0 to 10%. Figure 7 describes the mean of QFs in strict QoS requirements. `Conv-QF` outperforms `Dense-QF`. The gaps are significant.

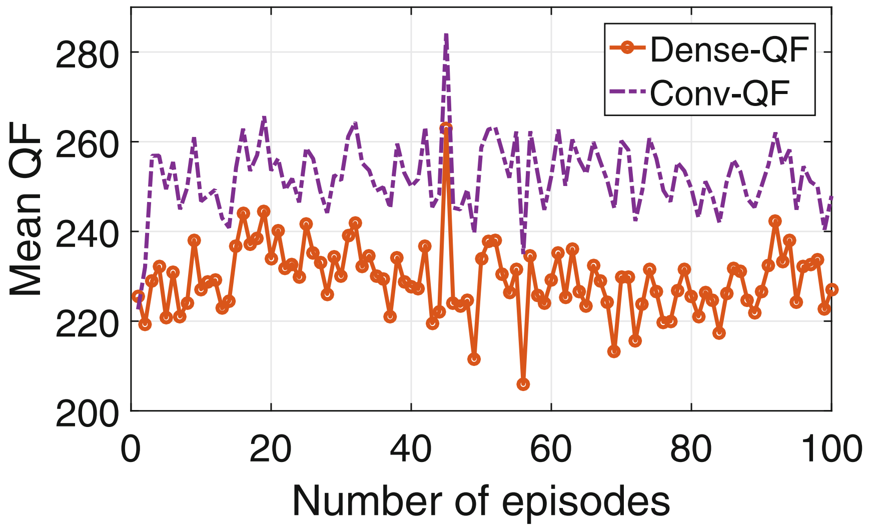

Fig. 7. Mean number of QF in strict QoS requirements

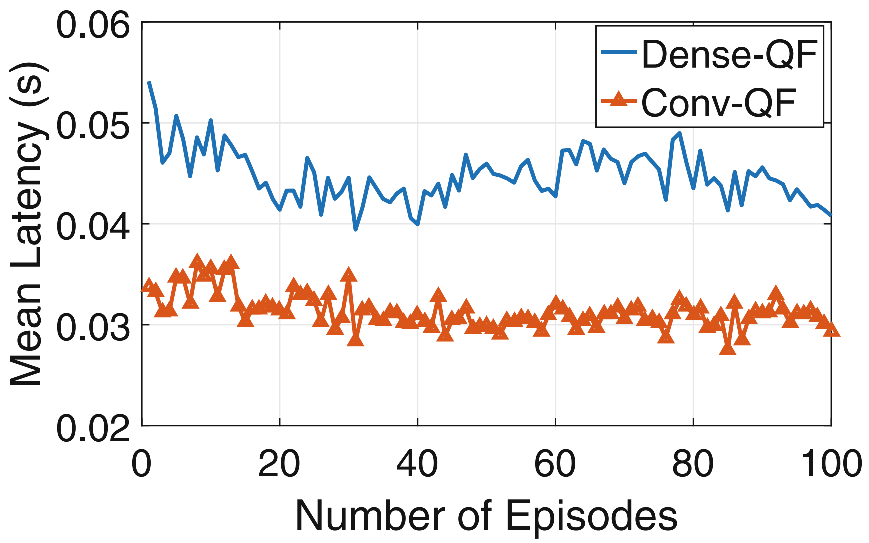

Fig. 8. Mean latency in heterogeneous networks

4.2 Heterogeneous Capacity Networks

In contrast to the previous section, the network composes of links with different capacities, i.e 10 Mbps, 20 Mbps, 50 Mbps, and 100 Mbps. This topology has higher capacity; however it is also more complicated than homogeneous networks.

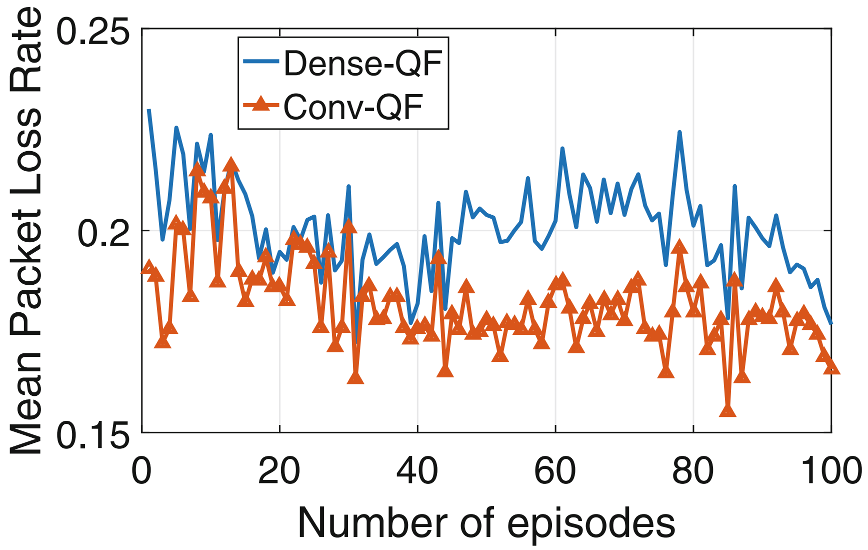

Fig. 9. Mean packet loss rate in heterogeneous networks

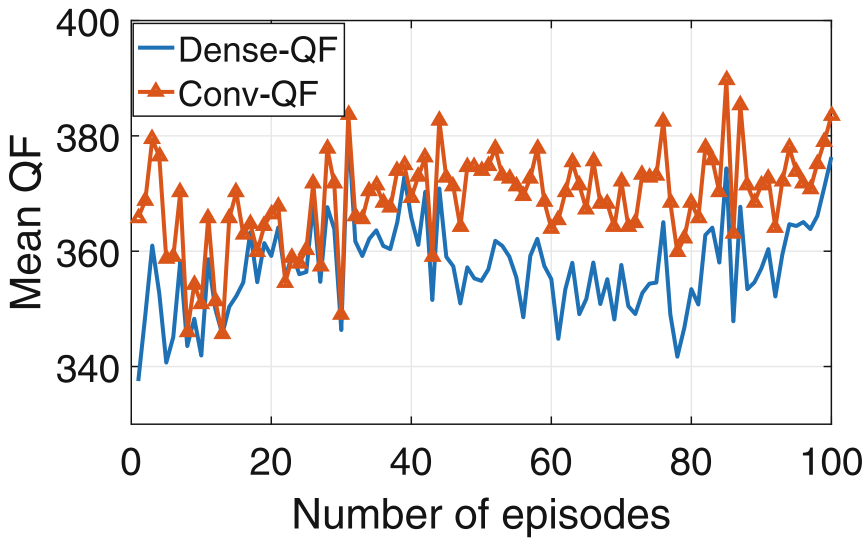

Fig. 10. Mean number of qualified streams under single and multiple channel input

The QoS requirements are from 1 ms to 100 ms for latency and 0 to 30% for packet loss rate.

Both latency and packet loss rate of `Conv-QF` are better than of `Dense-QF` as shown in Figs. 8 and 9. Consequently, `Conv-QF` has better the mean of QFs than `Dense-QF` as described in Fig. 10. The gaps between `Conv-QF` and `Dense-QF`

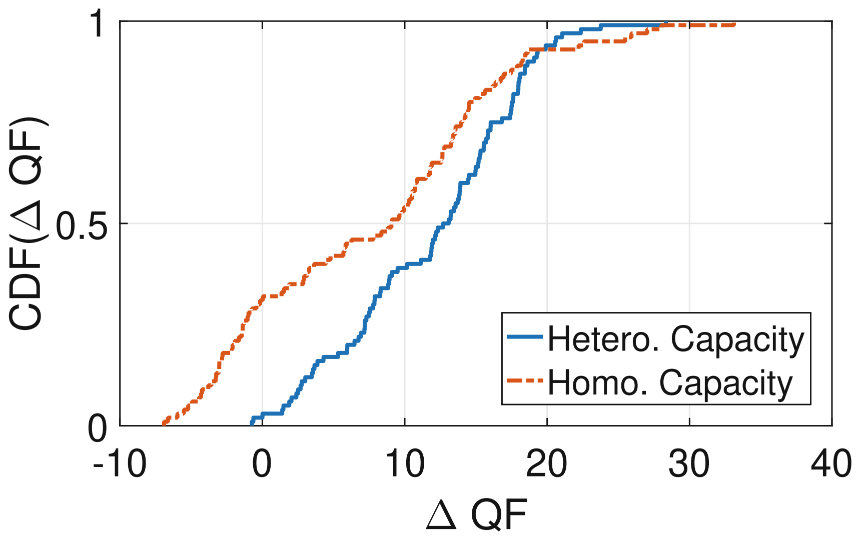

Fig. 11. Empirical CDF of Δ QF

in heterogeneous capacity networks are broader than in homogeneous capacity networks. To compare them, we define the metric $\Delta QF = QF_{\texttt{Conv-QF}} - QF_{\texttt{Dense-QF}}$ to indicate the gaps in the performance of `Conv-QF` and `Dense-QF`. The empirical cumulative distribution functions of ΔQF in homogeneous capacity network and heterogeneous capacity network are shown in Fig. 11. Most of ΔQF values (98%) in heterogeneous capacity network (capacity from 10 Mbps to 100 Mbps) is positive leading to `Conv-QF` clearly outperforms the `Dense-QF` in the mean of QFs. Meanwhile, around 30% of ΔQF in homogeneous capacity network (10 Mbps for all links) is negative. It means `Conv-QF` can cope with the heterogeneous and high capacity networks better than with the low and uniform capacity networks.

5 Conclusions

Knowledge Defined Networking is a potential paradigm for future data networks. This paper studied the structures of neural networks in deep reinforcement learning (DRL) in the context of routing. By adopting convolutional neural networks, the proposed mechanism is able to learn the mutual impacts between flows in the networks; therefore, it is able to provide a better routing configurations. The gaps is even broadened in heterogeneous capacity networks, indicating the advantages of convolutional neural networks in coping with complex scenarios. In future, we intend to extend this work by modifying DDPG algorithm in order to enhance the performance as well as apply it to more complicated problem in networking.

References

1. Akyildiz, I.F., Lee, A., Wang, P., Luo, M., Chou, W.: A roadmap for traffic engineering in SDN-OpenFlow networks. Comput. Netw. **71**, 1–30 (2014). https://doi.org/10.1016/j.comnet.2014.06.002, http://www.sciencedirect.com/science/article/pii/S1389128614002254
2. Layeghy, S., Pakzad, F., Portmann, M.: SCOR: software-defined constrained optimal routing platform for SDN. CoRR abs/1607.03243 (2016). http://arxiv.org/abs/1607.03243
3. Clark, D.D., Partridge, C., Ramming, J.C., Wroclawski, J.T.: A knowledge plane for the internet. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 3–10. ACM (2003)
4. 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). https://doi.org/10.1109/MNET.2017.1700200
5. Valadarsky, A., Schapira, M., Shahaf, D., Tamar, A.: A machine learning approach to routing. CoRR abs/1708.03074 (2017). http://arxiv.org/abs/1708.03074
6. Boyan, J.A., Littman, M.L.: Packet routing in dynamically changing networks: a reinforcement learning approach. In: Advances in Neural Information Processing Systems, pp. 671–678 (1994)
7. 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: 2016 IEEE International Conference on Services Computing (SCC), pp. 25–33, June 2016. https://doi.org/10.1109/SCC.2016.12
8. Li, Y.: Deep reinforcement learning: an overview. arXiv preprint arXiv:1701.07274 (2017)
9. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: A brief survey of deep reinforcement learning. CoRR abs/1708.05866 (2017). http://arxiv.org/abs/1708.05866
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
11. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
12. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
13. Stampa, G., Arias, M., Sanchez-Charles, D., Muntés-Mulero, V., Cabellos, A.: A deep-reinforcement learning approach for software-defined networking routing optimization. CoRR abs/1709.07080 (2017). http://arxiv.org/abs/1709.07080
14. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: Proceedings of the 31st International Conference on International Conference on Machine Learning, ICML 2014, vol. 32, pp. I-387–I-395. JMLR.org (2014). http://dl.acm.org/citation.cfm?id=3044805.3044850
15. Sutton, R.S., Barto, A.G., et al.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
16. The internet topology zoo. http://www.topology-zoo.org/dataset.html
17. Varga, A.: Discrete event simulation system. In: Proceedings of the European Simulation Multiconference (2011)

18. Roughan, M.: Simplifying the synthesis of internet traffic matrices. SIGCOMM Comput. Commun. Rev. **35**(5), 93–96 (2005). https://doi.org/10.1145/1096536.1096551
19. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980