Generative Adversarial Network-Based Transfer Reinforcement Learning for Routing With Prior Knowledge

Tianjian Dong, Qi Qi, *Member, IEEE*, Jingyu Wang, *Member, IEEE*, Alex X. Liu, *Fellow, IEEE*, Haifeng Sun, *Member, IEEE*, Zirui Zhuang, *Member, IEEE*, and Jianxin Liao, *Member, IEEE*

***Abstract*—With the incremental deployment of software defined networking, the routing algorithms have gained more power on observability and controllability. Deep reinforcement learning, as an experience-driven approach, shows considerable potential in routing problem with the help of the centralized controller. It is an adaptive, lightweight, and model-free approach to coping with dynamic runtime status, large-scale traffic, and heterogeneous objective of SDN routing. However, it is still not suitable for the variable and complex emerging networks, because the huge training cost prevents fast convergence in a varying or discrepant environment. In this paper, we propose a transfer reinforcement learning algorithm to improve the training efficiency, and handle the variation in network status and topology. Specifically, we leverage the generative adversarial network to learn domain-invariant features that is suitable for deep reinforcement learning-based routing in different network environments. This mechanism utilizes the previous model and accelerates the training process. We implement our routing algorithm in the production level software switches and controller, while evaluating it comprehensively with many topologies and network status distributions. The experimental results show that our work not only outperforms the state-of-the-art deep reinforcement learning-based routing frameworks, but also has more training efficiency than the naive transfer learning algorithm both on different topologies and network status distributions.**

***Index Terms*—Routing, transfer reinforcement learning, generative adversarial network, software defined networking.**

Manuscript received October 23, 2020; revised February 14, 2021 and April 26, 2021; accepted April 29, 2021. Date of publication May 4, 2021; date of current version June 10, 2021. This work was supported in part by the National Key R&D Program of China 2020YFB1807803 and 2020YFB1807800, in part by the National Natural Science Foundation of China under Grants 62071067, 62001054, 61872082, 61771068 and 61472184, and in part by the National Postdoctoral Program for Innovative Talents under Grant BX20200067, and in part by the Ministry of Education and China Mobile Joint Fund MCM20180101, and in part by the Beijing University of Posts and Telecommunications-China Mobile Research Institute Joint Innovation Center. The associate editor coordinating the review of this article and approving it for publication was M. Tortonesi. *(Tianjian Dong and Qi Qi are co-first authors.) (Corresponding authors: Jingyu Wang; Alex X. Liu.)*

Tianjian Dong, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, and Jianxin Liao are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China, and with EBUPT.COM, Beijing 100191, China (e-mail: dongtianjian@bupt.edu.cn; qiqi8266@bupt.edu.cn; wangjingyu@bupt.edu.cn; hfsun@bupt.edu.cn; zhuangzirui@bupt.edu.cn; liaojx@bupt.edu.cn).

Alex X. Liu is with the Department of Computer Science and Technology, Qilu University of Technology, Jinan 250353, China (e-mail: alexliu360@gmail.com).

Digital Object Identifier 10.1109/TNSM.2021.3077249

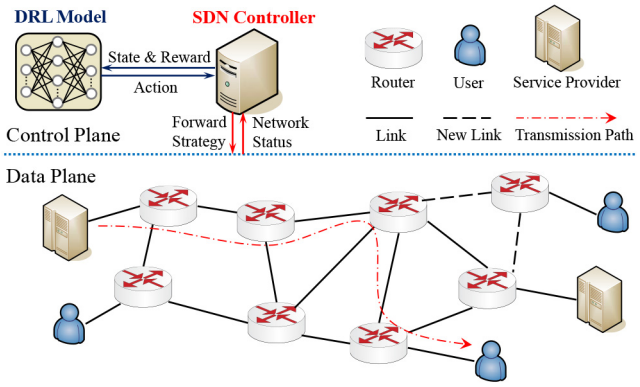


Fig. 1. The overview of DRL-based routing. SDN controller collects network status, communicate with DRL model, and installs routing strategies with intelligent decision-making. However, the variable topology and highly dynamic traffic will greatly affect the performance of DRL-based routing.

I. Introduction

A. Background and Motivation

ROUTING is a crucial function of the communication network and mostly determines the performance of networking. Software Defined Networking (SDN) unveils new capabilities in routing, which decouples the control-plane and data-plane for centralized control and provides a global observation of network status. Emerging networks, such as the Internet of Things (IoT) [1], vehicular networks [2], and smart grids [3], often have highly dynamic network status, large-scale traffic demand, and heterogeneous optimization objectives. For example, multimedia communications usually have multiple needs for QoS guarantees (e.g., throughput, latency, and packet loss), while the transmission requirements become more stringent with a rapid growth of video services [4]. The increasing complexity in emerging networks incurs higher requirements on the SDN-based routing. However, there are three problems in traditional routing methods under the SDN architecture. First, the state-oblivious methods such as shortest path usually follow a fixed control strategy. This characteristic makes them unable to utilize the observability and flexibility provided by SDN architecture to deal with the highly dynamic runtime network status [5]. Second, the heuristic-based methods are typically computationally expensive. When

facing with highly dynamic network status, these methods are difficult to make real-time routing decisions due to this limitation. In addition, since large-scale traffic demand often leads to the increasing of routing requests, heuristic methods will also greatly raise the burden of SDN controller [6]. Third, the optimization-based methods require expert knowledge to establish mathematical models for specific problems. However, it is difficult to predict, model, and control the complicated emerging networks [7].

Recently, learning-based algorithms have been used as promising solutions to various network optimization [8] and network visibility [9] problems. Deep Reinforcement Learning (DRL) is a learning-based approach, which automatically makes decisions, observes the environment, and improves its policy to achieve the optimum. Generally, DRL provides the following advantages [10] that makes it suitable for SDN routing. Firstly, DRL-based routing (shown in Fig. 1) has more flexibility than conventional routing methods because it is an autonomous trial-and-error process that adjusts the routing strategy based on current observed status. Secondly, compared with the heuristics, DRL has a faster decision-making with sophisticated status information. Network status is high-dimensional and time-variant, which includes a huge number of runtime statistics and varies with the service requirements. By training several neural networks, DRL can effectively learn the relationship between network status and routing strategy. Thirdly, DRL mainly depends on the experiences (i.e., the observations through the interaction with environments) obtained by network measurements, without the need of a complete or accurate model. Finally, the intelligence of DRL algorithms will be largely unleashed by the vigorous controllability and broad observability of SDN architecture [11]. In summary, SDN routing will preliminarily satisfy the complicated and dynamic requirements of emerging networks with the help of DRL algorithms.

B. Limitations of Prior Art

Almost none of prior DRL-based routing methods [7], [12]–[15] have addressed the important problem: how to train a new DRL model for a new network environment when network conditions have been changed. Intuitively, the changes in network conditions include the two aspects: network status distribution[1] and topology. For example, the spatial distribution of user traffic demand in mobile networks during different time periods is uneven [16], which will lead to great variations in the distribution of network status. In addition, due to link failures or network reconfiguration, the network topology will change over time [17]. Hence, there are three main challenges for DRL-based routing methods to solve the varying network condition problem:

1) *The Varying Network Status Distribution:* The network status is the input of DRL-based routing model. Since the neural networks perform well only if the input data follow the same distribution (i.e., the identical distribution assumption), the changed network status distribution will greatly affect their precision.
2) *The Variable Network Topology:* With the change of topology, the state and action spaces of DRL model will be different, i.e., the input and output dimensions of neural networks will be changed. The two dimensions are the hyper-parameters of DRL model, which cannot be changed once the training process of model is started. Hence, the neural network of DRL model cannot process the input and output data that have the varying dimension.
3) *The Requirement on the Convergence Speed:* In the emerging networks, the quality and stability are important for their services and applications [18], which need the online learning-based algorithm to reach a rapid convergence. For instance, the Tactile Internet services such as tele-surgery and industrial process automation mostly require a high level of reliability to function correctly and safely [19]. In addition, a rapid convergence not only achieves a stable service provisioning, but also provides a better network performance.

Since the three challenges are ubiquitously coexist in the emerging networks, they are crucial to the network routing and should be addressed simultaneously. In general, the varying network condition problem has three solutions: 1) still using the well-trained DRL model from the previous network environment; 2) learning a new DRL model from scratch in the new environment; and 3) fine-tuning the DRL existing model in the new environment. However, since the straightforward methods cannot fully addressed the above three challenges, they still present the following limitations:

1) *The first method cannot address the first and second challenges because of the identical distribution assumption and fixed dimension of neural networks:* Since the first method has no need to train a new model in a new environment, it has no concern on the convergence speed. However, it cannot address both the varying network status distribution and the variable topology. On the one hand, if the previous DRL model is leveraged to the new environment without any modification, its performance will drop seriously due to the difference between the data distribution of previous and new environments. This issue can be clarified in Fig. 2, which shows the experiment of keeping using the previous DRL model when facing varying network status distribution. On the other hand, the DRL model without any modification cannot process the data with different dimensions.
2) *The second method cannot address the third challenge due to the long convergence time during the online training of DRL model:* By retraining a new model from scratch, the second method can cope with the first and second challenges. However, the sample complexity of DRL is related to its state and action spaces [20], which are proportional to the network size. With the growth of network size (e.g., the increasing of network devices, links or servers), DRL algorithms usually require a huge

[1]The varying network status refers to the changes in runtime statistics due to the variation in short-term traffic demand or routing decisions. And the varying network status distribution mainly refers to the change due to the variation in long-term traffic distribution or the varying link status in a long timescale.

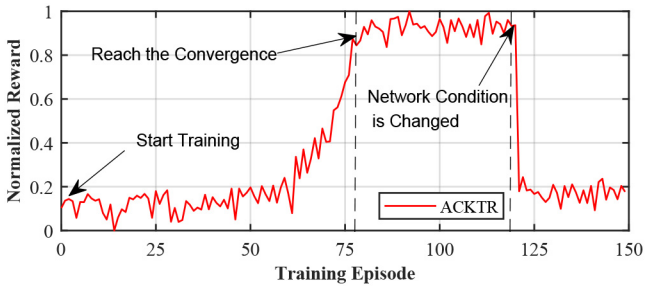


Fig. 2. The effect of varying network status distribution on DRL training. Firstly, the DRL model is trained to reach the convergence at an initial network environment. Then we increase the link delay of the shorter paths in the network to change the network status distribution. After that, the reward of DRL model plummets.

sampling cost to achieve convergence [21]. Thus, it is unacceptable for services that have stringent requirements, i.e., the huge cost of retraining a DRL model is not suitable for the highly dynamic emerging networks.

3) *The third method is not effective enough to address the third challenge since it does not utilize the knowledge from previous and new environments at the same time:* The third method is another retraining algorithm that is more reasonable than the second method since it will make a minor adjustment to the model based on the new environment. However, it lacks of efficiency since it only uses the data in the new environment during the training process. The sample complexity will be increased when the data of previous environment is unavailable [22]. Therefore, both old knowledge and new experience are important for learning to adapt to a new environment.

C. Proposed Approach

In this paper, we propose the *GAN-based transfer reinforcement learning*, an online, adaptive, and efficient transfer reinforcement learning approach for network routing, which integrally tackles the three challenges. To address the first and second challenges, we adopt the *transfer reinforcement learning* [23] method into DRL-based routing framework to cope with the variable network environment. As we mentioned above, it is prohibitively difficult for DRL-based routing to deal with the significantly changed network status distribution and variable topology, i.e., the previous model in the source task (routing in the previous network) cannot be straightforwardly utilized in the target task (routing in a new network). Our transfer reinforcement learning approach takes the advantages of *Domain Adaptation* (a kind of transfer learning methods) [24], which can handle these shortcomings of DRL-based routing in a new environment. When topology or network status distribution has changed, this method will adapt the previous model to a novel environment, instead of retraining a model with a huge cost. Furthermore, the model trained from a specific network environment can be reused in another environment with the help of this method.

To address the third challenge, we integrate the Generative Adversarial Network (GAN) [25] into our transfer reinforcement learning framework. The fine-tune method is a potential solution to the varying network condition problem, which trains the neural networks of DRL on the target task solely with the parameters learned from the source task. However, this method is unsuitable for stringent routing requirements in emerging networks, since it cannot effectively reduce the sample complexity. Therefore, we utilize GAN to learn a mapping that transforms the source and target domains into a common feature space (this approach is formally called *Adversarial Domain Adaptation*). Through this approach, the common feature representations will be domain-invariant, since they follow the same distribution regardless of whether the features are generated in the source domain or the target domain. By using the common features, the previous model can reach a fast convergence in the target task.

In summary, when the network environment has changed, we firstly reuse the well-trained model from the source task by the transfer reinforcement learning approach. Secondly, we leverage the GAN architecture to effectively map network status of the source and target tasks into domain-invariant features. Then the previous model can automatically make decisions by using these features as input data. Since the features of source and target domains follow the same distribution, the DRL model can easily reach the optimum (e.g., in terms of latency or throughput) in the target task rather than training a new model with a large sampling cost.

D. Technical Challenges and Solutions

There are three major technical challenges to fully implement our routing method with GAN-based transfer reinforcement learning. The first challenge is to formulate the DRL paradigm in the SDN routing scenario. If the definition of DRL elements (i.e., *state*, *action*, and *reward*) is not suitable for routing, the DRL algorithm can not achieve the optimal network performance. To cope with this challenge, we give a detailed formulation of DRL elements, which comprehensively considered the network statistics and the goal of routing.

The second challenge is to make the neural networks of DRL model to fit the target task, since the input and output layers of neural networks are respectively related to the dimensions of state and action spaces. To cope with this challenge, we first add feature encoders in front of the whole DRL model in different domains. The output dimensions of the feature encoders of source and target domains are fixed to generate features with the same size, while their input dimensions are different for adapting to various state spaces. Then we initialize an output layer with a different dimension, in order to fit the new action space in the target task.

The third challenge is to implement our transfer reinforcement learning-based routing in SDN architecture. To be specific, we need to realize the procedure that applies routing strategies based on network status. We first design a network monitor to measure the real-time remaining bandwidth and delay of network links. Then we develop a path allocator to install routing strategies based on the control policy of DRL. The transfer learning-enabled DRL model is integrated into the SDN controller, for autonomous decision-making in variable network environments. Our system is implemented into

TABLE I
COMPARISON AMONG RELATED WORKS

Reference	Method	Key features
OSPF [29], MPLS [30], and Segment Routing [31]	Traditional rule-based algorithms	Short of flexibility and difficult to find the optimal solution with little expenses
SDN-WAN [32], [33], and SMORE [34]	Optimization or heuristic algorithms	Highly rely on an accurate mathematical model and are time-consuming
[35], [36], RouteNet [37], and GADL [38]	Deep Belief Architectures, Convolutional Neural Networks, and Graph Neural Networks	Effective in modeling complex relationship between input and output data but not suitable enough for routing that does not have a direct label for every decision-making
DRL-TE [7] and AuTO [12]	Deep Deterministic Policy Gradient	Mainly focus on the design of state/action space and reward function for routing/traffic engineering but unaware of the variable network topology
DRL-MTC [13]	Deep Deterministic Policy Gradient	Trained on a specific network environment and can not be transferred into another scenario
DQR [14] and FE-DRL [15]	Deep-Q-Network and Trust Region Policy Optimization	Mainly focus on a customized feature representation and incapable of the topology change problem

Mininet [26], a real-world platform equipped with production level software switches, which enables several SDN elements.

E. Contributions and Organization

The proposed GAN-based transfer reinforcement learning can handle the network routing in variable environments. It not only effectively deals with the changeable topology, but also performs well in varied distribution of network status. We summarize our contributions in the following.

1) We utilize DRL model into SDN routing problem and propose a transfer reinforcement learning approach for routing with the significantly changed network status distribution and the variable topology of network.
2) For enhancing the transfer efficiency, we utilize GAN-based domain adaptation as our transfer learning method to learn a domain-invariant features of different networks.
3) We integrate our transfer reinforcement learning algorithm into the network controller and realize the SDN-based routing system.
4) We evaluate our algorithm both on variable topology and network status distribution scenarios. The experimental results show the proposed method largely outperforms the state-of-the-art DRL-based routing frameworks and the naive transfer method (that is based on fine-tune method).

The remainder of the paper is organized as follows. In Section II, we review the related works. In Section III, we formulate the routing problem. In Section IV, we introduce the DRL-based routing and elaborate our GAN-based transfer reinforcement learning algorithm. In Section V, we focus on the implementation of our system. In Section VI, we present our experiment settings and results. Finally, Section VII concludes the paper.

TABLE II
NOTATION DEFINITION

Variable	Definition
$\mathcal{F}^{(k)}$, K	A flow set and the number of flow sets
$\mathcal{P}^{(k)}$	The set of candidate paths of flow set $\mathcal{F}^{(k)}$
$B_l(\tau)$	The remaining bandwidth of link l at τ
$T_l(\tau)$	The delay of link l at τ
$x_{pf}(\tau)$	The path selection result of flow f
$B_p(\tau)$	The remaining capacity of transmission path p at τ
$D_f(\tau)$	The traffic demand of flow f at τ
$H_p(\tau)$	The end-to-end throughput of transmission path p at τ
$T_p(\tau)$	The end-to-end latency of transmission path p at τ
$\mathcal{S}^s$, $\mathcal{S}^t$	The source and target state space of routing
$\mathcal{A}$, $\mathcal{R}$	The action space of routing and the reward function.
$\boldsymbol{s}^s$, $\boldsymbol{s}^t$	The source and target state vector of routing
$\boldsymbol{h}^s$, $\boldsymbol{h}^t$	The source and target feature vector
a^s, a^t	The source and target action of routing
$\pi(a\|\boldsymbol{s})$	The policy of selecting action a under state $\boldsymbol{s}$
$\Re_\tau$	The accumulated reward at τ
$A^\pi(\boldsymbol{s}, a)$	The advantage function under the policy π
$V(\boldsymbol{s})$	The evaluated value of state $\boldsymbol{s}$
$\mathcal{G}_s$, $\mathcal{G}_t$	The source and target feature encoder
$\mathcal{D}$	The domain classifier

II. RELATED WORKS

A. Routing Techniques in Networks

With the emerging development of SDN, the centralized controller gains higher capability for networking. For example, OpenFlow [5] and P4 [27] provides more power on controlling, as well as more accurate network status is available for the controller by recent network measurement advancements [28]. These evolvements pave the way for more effective routing strategy to fulfill the potential of SDN architecture. We summarize the related works about routing in Table I.

Traditional routing method mainly refers to Open Shortest Path First (OSPF), which aims to find a transmission path with minimum summed weight by exchanging the link-state between routers. Nevertheless, this protocol is short of flexibility and has a slow convergence [29]. Multi-Protocol Label Switching (MPLS) [30] and Segment Routing [31] are used for better traffic engineering, but they are difficult to find the optimal solution with little expenses in time. Some recent works like [32]–[34] utilized optimization or heuristic methods in routing for higher adaptivity. Unfortunately, it is difficult to formulate good heuristic rules that match actual problems. And

these methods usually have an expensive computation cost, so that they are too time-consuming to be applied in real-time systems.

The development of deep learning algorithms provide a breakthrough on network routing. They have outstanding advantages over conventional methods, since they are effective in modeling nonlinear, deep, and complex relationship between input and output data. Mao *et al.* [35] used a supervised deep belief architectures to predict the next hop router, which is a distributed method and takes traffic pattern as the input. Tang *et al.* [36] constructed an input characterization by convolutional neural networks, in order to enhance the performance of wireless networks. Rusek *et al.* [37] proposed a graph neural network-based structure to produce accurate predictions of network statistics, and these predictions are used for achieving efficient routing optimization. Zhuang *et al.* [38] proposed a graph-aware deep learning model for learning intelligent routing strategy, which learns topological information efficiently. However, the network routing usually does not have a direct label for every decision-making, and it tends to take the optimal policy that contains a series of decisions to maximize the long-term return. This characteristic is suitable for DRL algorithms, which learn a control policy by trial-and-error solely from rewards of the environment. Many works have enhanced the capability of routing strategy by utilizing various DRL algorithms. Nevertheless, they all ignore the network variation problem, which means they can not effectively handle the topology change and dynamic distribution of network status at the same time. DRL-TE [7] uses the throughput and latency of sessions as the state space of DRL, and the action space is designed as the split ratio of candidate paths. This mechanism is highly related to the connectivity of forwarding nodes. When network topology has changed, the DRL model can not be utilized in a new environment any longer. AuTO [12] mainly aims to tackle the scalable traffic problem by DRL in networks, and does not take the variable topology into consideration. DRL-MTC [13] is inefficient with the large-scale traffic variation, since its reward function is constructed by a Quality-of-Experience (QoE) evaluation neural network, which is trained on a specific environment and can not be transferred into another scenario. Due to the huge cost of retraining and low adaptivity, this method is unsuitable for variable network environment. In addition, DQR [14] and FE-DRL [15] focus on how to customize an effective feature representation for DRL-based routing. But they are incapable of the device increment or connectivity change problems since their state and action spaces are related to the topological structure of network.

B. Transfer Learning Methods

Transfer learning has become a hot research topic that aims to apply knowledge and skills learned in previous domains to novel domains, which is mainly applied in computer vision [39] and natural language processing [40]. According to the survey [24], transfer learning can be divided into three main categories (i.e., inductive transfer learning, transductive transfer learning, unsupervised transfer learning). Domain adaptation method is a kind of transductive transfer learning methods, which are used for the case that the source and target domains are different but with the same objective. Sun *et al.* [41] combined traffic classification with domain adaptation techniques to transfer knowledge between tasks, and obtained good performance. Bartos *et al.* [42] used domain adaptation to learn invariant features of network traffic, in order to enhance the precision in malware detecting. Recently, adversarial domain adaptation [25] has become a major solution to the domain shift problem, which integrated GAN into domain adaptation approach. The GAN contains a generator network and a discriminator network. The goal of generator is to create target features that are close to the source features, while the discriminator aims to classify which domain the features are created from. Through a zero-sum game-based training, the GAN model can generate domain-invariant features, which can fit different domains.

Recently, there is a trending interest in leveraging transfer learning method into reinforcement learning approach, and these works have gained promising performance. Taylor and Stone [23] reviewed the transfer reinforcement learning algorithms and detailed their categorization. These methods attempt to utilize the transfer learning techniques for speeding up the training procedure in a novel environment. Mo *et al.* [43] transferred learned knowledge from the source domain to the target domain, and used personalized Q-function to avoid negative transfer. Li *et al.* [44] designed a transfer learning-enabled Actor-Critic method to learn the knowledge of neighboring regions, which achieved a significant energy efficiency improvement. Besides, Comşa *et al.* utilized the state space compression mechanism to tackle the problem of having a variable dimension for the state space [45], [46]. Wei *et al.* [47] proposed the discrete branching dueling Q-network to compression the action space of network slice reconfiguration problem, which can decrease the number of actions. However, works [45]–[47] cannot handle both varying state and action spaces at the same time. To the best of our knowledge, none has leveraged the transfer reinforcement learning method to tackle the limitations in DRL-based routing, and we are the first to tackle the varying state and action spaces of routing with the transfer method. Inspired by the works above, we design a transfer reinforcement learning approach to cope with the domain shift problem in network routing.

III. Problem Formulation

We consider a communication network that is represented by an undirected graph $G = (V, E)$, where V is the set of nodes (switches or routers) and E is the set of links, where $l = (u, v) \in E$ and $u, v \in V$. Each link l is associated with a remaining bandwidth $B_l(\tau)$ and a delay $T_l(\tau)$ at time slot τ. Note that $B_l(\tau) = C_l - B_l^r(\tau)$, where C_l is the bandwidth capacity of link l and $B_l^r(\tau)$ is the flow rate of link l at τ. There are K flow sets $\mathcal{F}^{(k)}$, $k = 1, \ldots, K$. Each flow set is identified by a tuple $\langle src_k, dst_k \rangle$, where src_k and dst_k are the source and destination nodes, which means all flows in the flow set $\mathcal{F}^{(k)}$ have the same source-destination

nodes pair. Each flow $f \in \mathcal{F}^{(k)}$ is identified by the 5-tuple: source/destination IP, source/destination port numbers, and transport protocol [5], i.e., a single flow is identified by both the tuple $\langle src_k, dst_k \rangle$ and its transport layer protocol.

In a communication network, each flow $f \in \mathcal{F}^{(k)}$ should be routed through a single path $p \in \mathcal{P}^{(k)}$, where $\mathcal{P}^{(k)}$ is the set of candidate paths for flow set $\mathcal{F}^{(k)}$. Since all flows in $\mathcal{F}^{(k)}$ have the same source-destination pairs, the path set $\mathcal{P}^{(k)}$ only contains paths that have the same source and destination nodes. We take a binary variable $x_{pf}(\tau)$ to denote the result of path selection, where $x_{pf}(\tau) = 1$ if the flow f selects transmission path $p \in \mathcal{P}^{(k)}$ at time slot τ, and $x_{pf}(\tau) = 0$ otherwise. Each flow will be only routed by one path at a given time slot, so that we have the constraint:

$$\sum_{p \in \mathcal{P}^{(k)}} x_{pf}(\tau) = 1, \ \forall f \in \mathcal{F}^{(k)}, \ k = 1, \ldots, K, \tag{1}$$

$$x_{pf}(\tau) \in \{0, 1\}, \ \forall p \in \mathcal{P}^{(k)}, \ \forall f \in \mathcal{F}^{(k)}, \ k = 1, \ldots, K. \tag{2}$$

Besides, at time slot τ, given the traffic demand $D_f(\tau)$ of flow f , the bandwidth limitation of a path cannot be exceeded:

$$\sum_{f \in \mathcal{F}^{(k)}} D_f(\tau) x_{pf}(\tau) \leq B_p(\tau), \ \forall p \in \mathcal{P}^{(k)}, \ k = 1, \ldots, K, \tag{3}$$

where $B_p(\tau) = \min_{l \in p} B_l(\tau)$ is the remaining bandwidth of path p. Routing is to find the optimal path between source and destination nodes for transmitting traffic flows. Before the formulation of the routing problem, we firstly give the network utility function of flow set $\mathcal{F}^{(k)}$:

$$U_k(\tau) = \sum_{f \in \mathcal{F}^{(k)}} \sum_{p \in \mathcal{P}^{(k)}} x_{pf}(\tau) \big[\alpha \cdot H_p(\tau) - \beta \cdot T_p(\tau)\big], \tag{4}$$

where $T_p(\tau) = \sum_{l \in p} T_l(\tau), \ p \in \mathcal{P}^{(k)}$, which is represented the end-to-end latency of path p at time slot τ, $H_p(\tau)$ is the throughput of path p at τ, α and β are constant coefficients determined by specific requirements of traffic flows. This utility function aims to maximize the total throughput of all flows and minimize the end-to-end latency simultaneously.

Then the routing problem can be formulated as:

$$\begin{aligned} \max \ & \sum_{\tau=0}^{\mathcal{T}} \sum_{k=1}^{K} U_k(\tau), \\ s.t. \ & (1)(2)(3). \end{aligned} \tag{5}$$

IV. Methodology

A. Deep Reinforcement Learning-Based Routing

There are three main challenges to solve the routing problem formulated in Eq. (5). Firstly, the end-to-end latency is hard to be modeled and estimated, which includes process time, queue time, and transmission time, since an accurate mathematical model does not exist to formulate the relationship between end-to-end latency and the other variables (e.g., bandwidth) [7]. Secondly, the network status is high-dimensional, and the relationship between status and routing strategy is nonlinear. Conventional methods are usually inefficient in problems with these characteristics. Thirdly, the objective of routing is to obtain the long-term optimal solution, so that the greedy method looking for maximizing an immediate reward is not suitable for addressing the routing problem above since it usually leads to the local optimum. Therefore, a better method is supposed to address these challenges.

Because routing is a decision-making process based on the current network status, it can be described as a Markov Decision Process (MDP). Even though there are many challenges in the routing problem of emerging networks, it can be solved by DRL algorithms for the following three reasons. Firstly, DRL algorithms are used to learn a decision-making policy for the scenario of MDP without an accurate model for network statistics, which makes it suitable for solving the routing problem. Secondly, DRL algorithms take the advantage of the deep neural network, which is a well-known algorithm to establish high-dimensional and nonlinear relationships. Thirdly, the goal of DRL algorithms is to find a control policy that maximizes the long-term return rather than makes a decision for local optimum. In general, MDP is characterized by a 5-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space of the agent, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ represents the transition probability of MDP, and $\gamma \in (0, 1)$ is the discount coefficient. Then we introduce the definition of our DRL-based routing.

Agent: The controller can be regarded as the DRL agent since it is the brain of the whole communication network, which works in an event-driven manner (The detailed workflow of the controller can be seen in Section V). Hence, in each decision epoch, the agent only selects one path for the flow to which the current packet belongs. In other words, the controller does not choose paths for multiple flows at the same time, but sequentially makes route planning in the order at which they arrive. When the corresponding path is deployed in the network, the agent will receive a reward from the environment. After a certain number of decision epochs, the agent will accumulate the rewards acquired from each epoch, then utilize them to update its model. Hence, this packet-driven mechanism determines the definition of state space, action space, and reward function.

State Space: Since routing aims to find the optimal path for a traffic flow according to current network status and flow characteristics, the state space of routing is defined as:

$$\boldsymbol{s} = \big[B_1, \ldots, B_{N_E}, T_1, \ldots, T_{N_E}, src_f, dst_f, D_f\big], \tag{6}$$

where B_n and T_n represent the remaining bandwidth and delay of the *n*-th link in the network at this decision epoch respectively, N_E is the number of links, src_f, dst_f, and D_f are the source node, destination node, and traffic demand of flow f that is requesting a path respectively.

Action Space: When a traffic flow is requesting a route, the DRL agent will select a path for transmitting it. Therefore, the action a is defined as the index of a candidate path, and the size of action space is determined by the size of path sets $\mathcal{P}^{(k)}, \ k = 1, \ldots, K$. By utilizing the k-shortest path algorithm, we give N_P candidate paths for each flow set $\mathcal{F}^{(k)}$ (i.e.,

the size of candidate path set of all flow sets is the same), so that the action space is expressed in the following:

$$\mathcal{A} = \{a \mid a = 1, 2, \ldots, N_P\}. \tag{7}$$

Due to our packet-driven mechanism, an action at one decision epoch refers to a single path for one arrived flow.

Reward Function: A reward, as a feedback, will be given to the DRL agent based on the current state of network environments. Hence, the reward function should be designed according to the objective of routing (seen in Eq. (4)). Therefore, it is defined as the total utility of all the flow sets:

$$\mathcal{R}_\tau = \sum_{k=1}^{K} \sum_{f \in \mathcal{F}^{(k)}} \sum_{p \in \mathcal{P}^{(k)}} x_{pf}(\tau) \big[\alpha \cdot H_p(\tau) - \beta \cdot T_p(\tau)\big], \tag{8}$$

where $H_p(\tau)$ and $T_p(\tau)$ are the throughput and latency of the chosen path p at time slot τ respectively, as well as α and β are constant coefficients.

At a time slot τ, the SDN controller is served as the DRL agent to collect the state $\boldsymbol{s}_\tau$, select an action a_τ based on its policy $\pi(a|\boldsymbol{s}) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ and get the corresponding reward $\mathcal{R}_\tau$ from the network environment. The policy function is a conditional probability distribution that maps the state space onto the action space. This function works as the core of DRL since it is the rule of the decision-making of DRL agent. The objective of a policy at time slot τ is to maximize the accumulated reward:

$$\mathfrak{R}_\tau = \sum_{k=0}^{\infty} \gamma^k \mathcal{R}_{\tau+k}, \tag{9}$$

where $\gamma \in (0, 1)$ is the discount coefficient that trades off the weights of historical and current reward data for the policy.

The expected return after taking action a in state $\boldsymbol{s}$ in the *Actor-Critic* method [48] is defined as:

$$A^\pi(\boldsymbol{s}, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}_{\tau+k} | \boldsymbol{s}_\tau = \boldsymbol{s}, a_\tau = a \right] - V(\boldsymbol{s}), \tag{10}$$

where $V(\boldsymbol{s})$ is the mathematical expectation of the evaluated values of all possible actions in state $\boldsymbol{s}$. We can see that $A^\pi(\boldsymbol{s}, a)$ denotes the superiority of the chosen action a among other actions at the state $\boldsymbol{s}$. In DRL paradigm, the policy is usually represented by a neural network and its parameters θ. Then we can utilize the policy gradient method to optimize the policy according to the gradients of its parameters [49]:

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{\tau=0}^{\infty} \nabla_\theta \log \pi_\theta(a_\tau | \boldsymbol{s}_\tau) A^{\pi_\theta}(\boldsymbol{s}_\tau, a_\tau) \right]. \tag{11}$$

In *Actor-Critic* method [48], there are an *Actor* network to generate policy $\pi(a|\boldsymbol{s})$ and a *Critic* network to estimate the evaluated value $V(\boldsymbol{s})$. The Actor network is updated by Eq. (11), and the Critic network is updated by the temporal-difference error of $V(\boldsymbol{s})$. ACKTR [50] is proposed to reduce the sampling cost and computational complexity of Actor-Critic by solving the inverse of Fisher information matrix, which will accelerate the updating process of Actor-Critic. It has achieved better performance than current DRL algorithms like Asynchronous Advantage Actor-Critic (A3C) [51] and Trust Region Policy Optimization (TRPO) [52]. Thus, we will use ACKTR as our baseline algorithm.

B. Domain Shift Problem in DRL-Based Routing

DRL algorithms with extraordinary modeling power and adaptive decision-making, have successfully improved the performance of routing [7], [12]–[15]. However, there are still many shortcomings in prior DRL-based routing. Firstly, DRL is difficult to find the optimal solution when the network status distribution changes greatly. In general, there is an important condition for neural networks to perform well, that is, the input data collected by DRL agent follow the same distribution. Thus, the model trained in the source domain can not give a good routing strategy in the target domain, resulting the decline of network performance. Secondly, if the network topology has changed, the number and connectivity of switches or routers will be varied, so that the dimensions of state and action spaces will also be different. The dimensions of input and output layers are fixed since the number of nodes in neural networks should be initialized before training. Therefore, the previous model cannot process the state vector with a different dimension in a new network even the change is slight. Thirdly, to cope with the problems above, we have to retrain the DRL model in a new network environment. However, the high sampling cost of DRL algorithms leads to a slow convergence speed, so that it is impractical to retrain the model in the highly dynamic emerging networks. Moreover, the retraining process will also completely waste the knowledge learned from the previous environment.

Transfer learning methods have achieved good results in many problems using supervised learning [39], [40], which adapt the previous model in new scenarios for reducing the cost of retraining. Likewise, transfer reinforcement learning is proposed to utilize the prior knowledge (e.g., the model trained on previous tasks) in the DRL paradigm. Inspired by the advantages of transfer learning methods, we can utilize the transfer reinforcement learning into the routing problem with variable topology and network status distribution, in order to tackle the shortcomings above. In general, transfer reinforcement learning methods are classified into three main categories in light of the relationship between source domain and target domain, source task objective and target task objective [23]. Specifically, the objective of routing in different topologies or network status distributions is the same, while their domains are different. Since there are some relationships in different topologies and network status distributions, the *Domain Adaptation* methods can effectively deal with this problem. For aligning different domains, domain adaptation methods will initialize the corresponding feature encoder, which is trained to fit the previous model (the training algorithm is described in the next subsection). After that the well-trained model from the source task can be reused in target tasks. In fact, it is not advisable to completely abandon a well-trained model.

With the help of domain adaptation, the DRL-based routing approach can be adjusted to fit various network environments.

Then we briefly introduce the training approach when adding the domain adaptation. In the source task, a feature encoder will be added in front of the DRL model and trained as a part of it. When the environment has changed, we only replace the feature encoder of the source domain with a new encoder of the target domain. Then we reuse the remaining part of the well-trained model, and train the feature encoder on the target domain. Note that the training of feature encoder and the decision-making of DRL are simultaneous (both of them are online approach). In this paper, we leverage ACKTR as the baseline of DRL-based routing, which is based on the Actor-Critic model. Hence, we discuss which part of Actor-Critic model should be modified for utilizing the domain adaptation into DRL-based routing. The detailed analyses are as follows.

- *Input Layer of Actor and Critic:* In the DRL-based routing paradigm, the state vector is the input of Actor network and Critic network. To cope with the variable environment, we add the feature encoders in front of the Actor network in source and target domains. On the one hand, the two encoders share the same output dimension but are different in the input dimension, in order to cope with the changed topology. They can process the state vector with different dimensions in various domains. On the other hand, for handling the variable network status distribution, the two encoders are trained respectively in source and target domains. This mechanism can adapt the DRL model to various distributions of input data.
- *Output Layer of Actor:* The output of Actor network is an action index that indicates the transmission path. The dimension of Actor's output layer is determined by the topology according to the definition of DRL-based routing. With the change of topology, the dimension of action space (i.e., the size of path set) will be varied. Therefore, we can fix the front layer of Actor network and train an new output layer with a different dimension in the target task.
- *Output Layer of Critic:* The Critic's output is an evaluated value of the current state. This value is irrelevant to the topology or the network status distribution. Hence it has no need to be changed.

Based on the analysis of model modifications above, we will detail the proposed algorithm in the next subsection.

C. GAN-Based Transfer Reinforcement Learning

In the previous section we introduce the domain adaptation method in routing with variable network environments. However, the conventional training algorithm for the target encoder, mainly refers to the fine-tune method, has less power on accelerating the convergence speed since it lacks of utilizing the source feature encoder to training the target feature encoder. To tackle this problem, we need to design an effective training method for the feature encoder in target domain.

The *Generative Adversarial Network* (GAN) is a well-known deep learning model, which consists of two neural networks, the *Generator* and the *Discriminator*. The entire model is trained by a zero-sum game between the Generator and Discriminator in order to generate samples from training data that meets the specific requirements. Generator is seeking to generate samples for fooling the Discriminator as much as possible. The Discriminator plays the role of domain classifier, aiming to distinguish the samples generated by the Generator from the real samples. Their objectives are defined as:

$$\max_{\mathcal{D}} V(\mathcal{D},\mathcal{G}) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_{data}(\boldsymbol{x})}[\log \mathcal{D}(\boldsymbol{x})] + \mathop{\mathbb{E}}_{\boldsymbol{z}\sim P_z(\boldsymbol{z})}[\log(1-\mathcal{D}(\mathcal{G}(\boldsymbol{z})))], \quad (12)$$

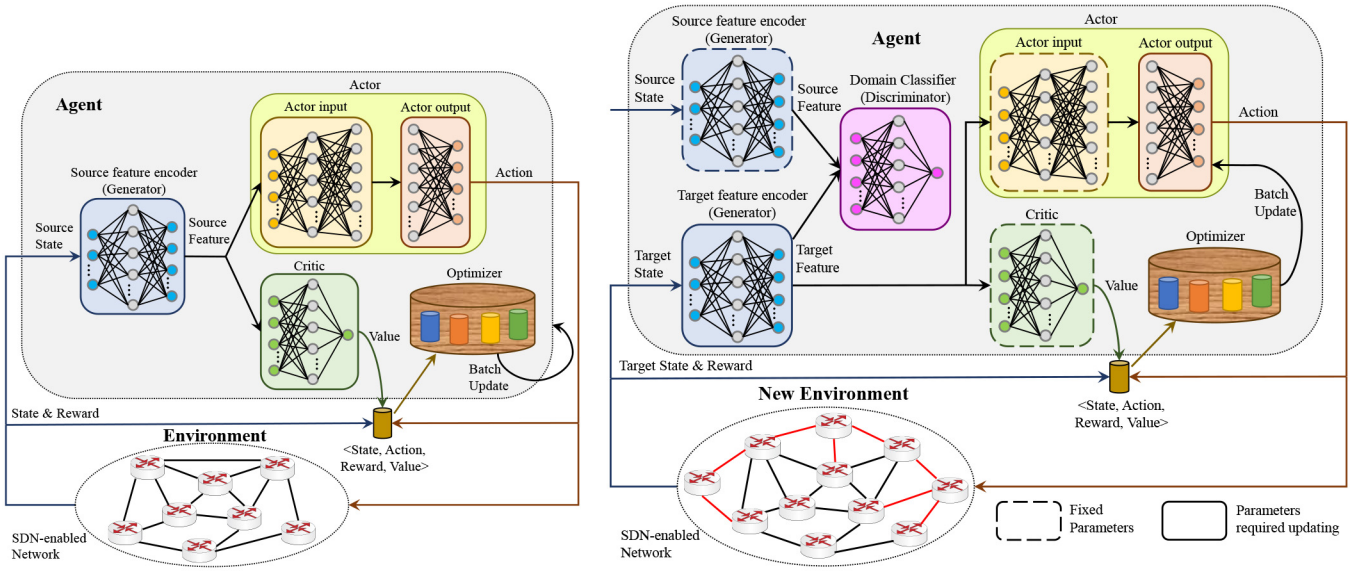
$$\min_{\mathcal{G}} V(\mathcal{D},\mathcal{G}) = \mathop{\mathbb{E}}_{\boldsymbol{z}\sim P_z(\boldsymbol{z})}[\log(1-\mathcal{D}(\mathcal{G}(\boldsymbol{z})))], \quad (13)$$

where $\boldsymbol{x}$ is the sample from the real distribution (e.g., the sample from dataset) and $\mathcal{G}(\boldsymbol{z})$ is the fake sample generated from the sample $\boldsymbol{z}$ of another distribution. In summary, the GAN model is a minimax optimization to generate samples that are close to the given distribution.

Adversarial Domain Adaptation [25] is a promising transfer learning algorithm, which integrates GAN into domain adaptation approach. It learns a mapping to transform the network status in source and target domains into a common feature (i.e., *transferring knowledge of feature representation*). The features share the same distribution, so that they are invariant in different domains. After the mapping, the data in the target domain will be converted into the domain-invariant feature, which are suitable for the model trained in the source domain. Formally, the aim of adversarial domain adaptation is to learn the common feature representation of the source state space $\mathcal{S}^s$ (i.e., source domain) and target state space $\mathcal{S}^t$ (i.e., target domain). In our GAN-based transfer approach, we have a source feature encoder $\mathcal{G}_s$, a target feature encoder $\mathcal{G}_t$, a domain classifier $\mathcal{D}$, *Actor* network and *Critic* network. Our algorithms can be divided into two stages.

Source Task: We use the conventional DRL algorithm to train the whole model, including $\mathcal{G}_s$, *Actor* and *Critic*. As depicted in Fig. 3(a), $\mathcal{G}_s$ is used to generate source feature $\boldsymbol{h}^s$ for *Actor* and *Critic*. Then the two components can utilize $\boldsymbol{h}^s$ (instead of $\boldsymbol{s}^s$) to give an action a^s and the evaluated value $V(\boldsymbol{h}^s)$. When the network environment has changed (either topology or network status distribution), we will reuse the well-trained model from the source task to the target task. Specifically, the parameters of $\mathcal{G}_s$, *Actor* and *Critic* will be fixed, i.e., the parameters of these three components will be copied from the source task to the target task, and their parameters will never be updated during the target task.

Target Task: Similarly, as depicted in Fig. 3(b), $\mathcal{G}_t$ is used to generate target feature $\boldsymbol{h}^t$ for *Actor* and *Critic*. Since *Actor* and *Critic* are trained with the input $\boldsymbol{h}^s$ in the source task, the model can achieve the same performance as the source task only if the distributions of $\boldsymbol{h}^t$ and $\boldsymbol{h}^s$ is close enough. In our framework, $\mathcal{G}_t$ (whose parameters will be updated) plays the role of *Generator*, $\mathcal{G}_s$ (whose parameters are fixed) is like the *dataset*, and $\mathcal{D}$ performs as the *Discriminator*. On the one hand, the goal of $\mathcal{D}$ is to distinguish $\boldsymbol{h}^t$ generated by $\mathcal{G}_t$ from $\boldsymbol{h}^s$ generated by $\mathcal{G}_s$ as much as possible. On the other hand, $\mathcal{G}_t$ will train the $\boldsymbol{h}^t$ to match $\boldsymbol{h}^s$ by preventing $\mathcal{D}$ from classifying the label of two domains. Our proposed GAN-based transfer reinforcement learning corresponds to the following



(a) Training in source task

(b) Transferring feature representation in target task

Fig. 3. An overview of our proposed transfer reinforcement learning method. (a) We first trained a model in source task environment. Then we save the model's parameters when network environment has changed. (b) By doing the domain adaptation with GAN, we can align the target features to source features. Next, we put the generated target features into the well-trained Actor-Critic model to take an action on the new environment.

unconstrained optimization:

$$\min_{\mathcal{D}} \mathcal{L}_{\text{adv}\,\mathcal{D}}(\mathcal{S}^s, \mathcal{S}^t, \mathcal{G}_s, \mathcal{G}_t) = -\mathop{\mathbb{E}}_{\boldsymbol{s}^s\sim\mathcal{S}^s}[\log(\mathcal{D}(\mathcal{G}_s(\boldsymbol{s}^s)))] - \mathop{\mathbb{E}}_{\boldsymbol{s}^t\sim\mathcal{S}^t}\big[\log\big(1-\mathcal{D}(\mathcal{G}_t(\boldsymbol{s}^t))\big)\big], \tag{14}$$

$$\min_{\mathcal{G}_t} \mathcal{L}_{\text{adv}\,\mathcal{G}_t}(\mathcal{S}^s, \mathcal{S}^t, \mathcal{D}) = -\mathop{\mathbb{E}}_{\boldsymbol{s}^t\sim\mathcal{S}^t}\big[\log\big(\mathcal{D}(\mathcal{G}_t(\boldsymbol{s}^t))\big)\big], \tag{15}$$

where $\mathcal{S}^s$ and $\mathcal{S}^t$ mean the source state space and target state space respectively. Note that we use independent mappings for source and target data but only update the target feature encoder $\mathcal{G}_t$. Since the topology of network generally becomes wider than before (due to the access of new devices), the dimension of the target state vector $\boldsymbol{s}^t$ is usually greater than the source state vector $\boldsymbol{s}^s$. Therefore, the information in $\boldsymbol{s}^t$ will include the information of $\boldsymbol{s}^s$. In other words, the $\boldsymbol{s}^s$ can be extracted from the $\boldsymbol{s}^t$ in our framework.

The entire process of our work can be seen in Algorithm 1. $\mathcal{G}_t$ and $\mathcal{D}$ will be optimized using alternating gradient descent (AGD), which is usually used for updating GANs' parameters [53] and can be seen in Line 13-17 of Algorithm 1. In addition, the original GAN usually updates parameters with the sigmoid cross entropy loss function, which may lead to vanishing gradients and unstable training problem. Thus, we replace the loss function in Eq. (14) and Eq. (15) with the *Least Squares Loss Function* [54], which are displayed as:

$$\mathcal{L}_{\text{adv}\,\mathcal{D}} = \frac{1}{2}\mathop{\mathbb{E}}_{\boldsymbol{s}^s\sim\mathcal{S}^s}\Big[\big(\mathcal{D}(\mathcal{G}_s(\boldsymbol{s}^s))-1\big)^2\Big] + \frac{1}{2}\mathop{\mathbb{E}}_{\boldsymbol{s}^t\sim\mathcal{S}^t}\Big[\mathcal{D}\big(\mathcal{G}_t(\boldsymbol{s}^t)\big)^2\Big], \tag{16}$$

$$\mathcal{L}_{\text{adv}\,\mathcal{G}_t} = \frac{1}{2}\mathop{\mathbb{E}}_{\boldsymbol{s}^t\sim\mathcal{S}^t}\Big[\big(\mathcal{D}\big(\mathcal{G}_t(\boldsymbol{s}^t)\big)-1\big)^2\Big]. \tag{17}$$

Algorithm 1 GAN-Based Transfer Reinforcement Learning

Require: Well-trained source feature encoder $\mathcal{G}_s$, *Actor* and *Critic* in source domain.

1: Randomly initialize the parameters of target feature encoder $\mathcal{G}_t$ and domain classifier $\mathcal{D}$.
2: Replace the output layer of *Actor* with a new randomly initialized layer.
3: * Online Learning *\
4: **while** TRUE **do**
5: **for** $\tau = 0 \rightarrow \mathcal{T}$ **do**
6: * Decision Making *\
7: Receive the network state $\boldsymbol{s}_\tau^t$ in target domain and extract the correspond source state $\boldsymbol{s}_\tau^s$.
8: Calculate the source feature $\boldsymbol{h}_\tau^s = \mathcal{G}_s(\boldsymbol{s}_\tau^s)$ and target feature $\boldsymbol{h}_\tau^t = \mathcal{G}_t(\boldsymbol{s}_\tau^t)$.
9: Put the target feature $\boldsymbol{h}_\tau^t$ into the *Actor*, i.e., π_θ to get the action a_τ^t, then execute it on the network.
10: Calculate the reward $\mathcal{R}_\tau$ based on network statistics according to Eq. (8).
11: Obtain an evaluated value $V(\boldsymbol{h}_\tau^t)$ from the *Critic*.
12: * Train the GAN *\
13: **if** $\tau \bmod 2 = 0$ **then**
14: Update the parameters of $\mathcal{D}$ with Eq. (16)
15: **else**
16: Update the parameters of $\mathcal{G}_t$ with Eq. (17)
17: **end if**
18: **end for**
19: * Update the output layer of *Actor* *\
20: Compute the accumulated reward of each time slot:
$$\mathfrak{R}_\tau = \begin{cases} V(\boldsymbol{h}_\mathcal{T}^t), & \tau = \mathcal{T}, \\ \mathcal{R}_\tau + \gamma\mathfrak{R}_{\tau+1}, & \tau = 0, ..., \mathcal{T}-1. \end{cases}$$
21: Update the output layer of *Actor* based on the gradient $\nabla_\theta \mathcal{J} = \mathbb{E}_{\pi_\theta}\{\sum_{\tau=0}^{\mathcal{T}} \nabla_\theta \log \pi_\theta(a_\tau^t|\boldsymbol{h}_\tau^t)\cdot[\mathfrak{R}_\tau - V(\boldsymbol{h}_\tau^t)]\}$.
22: **end while**

In our algorithm, the parameters of *Actor* are fixed except its output layer, while the parameters of *Critic* are completely fixed. A new output layer of *Actor* will be randomly initialized

and it will be updated on the basis of samples observed from a new network environment. Since the ACKTR algorithm updates the *Actor* using the batched policy gradient (seen in Eq. (11)), we also utilize it to update the output layer of *Actor* in our algorithm, which can be seen in Line 21 of Algorithm 1. In summary, this mechanism enables changing the dimension of the action space if we want to modify the size of candidate path set, and the convergence speed of DRL model will also benefit from this procedure.

Since the common feature representations of source and target state spaces are trained with the adversarial domain adaptation method, the neural network behind feature encoder can be reused and transferred between different routing tasks. Due to this mechanism, the GAN-based transfer reinforcement learning algorithm can not only handle the variable topology and network status distribution, but also accelerate the training process of DRL-based routing.

D. Discussion of the Model Complexity

Firstly, we analyze the computational complexity of the GAN-based transfer reinforcement learning. We mainly discuss the computational complexity of the fully-connected layer, since our model is consists of numerous 2-layer fully-connected neural networks. In general, one fully-connected layer requires $n_i \cdot n_o$ multiply-accumulate operations (MACCs) during its forward propagation, where n_i and n_o are the input and output dimension of it [55]. In a network with $|E|$ transmission links, the input dimension of $\mathcal{G}_t$ is $n_I = (2|E| + 3)$ according to the definition of state space in Eq. (6). The output dimension of *Actor* is $n_O = N_P$ according to Eq. (7). Assuming the size of hidden and output layers of $\mathcal{G}_t$, and the dimension of input and hidden layers of $\mathcal{D}$, *Actor* and *Critic* are all n_h, the MACCs in our model are (noting that the output size of $\mathcal{D}$ and *Critic* are 1):

$$\begin{aligned} \text{MACC} &= n_I \cdot n_h + (n_h)^2 + (n_h)^2 + n_h \cdot n_O \\ &\quad + 2\Big[(n_h)^2 + n_h\Big] \\ &= n_h(2|E| + N_P + 4n_h + 5), \end{aligned} \tag{18}$$

Thus, the computational complexity of our model per time slot is $O(2|E| + N_P)$ since n_h is fixed in our experiments. This result indicates that the computational complexity of our model is proportional to the number of links and paths, which means that it will not rise sharply as the size of the network increases (Noting that we mainly utilize the k-shortest path algorithm to determine the candidate paths instead of using all paths in the network).

Secondly, we want to discuss the sample complexity of the GAN-based transfer reinforcement learning, since it is an essential trait when evaluating DRL algorithms. Sample complexity measures the amount of experience it takes to learn until behave well [20]. In [56], the overall sample complexity in solving numerous target tasks with the prior knowledge transferred from the source task is proved to be substantially smaller than solving them individually without transfer. In other words, the transfer learning method can be proved helpful in reducing sample complexity of reinforcement learning algorithm. In addition, there are two condition to ensure efficient domain adaptation [22]: 1) there is the need to utilize a certain amount of source data; and 2) there is the need to properly minimize the difference between the source and target domains. As stated in Section IV-C, our GAN-based transfer method utilize the experience collected from the source domain, in order to effectively reduce the divergence between source and target distributions (through generating common feature representations of source and target domains). Thus, the design of GAN-based transfer method reduces the sample complexity of a reinforcement learning model in a new environment, i.e., accelerates the model's convergence speed.

V. System Implementation

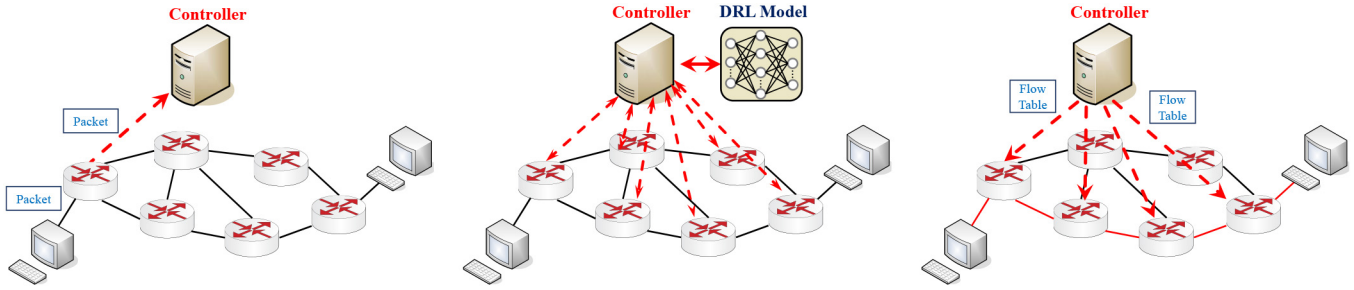
For realizing the network routing system with SDN architecture, we deploy the Open vSwitch (OVS) [57] on the network devices with Linux system, as the data plane. The controller is developed based on OpenDayLight [58], which consists of three components, including network monitor, path allocator, and transfer learning-enabled DRL model.

A. Network Monitor

The network monitor module collects network statistics, including remaining bandwidth and delay of all links. The link bandwidth is calculated by the statistics of every OVS port. This module periodically sends an OFPortStatsRequest message to every switch. Then the switch generates OFPortStatsReply message in response to the controller's message and uploads the total received bytes on its ports at the same time. Meanwhile, the controller has stored the capacity of each link. By subtracting the bytes received per second from the capacity, we can obtain the current remaining bandwidth of every link. The link delay is measured by triangulation method. The controller sends a probe packet with IP protocol number 253 to each switch through the Packet-Out message. Before sending the packet, this module tags a timestamp in IP option of the packet. After the packet passes through the first switch, it is forwarded to the neighbor switch and then transported back to the controller. When the controller received the packet through the Packet-In message, it compares the current system time with the timestamp to obtain a loop delay, which contains the link delay between two switches and the control link delay. This module simultaneously sends an Echo message to each switch, and uses this symmetric message to obtain the control link delay. We can calculate the delay of a link by subtracting the control link delay from the loop delay.

B. Path Allocator

When the system is initializing, the path allocator analyzes the topology and establishes the relationship between paths and the action space of DRL, preparing for path allocation. The process of path allocator is in Fig. 4. When a packet comes into a switch, it will be uploaded to the controller immediately. The controller extracts and stores the 5-tuple contained in the packet (i.e., source/destination IP, source/destination port numbers, and transport protocol). An array of 5-tuples is stored



(a) Switch requests a path. (b) Controller collects status and computes a path. (c) Controller installs the path.

Fig. 4. The process of our routing system.

in the controller, which will be emptied periodically. If the 5-tuple of the current packet is not contained, this module will put it into the array. Then the current network status (i.e., remaining bandwidth and delay of every link) will be sent into the DRL model to compute an action index. This module selects the corresponding path according to the index and installs flow tables to switches. After that, the selected path at this moment will be installed into the network. When the topology or network status distribution has changed, the path allocator sends the new topology to DRL model and informs it to start transfer reinforcement learning. This module will work as before in the transfer reinforcement learning stage.

C. Transfer Learning-Enabled DRL Model

The transfer learning-enabled DRL model has two working stages. One is the conventional deep reinforcement learning stage and the other is the transfer reinforcement learning stage.

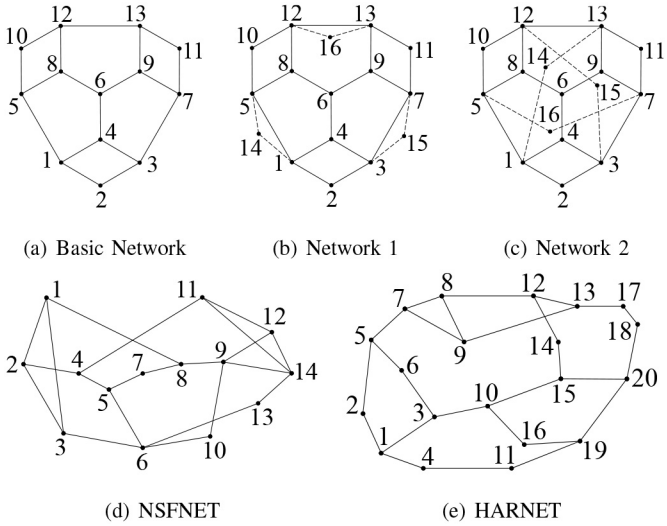
In the conventional deep reinforcement learning stage, the DRL model receives data from other modules. The statistics of all links are used as the input of neural networks to obtain the corresponding action. Then the action index is sent to the path allocator to install a transmission path. Finally, the reward is calculated according to the latency and throughput of the selected path, which will be the feedback of the DRL model.

When the path allocator informs the DRL model that the topology or network status distribution has changed, the model will stop updating and step into the transfer reinforcement learning stage. In this stage, the workflow is still similar to the conventional deep reinforcement learning stage, which contains that the model accepts a state as input, generates action index and receives reward as feedback. The processes of the two stages are depicted in Fig. 3.

VI. Performance Evaluation

A. Experiment Setting

In this section, we conduct extensive experiments to evaluate the proposed GAN-based transfer reinforcement learning algorithm in routing with variable environments. As we mentioned above, the Mininet [26] that contains numerous OVSs is used as the data plane, which runs on a Intel Quad-Core 2.40GHz CPU and 24GB RAM server. The SDN controller is deployed on a standalone machine which has the Intel Quad-Core 3.40GHz CPU with 8GB RAM.



(a) Basic Network (b) Network 1 (c) Network 2

(d) NSFNET (e) HARNET

Fig. 5. The topologies used in experiments.

To evaluate the performance of our method in real-world scenarios, we use three online network topologies from The Internet Topology Zoo [59] in our experiments. The *Basic Network* is modified from the *T-lex* in Tokyo, Japan, which contains 13 nodes and 18 links. We also select *NSFNET* (14 nodes and 21 links) in USA and *HARNET* (20 nodes and 26 links) in Hong Kong, China. These topologies are shown in Fig. 5. Each link in the network has 100Mbps bandwidth capacity and 5ms delay as default. The traffic packets used in our experiments are from an open real traffic dataset [60], and we utilize Tcpreplay [61] to replay packets in the dataset. In our experiments, we replay traffic flows between six different $\langle src, dst \rangle$ pairs, i.e., we have six flow sets in our experiments. Each flow set has 16 candidate paths, i.e., the size of action space is 16. In each training episode, 50 flows are replayed into the network, which are distinguished by the 5-tuple [5]. We set the parameters according to the method of grid search. The parameters of Eq. (8) are set as $\alpha = 0.05$, $\beta = 0.1$, when the units of bandwidth and delay are **Mbps** and **ms** respectively. Each neural network component is set to a 2-layer fully-connected structure with 64 neurons per layer (the activation function is *tanh*()). The learning rate is 0.01, and the discount coefficient is $\gamma = 0.95$.

In the evaluations, we design four experiments on transferring between different topologies and four experiments

TABLE III
EXPERIMENT SETUP

Network Topology Settings	Source Task	Target Task: New topology				Target Task: New network status distribution			
	Basic Network (BN)	Network 1	Network 2	NSFNET	HARNET	BN & Status 1	BN & Status 2	BN & Status 3	BN & Status 4
Nodes	13	16	16	14	20	13	13	13	13
Links	18	24	24	21	26	18	18	18	18
# of link bandwidth capacity	default[1]	default	default	default	default	default	default	4-6, 6-8, 6-9=10Mbps	4-6, 6-8, 6-9,1-5, 3-7, 12-13=10Mbps
# of link delay	default[2]	default	default	default	default	4-6, 6-8, 6-9=30ms	4-6, 6-8, 6-9,1-5, 3-7, 12-13=30ms	default	default

[1] Default means the bandwidth of all links is 100Mbps.
[2] Default means the delay of all links is 5ms.

on transferring with changed network status distribution. In detail, our source and target tasks are described as follows. The experiment settings are shown in Table III.

- *Source Task:* We train a model using ACKTR algorithm on the basic network, shown in Fig. 5(a).
- *Target Tasks (Basic Network With New Links):* In these target tasks, we add some links on the basic network as the new topologies, seen in Figs. 5(b) and 5(c).
- *Target Tasks (NSFNET and HARNET):* To evaluate the performance of our method in the scenario that topology has a great change, we use NSFNET and HARNET as the topologies of target tasks, which are displayed in Fig. 5(d) and Fig. 5(e).
- *Target Tasks (Basic Network With new Distributions of Network Status):* In these target tasks, the distribution of network status will be changed. To be specific, we simulated different network status distributions by changing the delay or bandwidth capacity of some links.

To evaluate our algorithm, we compared the performance with the following algorithms.

- *GAN-Based Transfer Method*: Train a model on the target task with the proposed GAN-based transfer reinforcement learning algorithm.
- *ACKTR:* Train a *new model* on the target task with random initialized weights, using ACKTR algorithm.
- *Naive Transfer Method:* Train a model on the target task with the pre-trained parameters on the source task (i.e., *fine-tune* the model with new experience from the new network environment). The weights and bias of feature encoder and output layer of Actor network are randomly initialized. The weights and bias of other layers are accepted from the source task.
- *Other Learning-Based Algorithms:* Test DRL-TE [7], DQR [14], and FE-DRL [15] on the target task.
- *Traditional Routing Algorithms:* Test OSPF and SMORE [34] on the target task.

B. Evaluation Metrics

In supervised learning, the prediction accuracy is used to assess the ability of models. Likewise, the cumulative reward per episode is crucial for evaluating DRL methods, which represents the adaptability of DRL agent in a new environment. Since the goal of transfer reinforcement learning is to shorten the training process, the performance improvement by the transfer method can be reflected in the contrast of two reward curves (one belongs to deep reinforcement learning algorithm and the other belongs to transfer reinforcement learning algorithm). For the sake of simplicity, the reward curves in each target task are normalized by $(r-r_{min})/(r_{\max}-r_{\min})$, where $r_{\max}$ and $r_{\min}$ are the maximum and minimum rewards of all methods in this target task.

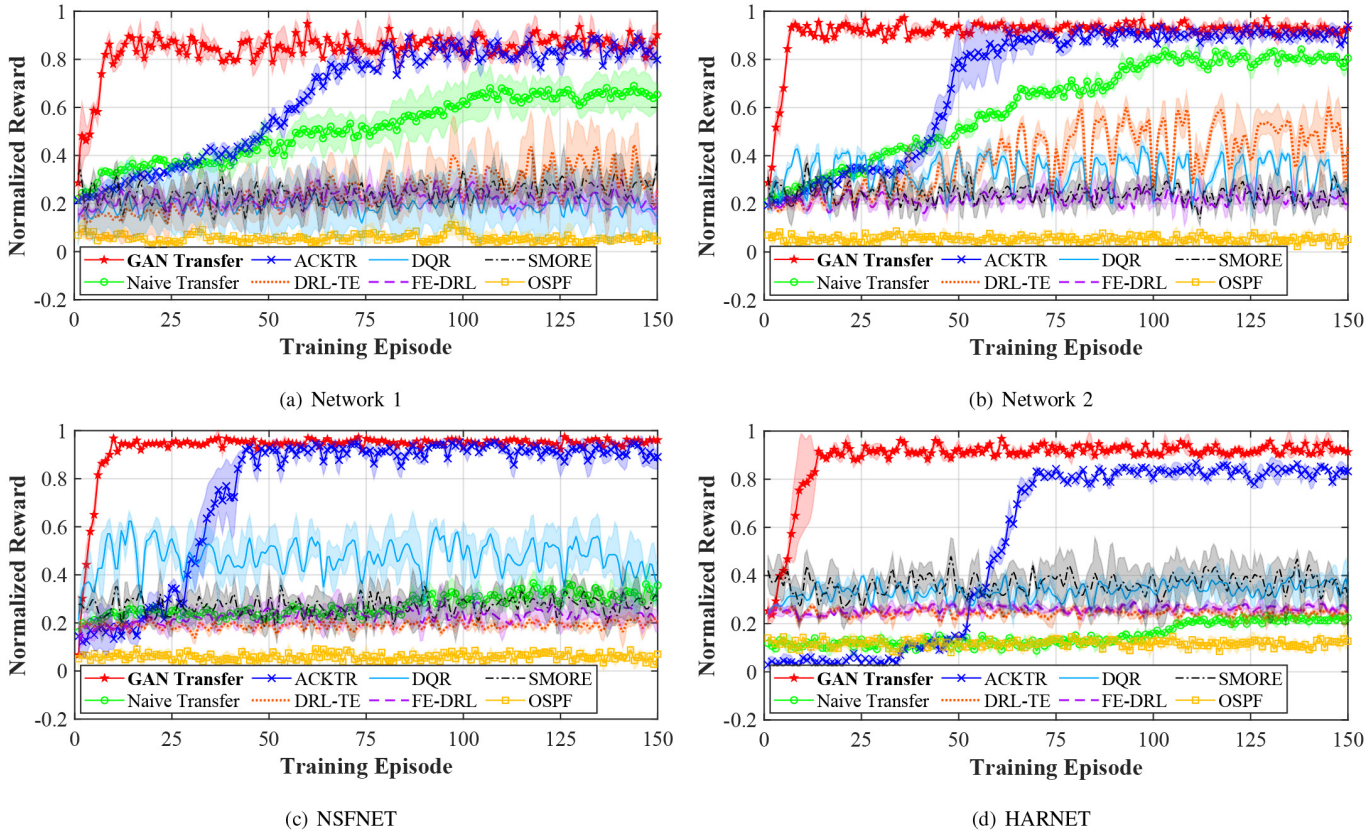
Moreover, the metrics to measure the transfer efficiency are crucial to the analysis of the experimental results. According to [23], we can use the following metrics to evaluate the transfer efficiency of transfer methods in the experiments.

- *Time to Threshold:* This metric is the convergence time of DRL algorithms. To be specific, the threshold time is determined by the time when the reward is higher than 90% of its maximum value. Time to threshold can show the difference in training time between a conventional DRL and a transfer reinforcement learning method, which means the reduction in the learning experience needed to solve the target task at hand.
- *Asymptotic Performance:* It is the difference in the final reward level between a conventional DRL and a transfer method, which compares the final performance of models in the target task both with and without transfer.
- *Transfer Ratio:* The ratio of total reward accumulated by a transfer method and a conventional DRL method, which is a obvious metric to quantify the enhancement of transfer method. Formally, it is defined as $r = (A_t - A_n)/A_n$, where A_t is the area under curve with transfer method and A_n is the area under curve without transfer.

At last, we will examine the average end-to-end latency and throughput of flows in the network. These show the superiority of transfer reinforcement learning in real-world scenarios.

C. Result and Analysis

From the Figs. 6 and 7, we obtain the reward curves of different algorithms in various settings. Although OSPF and SMORE are not the DRL-based routing method that needs a training procedure, we compute its reward using the throughput and latency, to provide more comprehensive comparison between the traditional and DRL-based routing



(a) Network 1

(b) Network 2

(c) NSFNET

(d) HARNET

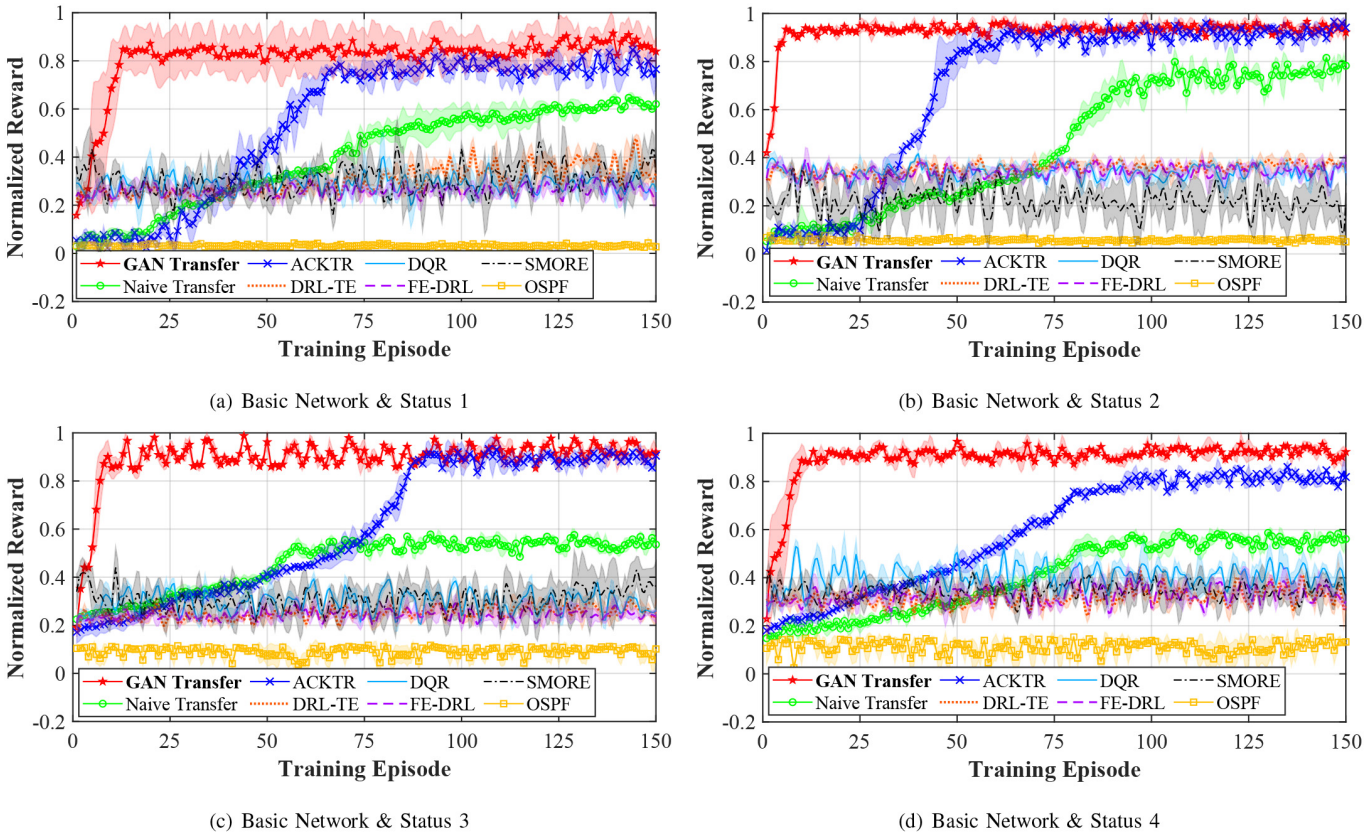
Fig. 6. The performance of varying topology experiments. The results are averaged over three runs and the shadows represent standard deviation.

methods. Compared to the state-of-the-art DRL algorithm ACKTR, GAN-based transfer reinforcement learning conspicuously improves the convergence speed. In detail, our method is 8.75X, 9.50X, 5.25X, 5.23X, 5.58X, 13.25X, 7.17X, and 10.33X respectively faster than ACKTR, in the routing on *Network1*, *Network2*, *NSFNET*, *HARNET*, *Status1*, *Status2*, *Status3*, and *Status4*. And it also has greater convergence speed advantage over other DRL-based routing frameworks DRL-TE, DQR, and FE-DRL, as well as the traditional routing method OSPF and SMORE. Overall, the GAN-based transfer reinforcement learning algorithm not only reaches the same reward level as the conventional DRL method in target tasks, but also reduces the convergence time of DRL algorithms, both in variable network topology and network status distribution scenarios. These results demonstrate that our method makes DRL-based routing more suitable for the highly dynamic emerging networks. The achievements of our algorithm mainly benefit from both the transfer learning mechanism and the GAN-based domain adaptation method. In terms of the end-to-end latency and throughput, Fig. 8 shows the GAN-based transfer reinforcement learning delivers satisfying results. In the experiments, GAN-based transfer method achieves better latency and throughput performance compared to all the other methods, which demonstrates our algorithm achieves its objective to obtain the optimal path with minimal end-to-end latency and maximal throughput.

We can see that the naive transfer method outperforms DRL-TE, DQR, and FE-DRL on many target tasks (seen in Figs. 6(a), 6(b), and 7(a)-7(d)), and the latter three DRL-based routing methods is obviously unable to reach the optimum. The idea of transfer learning-empowered DRL models the ability of adapting to network variations. Consequently, DRL models with transfer learning mechanism can reach the convergence in a new environment faster than without transfer.

It is noted that the naive transfer method does not work very well in NSFNET and HARNET topologies (seen in Figs. 6(c) and 6(d)), where it fails to reach the convergence on target tasks. Moreover, it performs worse than the state-of-the-art DRL algorithm ACKTR in all experiments. The comparison between the curves of two transfer methods obviously shows that our GAN-based transfer method outperforms the naive transfer method in terms of both the training speed and the total accumulated reward. Then we analyze the reason why the GAN-based transfer method is more effective than the naive transfer method. On the one hand, the naive transfer algorithm in our experiments is based on the fine-tune method, which directly runs on the well-trained model from the source task in target tasks. This method simply updates the model in a new environment and lacks a efficient way to establish the relationship between source and target domains. On the other hand, the GAN-based algorithm utilizes the adversarial domain adaptation to learn the common feature representation of source and target domains. This mechanism effectively promotes the adaptation of previous model trained in the source state space in a new target state space (i.e., a new environment), which greatly accelerates the training process of transfer reinforcement learning.

Furthermore, we compare the performance of two transfer methods on specific metrics, where ACKTR is utilized as the baseline algorithm. Fig. 9(a) clearly demonstrates that



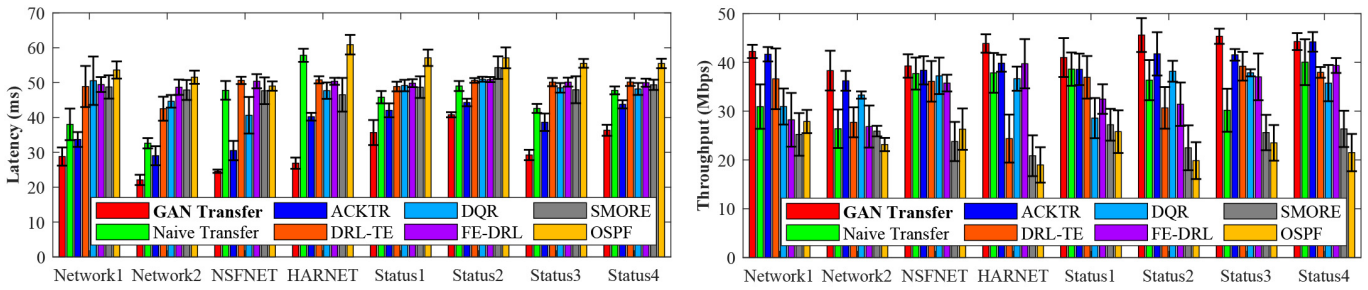
(a) Basic Network & Status 1

(b) Basic Network & Status 2

(c) Basic Network & Status 3

(d) Basic Network & Status 4

Fig. 7. The performance of varying network status distribution experiments. The results are averaged over three runs and the shadows represent standard deviation.



(a) Average End-to-End Latency of Flows

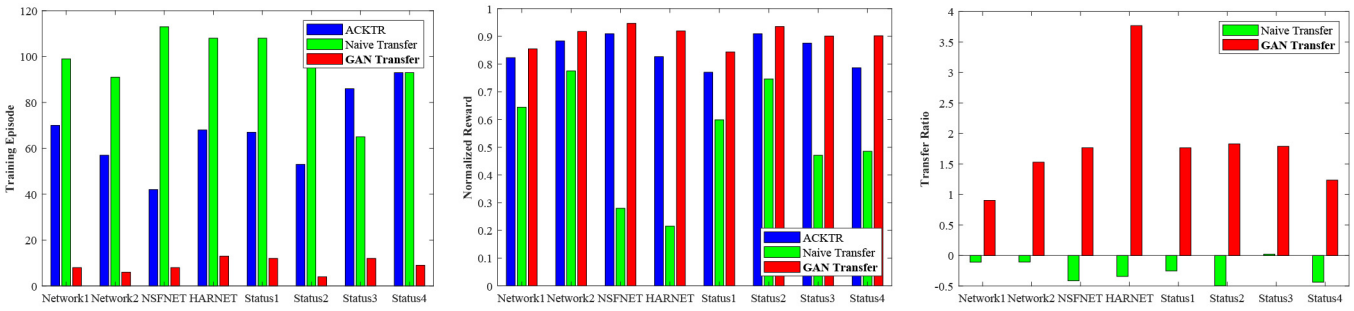
(b) Average End-to-End Throughput of Flows

Fig. 8. The average latency and throughput of different algorithms. The error bar shows the 95% confidence interval.

the GAN-based transfer method significantly outperforms the naive transfer method in terms of the time to threshold, and the latter even needs more episodes than the baseline in most of experiments. Moreover, the final performance of our method is much better than the naive transfer method, as shown in Fig. 9(b). The GAN-based method has a higher reward level than the baseline, while the naive method usually gains less reward in total. Finally, Fig. 9(c) presents the transfer ratio of the two methods over the baseline, which obviously reveals that the GAN-based method has a huge advantage over the naive method in transfer efficiency. In other words, our method accumulates more rewards overall in the same episodes. These results also indicate the effectiveness of the proposed GAN-based transfer reinforcement learning in the network routing.

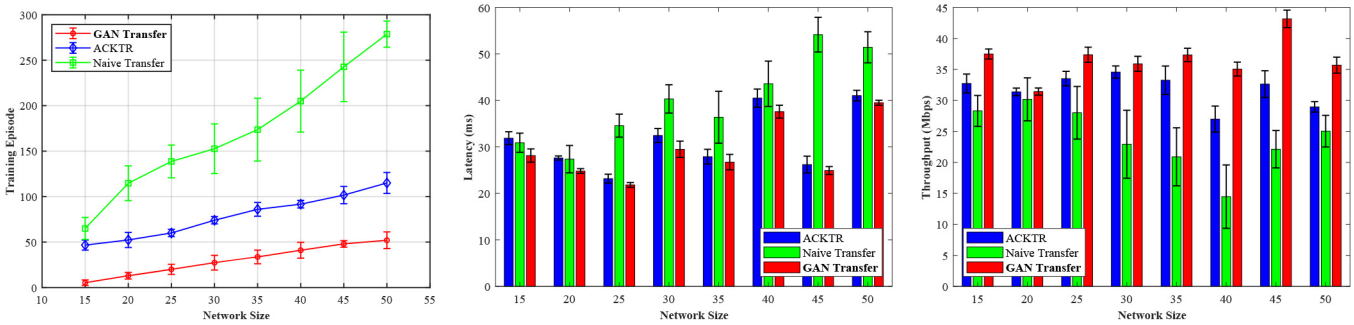
Finally, we evaluate the model performance under different network sizes since the scalability of the DRL model is critical to routing in the real-world scenarios. We utilize the Barabási-Albert model [62], [63] to generate topologies with different size. Fig. 10(a) shows that the convergence time of the GAN-based transfer method is consistently smaller than ACKTR and naive transfer method under different network sizes. From Figs. 10(b) and 10(c), the GAN-based method also performs better than ACKTR and naive method in terms of latency and throughput. These results proved our GAN-based transfer reinforcement learning has good scalability and achieves better performance (i.e., convergence time, latency and throughput) than other methods under different network sizes.

In addition, we want to discuss the limitations of our model. Firstly, we talk about how the topology setup affects the



(a) Time to Threshold (b) Asymptotic Performance (c) Transfer Ratio

Fig. 9. The comparison between ACKTR, naive transfer method, and GAN-based transfer method.



(a) Convergence Time (b) Average Latency (c) Average Throughput

Fig. 10. The models' convergence time, average latency and throughput for networks with different sizes. The error bar shows the 95% confidence interval.

performance of our model. As depicted in Fig. 10(a), the convergence time of ACKTR increases with the growth of network size. This phenomenon still applies to our GAN-based transfer learning model, but the gap of convergence time between a larger network size and a smaller network size is much smaller. However, with the increasing of network size, the training time is unacceptable even if our model can provide a shorter convergence time than other DRL models. In that case, the distributed SDN architecture [64] is used to divide a larger network into multiple small sub-domains, and each sub-domains can run a DRL model for route planning in it. Secondly, we want to discuss the training of GAN. The GAN model is known to be difficult to train, with little stability or convergence guarantees [65]. This may lead to unstable training procedure or mode collapse. Since our DRL model is lightweight enough (every component is a 2-layer neural network), manually tuning the hyper-parameters of GAN can achieve good training efficiency [66]. When facing a larger problem scale or more complex transfer learning setting, methods for stabilizing GAN training can be considered, such as Wasserstein GAN [67] (that greatly overcomes the training instability problem and partially solves the collapse mode problem). Nevertheless, how to completely avoid collapse mode and further optimize the training process remains a research direction of GANs.

VII. Conclusion

In this paper, we propose a GAN-based transfer reinforcement learning algorithm for SDN routing in variable network environments. The goal is to shorten the training time of DRL and utilize the well-trained model when the network environment has been changed. To be specific, the GAN is integrated into DRL algorithm, which can learn a common feature representation of two domains, in order to effectively adapt the previous model from the source domain to the target domain. Our transfer-enabled intelligent routing is implemented into SDN controller in the variable network environments. We conduct extensive experiments on Mininet platform to evaluate its performance. The evaluation results show that 1) GAN-based transfer method obviously accelerates the training progress in target tasks, compared to traditional and DRL-based routing frameworks; 2) GAN-based transfer method performs well on both scenarios with variable topology and network status distribution; and 3) in contrast with the naive transfer method, our algorithm largely outperforms it in terms of the total reward and the convergence time.

For future works, while this paper focuses on employing transfer learning to utilize the model from one source domain to one target domain, the GAN-based transfer reinforcement learning model can be improved for transferring knowledge between one source domain and multiple target domains. In addition, we also plan to integrate the graph neural networks for improving the transfer efficiency our model, or enhance the training stability with some advanced GAN architecture like Wasserstein GAN [67].

References

[1] A. Čolaković and M. Hadžialić, "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues," *Comput. Netw.*, vol. 144, pp. 17–39, Oct. 2018.

[2] L. Zhao, W. Zhao, A. Al-Dubai, and G. Min, "A novel adaptive routing and switching scheme for software-defined vehicular networks," in *Proc. IEEE ICC*, May 2019, pp. 1–6.

[3] M. H. Rehmani, A. Davy, B. Jennings, and C. Assi, "Software defined networks based smart grid communication: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2637–2670, 3rd Quart., 2019.

[4] J. Yang, K. Zhu, Y. Ran, W. Cai, and E. Yang, "Joint admission control and routing via approximate dynamic programming for streaming video over software-defined networking," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 619–631, Mar. 2017.

[5] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[6] J. Xie *et al.*, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, 1st Quart., 2019.

[7] Z. Xu *et al.*, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 1871–1879.

[8] Z. M. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 4th Quart., 2017.

[9] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 445–458, Jun. 2019.

[10] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[11] A. Mestres *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 2–10, Sep. 2017.

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

[13] X. Huang, T. Yuan, G. Qiao, and Y. Ren, "Deep reinforcement learning for multimedia traffic control in software defined networking," *IEEE Netw.*, vol. 32, no. 6, pp. 35–41, Nov./Dec. 2018.

[14] S. Q. Jalil, M. H. Rehmani, and S. Chalup, "DQR: Deep Q-routing in software defined networks," in *Proc. IEEE IJCNN*, Jul. 2020, pp. 1–8.

[15] J. Suarez-Varela *et al.*, "Feature engineering for deep reinforcement learning based routing," in *Proc. IEEE ICC*, May 2019, pp. 1–6.

[16] L. Chen, D. Yang, D. Zhang, C. Wang, J. Li, and T.-M.-T. Nguyen, "Deep mobile traffic forecast and complementary base station clustering for C-RAN optimization," *J. Netw. Comput. Appl.*, vol. 121, pp. 59–69, Nov. 2018.

[17] Z. Yang and K. L. Yeung, "SDN candidate selection in hybrid IP/SDN networks for single link failure protection," *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 312–321, Feb. 2020.

[18] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[19] N. Promwongsa *et al.*, "A comprehensive survey of the tactile Internet: State-of-the-art and research directions," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 472–523, 1st Quart., 2020.

[20] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "PAC model-free reinforcement learning," in *Proc. ICML*, Jun. 2006, pp. 881–888.

[21] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," Sep. 2017. [Online]. Available: arXiv:1708.05866v2.

[22] S. Ben-David and R. Urner, "On the hardness of domain adaptation and the utility of unlabeled target samples," in *Proc. ALT*, Oct. 2012, pp. 139–153.

[23] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Dec. 2009.

[24] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2009.

[25] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. IEEE CVPR*, Jul. 2017, pp. 2962–2971.

[26] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. ACM SIGCOMM Workshop HotNets*, Oct. 2010, pp. 1–19.

[27] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[28] Y. Li, R. Miao, C. Kim, and M. Yu, "FlowRadar: A better NetFlow for data centers," in *Proc. USENIX NSDI*, Mar. 2016, pp. 311–324.

[29] M. Goyal *et al.*, "Improving convergence speed and scalability in OSPF: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 443–463, 2nd Quart., 2011.

[30] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni, "Traffic engineering with MPLS in the Internet," *IEEE Netw.*, vol. 14, no. 2, pp. 28–33, Mar. 2000.

[31] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *Proc. IEEE GLOBECOM*, Dec. 2015, pp. 1–6.

[32] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM Conf.*, Aug. 2013, pp. 15–26.

[33] H. Xu, X.-Y. Li, L. Huang, H. Deng, H. Huang, and H. Wang, "Incremental deployment and throughput maximization routing for a hybrid SDN," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1861–1875, Jun. 2017.

[34] P. Kumar *et al.*, "Semi-oblivious traffic engineering: The road not taken," in *Proc. USENIX NSDI*, Apr. 2018, pp. 157–170.

[35] B. Mao *et al.*, "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.

[36] F. Tang *et al.*, "On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 154–160, Feb. 2018.

[37] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2260–2270, Oct. 2020.

[38] Z. Zhuang, J. Wang, Q. Qi, H. Sun, and J. Liao, "Toward greater intelligence in route planning: A graph-aware deep learning approach," *IEEE Syst. J.*, vol. 14, no. 2, pp. 1658–1669, Jun. 2020.

[39] H.-S. Fang, G. Lu, X. Fang, J. Xie, Y.-W. Tai, and C. Lu, "Weakly and semi supervised human body part parsing via pose-guided knowledge transfer," in *Proc. IEEE CVPR*, Jun. 2018, pp. 70–78.

[40] R. Sharma, P. Bhattacharyya, S. Dandapat, and H. S. Bhatt, "Identifying transferable information across domains for cross-domain sentiment classification," in *Proc. ACL*, Jul. 2018, pp. 968–978.

[41] G. Sun, L. Liang, T. Chen, F. Xiao, and F. Lang, "Network traffic classification based on transfer learning," *Comput. Elect. Eng.*, vol. 69, pp. 920–927, Jul. 2018.

[42] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," in *Proc. USENIX Security Symp.*, Aug. 2016, pp. 807–822.

[43] K. Mo, Y. Zhang, S. Li, J. Li, and Q. Yang, "Personalizing a dialogue system with transfer reinforcement learning," in *Proc. AAAI Conf.*, Feb. 2018, pp. 5317–5324.

[44] R. Li, Z. Zhao, X. Chen, J. Palicot, and H. Zhang, "TACT: A transfer actor–critic learning framework for energy saving in cellular radio access networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 4, pp. 2000–2011, Apr. 2014.

[45] I.-S. Comşa *et al.*, "Towards 5G: A reinforcement learning-based scheduling solution for data traffic management," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1661–1675, Dec. 2018.

[46] I.-S. Comşa, R. Trestian, G.-M. Muntean, and G. Ghinea, "5MART: A 5G sMART scheduling framework for optimizing QoS through reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 1110–1124, Jan. 2020.

[47] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin, and Y.-C. Liang, "Network slice reconfiguration by exploiting deep reinforcement learning with large action space," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2197–2211, Dec. 2020.

[48] V. R. Konda and J. N. Tsitsiklis, "Onactor–critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, 2003.

[49] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. ICLR*, May 2016, pp. 1–6.

[50] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation," in *Proc. NeurIPS*, Dec. 2017, pp. 5279–5288.

[51] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, Jun. 2016, pp. 1928–1937.

[52] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. ICML*, Jul. 2015, pp. 1889–1897.

[53] B. Zhu, J. Jiao, and D. Tse, "Deconstructing generative adversarial networks," *IEEE Trans. Inf. Theory*, vol. 66, no. 11, pp. 7155–7179, Nov. 2020.

[54] X. Mao *et al.*, "Least squares generative adversarial networks," in *Proc. IEEE ICCV*, Oct. 2017, pp. 2813–2821.

[55] M. Hollemans. *How Fast Is My Model?* Accessed: Jun. 2018. [Online]. Available: http://machinethink.net/blog/how-fast-is-my-model/

[56] E. Brunskill and L. Li, "Sample complexity of multi-task reinforcement learning," in *Proc. Conf. Uncertainty Artif. Intell.*, Aug. 2013, pp. 1–10.
[57] B. Pfaff *et al.*, "The design and implementation of Open VSwitch," in *Proc. USENIX NSDI*, May 2015, pp. 117–130.
[58] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDayLight: Towards a model-driven SDN controller architecture," in *Proc. IEEE WoWMoM*, Jun. 2014, pp. 1–6.
[59] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.
[60] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proc. ICISSP*, Feb. 2017, pp. 253–262.
[61] A. Turner. *Tcpreplay*. Accessed: Mar. 2019. [Online]. Available: https://tcpreplay.appneta.com/
[62] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.
[63] S.-H. Yook, H. Jeong, and A.-L. Barabási, "Modeling the Internet's large-scale topology," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 21, pp. 13382–13386, 2002.
[64] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 333–354, 1st Quart., 2018.
[65] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: Introduction and outlook," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 4, pp. 588–598, Sep. 2017.
[66] M. Lucic, K. Kurach, M. Michalski, O. Bousquet, and S. Gelly, "Are GANs created equal? A large-scale study," in *Proc. NeurIPS*, Dec. 2018, pp. 698–707.
[67] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. ICML*, Aug. 2017, pp. 214–223.

Tianjian Dong is currently pursuing the Ph.D. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His research interests include network routing, traffic engineering, resource management, deep learning, reinforcement learning, and graph neural networks.

Qi Qi (Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010, where she is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology. She has authored or coauthored more than 30 papers in international journal. Her research interests include edge computing, cloud computing, Internet of Things, ubiquitous services, deep learning, and deep reinforcement learning. She is the recipient of two National Natural Science Foundations of China.

Jingyu Wang (Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2008, where he is currently a Professor with the State Key Laboratory of Networking and Switching Technology. He has published more than 50 papers in international journal and famous conferences, including the *IEEE Communication Magazine*, IEEE SYSTEMS JOURNAL, CVPR, AAAI, and EMNLP. His research interests include broad aspects of SDN, multimedia services, multipath transmission, overlay networks, and traffic engineering.

Alex X. Liu (Fellow, IEEE) received the Ph.D. degree in computer science from The University of Texas at Austin in 2006. He is an Honorary Professor with the Qilu University of Technology. He was a Professor with the Department of Computer Science and Engineering, Michigan State University. His research interests focus on networking, security, and privacy. He received the IEEE and IFIP William C. Carter Award in 2004, the National Science Foundation CAREER Award in 2009, the Michigan State University Withrow Distinguished Scholar (Junior) Award in 2011, and the Michigan State University Withrow Distinguished Scholar (Senior) Award in 2019. He received Best Paper Awards from SECON-2018, ICNP-2012, SRDS-2012, and LISA-2010. He has served as an Editor for IEEE/ACM TRANSACTIONS ON NETWORKING and an Area Editor for *Computer Communications*. He is currently an Associate Editor for IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING and IEEE TRANSACTIONS ON MOBILE COMPUTING. He has served as the TPC Co-Chair for ICNP 2014 and IFIP Networking 2019. He is an ACM Distinguished Scientist.

Haifeng Sun (Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2017, where he is currently a Lecturer with the State Key Laboratory of Networking and Switching Technology. His research interests include broad aspects of AI, NLP, big data analysis, object detection, deep learning, and deep reinforcement learning.

Zirui Zhuang (Member, IEEE) received the B.S. degree in electronic information engineering and the Ph.D. degree in information and communication engineering from the Beijing University of Posts and Telecommunications in 2015 and 2020, respectively.

He is currently a Postdoctoral Researcher with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. In 2019, he visited the Department of Electrical and Computer Engineering, University of Houston, doing research on network routing and optimization. His research interests involve network routing and management for next-generation network infrastructures, using machine learning and artificial intelligence techniques, including deep learning, reinforcement learning, graph representation, multiagent system, and Lyapunov-based optimization.

Jianxin Liao (Member, IEEE) received the Ph.D. degree from the University of Electronics Science and Technology of China, Chengdu, China, in 1996. He is currently the Dean of the Network Intelligence Research Center and a Full Professor with the State Key laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He has authored or coauthored hundreds of research papers and several books. He has won a number of prizes in China for his research achievements, which include the Premiers Award of Distinguished Young Scientists from the National Natural Science Foundation of China in 2005, and the specially invited Professor of the "Yangtse River Scholar Award Program" by the Ministry of Education in 2009. His main research interests include cloud computing, mobile intelligent network, service network intelligent, networking architectures and protocols, and multimedia communication.