Deep Reinforcement Learning for Next Generation Wireless Networks with Echo State Networks

Hao-Hsuan Chang

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

August 3, 2021

Blacksburg, Virginia

Keywords: Deep Reinforcement Learning, Echo State Network, Dynamic Spectrum Sharing

Deep Reinforcement Learning for Next Generation Wireless Networks with Echo State Networks

Hao-Hsuan Chang

(ABSTRACT)

This dissertation considers a deep reinforcement learning (DRL) setting under the practical challenges of real-world wireless communication systems. The non-stationary and partially observable wireless environments make the learning and the convergence of the DRL agent challenging. One way to facilitate learning in partially observable environments is to combine recurrent neural network (RNN) and DRL to capture temporal information inherent in the system, which is referred to as deep recurrent Q-network (DRQN). However, training DRQN is known to be challenging requiring a large amount of training data to achieve convergence. In many targeted wireless applications in the 5G and future 6G wireless networks, the available training data is very limited. Therefore, it is important to develop DRL strategies that are capable of capturing the temporal correlation of the dynamic environment that only requires limited training overhead. In this dissertation, we design efficient DRL frameworks by utilizing echo state network (ESN), which is a special type of RNNs where only the output weights are trained. To be specific, we first introduce the deep echo state Q-network (DEQN) by adopting ESN as the kernel of deep Q-networks. Next, we introduce federated ESN-based policy gradient (Fed-EPG) approach that enables multiple agents collaboratively learn a shared policy to achieve the system goal. We designed computationally efficient training algorithms by utilizing the special structure of ESNs, which have the advantage of learning a good policy in a short time with few training data. Theoretical analyses are conducted for DEQN and Fed-EPG approaches to show the convergence properties and to provide a guide to hyperparameter tuning. Furthermore, we evaluate the performance under the dynamic spectrum sharing (DSS) scenario, which is a key enabling technology that aims to utilize the precious spectrum resources more efficiently. Compared to a conventional spectrum management policy that usually grants a fixed spectrum band to a single system for exclusive access, DSS allows the secondary system to dynamically share the spectrum with the primary system. Our work sheds light on the real deployments of DRL techniques in next generation wireless systems.

Deep Reinforcement Learning for Next Generation Wireless Networks with Echo State Networks

Hao-Hsuan Chang

(GENERAL AUDIENCE ABSTRACT)

Model-free reinforcement learning (RL) algorithms such as Q-learning are widely used because it can learn the policy directly through interactions with the environment without estimating a model of the environment, which is useful when the underlying system model is complex. Q-learning performs poorly for large-scale models because the training has to updates every element in a large Q-table, which makes training difficult or even impossible. Therefore, deep reinforcement learning (DRL) exploits the powerful deep neural network to approximate the Q-table. Furthermore, a deep recurrent Q-network (DRQN) is introduced to facilitate learning in partially observable environments. However, DRQN training requires a large amount of training data and a long training time to achieve convergence, which is impractical in wireless systems with non-stationary environments and limited training data. Therefore, in this dissertation, we introduce two efficient DRL approaches: deep echo state Q-network (DEQN) and federated ESN-based policy gradient (Fed-EPG) approaches. Theoretical analyses of DEQN and Fed-EPG are conducted to provide the convergence properties and the guideline for designing hyperparameters. We evaluate and demonstrate the performance benefits of the DEQN and Fed-EPG under the dynamic spectrum sharing (DSS) scenario, which is a critical technology to efficiently utilize the precious spectrum resources in 5G and future 6G wireless networks.

Dedication

To my parents, my sister, and my dear wife

Acknowledgments

First and foremost, I want to express my deepest gratitude to my Ph.D. advisor, Dr. Lingjia Liu, for his continuous support and guidance of my Ph.D. study. His tremendous patience, knowledge, and experience made my Ph.D. journey extremely rewarding and productive. I am grateful to him for believing in me and dedicating his time and efforts to help me grow not only as an independent researcher but also as a good person. I would like to thank the members of my Ph.D. advisory committee, Dr. Jeffrey H. Reed, Dr. R. Michael Buehrer, Dr. Yang Yi, and Dr. Zhenyu Kong for their valuable comments and suggestions, which have helped me to substantially improve the quality of this dissertation. I would also like to thank all my colleagues in Wireless@VT for their help and support. I have benefited a lot from their diverse expertise through the discussion and collaboration with them. I am grateful to all my friends in Blacksburg for encouraging me when I am depressed, which helped me get through many difficulties.

Last, but not the least, I want to express my heartiest gratitude to my parents, my sister, and my dear wife. Words are powerless to express my gratitude to my family. Special thanks to my dear wife, Yuchia Hsueh, who has always been supporting me, and believing in me through up and downs. She has always been the spiritual prop and the source of my strength. Without my family's unconditional love and persistent support, I would never be able to finish this Ph.D. dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Reinforcement learning (RL)

RL is an important type of machine learning where an RL agent learns how to behave in an environment. It lies between supervised and unsupervised learning. The supervised learning algorithm learns a function that maps an input to an output based on a labeled dataset, which provides answers of example input-output pairs. In contrast, the unsupervised learning algorithms learns how to infer a function to describe a hidden structure from unlabeled data. Unlike the ground truth lables provided by the supervised learning, the RL agent is not told which action to take. Instead, the RL agent must explore the environment to accumulate knowledge of the environment and exploit its accumulated knowledge to take the best action. Specifically, the RL algorithm aims to learn a policy, a mapping from the observed states to the action, to maximize the cumulative reward. RL provides a flexible architecture for solving many types of practical problems because it does not need to model complex systems or to label data for training.

The dynamics of the stochastic environment in RL is usually modeled as a Markov decision process (MDP), which characterized by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the state transition probability, R is the reward function, and γ is a discount factor for calculating cumulative reward. Specifically, at time t, the state is $s_t \in \mathcal{S}$, the RL agent selects an action $a_t \in \mathcal{A}$ by following a policy $\pi(s_t)$ and receives the reward $r_t = R(s_t, a_t)$, and then the system shifts to the next state s_{t+1} according to the state transition probability $\mathcal{T}(s_{t+1}|s_t, a_t)$. Note that the action a_t affects both the immediate reward r_t and the next state s_{t+1}. Consequently, all subsequent rewards are affected by the current action. The goal of the RL agent is to find a policy π to maximize the expected cumulative reward, $\mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t\right]$.

1.2 Model-free RL

In RL, a model-free algorithm directly learns the policy through interactions with the environment without explicitly estimating the transitions between states and the reward function associated with the MDP. Model-free RL algorithms are helpful if the underlying dynamics of the environment is too complex to be estimated. Q-learning [1] is the most widely used model-free RL algorithm that iteratively improve the decisions of the learning agent through estimating the Q-function. Each element of the Q-function represents the Q-value of a state-action pair for a given policy, which is defined as

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_1 = s, a_1 = a \right]. \tag{1.1}$$

Accordingly, $Q^\pi(s, a)$ represents the expected cumulative reward when taking action a in the initial state s and then following the policy π. The optimal policy has the maximum expected cumulative reward for all state-action pairs, which is defined as

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q^{\pi^*}(s, a). \tag{1.2}$$

To find the optimal policy, Q-learning constructs a Q-table to estimate the Q-function of the optimal policy by iteratively updating each Q-value through a value iteration approach.

To be specific, the update rule of the value iteration is written as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right], \tag{1.3}$$

where $\alpha \in (0, 1)$ is the learning rate. Once the Q-table is updated, the policy π is also updated using the ϵ-greedy policy as follows:

$$a_t = \begin{cases} \operatorname{argmax}_a Q(s_t, a) & \text{, with probability } 1 - \epsilon, \\ \text{random action} & \text{, with probability } \epsilon, \end{cases} \tag{1.4}$$

where $\epsilon \in [0, 1]$ is the exploration probability. The Q-function and the policy updates alternatively until convergence. It is well-known that Q-learning can converge to the optimal policy and the corresponding Q-function [1].

Although Q-learning is a simple and effective method, it performs poorly when the dimension of the state is high because updating a large Q-table makes training difficult or even impossible. Deep Q-Networks (DQN) [2] is introduced to solve high-dimensional state problems by leveraging a neural network as the function approximator of the Q-table, which is referred to as the Q-network. DQN has attracted much attention in recent years because it enables RL to efficiently learn in a large state and action spaces by providing a good approximation of Q-value using the powerful deep neural networks. Specifically, the Q-network takes the state as input and outputs the estimated Q-function of all possible actions. To improve the stability of the Q-newtwork training, DQN creates two Q-networks: the evaluation network $Q(s, a; \theta)$ and the target network $Q(s, a; \hat{\theta})$. The target network is utilized to generate the targets for training the evaluation network, while the evaluation network is utilized to determine the actions. The loss function for training the weights of the evaluation network θ

is written as

$$\left(r_t + \gamma \max_a Q(s_{t+1}, a; \hat{\theta}) - Q(s_t, a_t; \theta)\right)^2, \tag{1.5}$$

where $r_t + \gamma \max_a Q(s_{t+1}, a; \hat{\theta})$ is the target Q-value. The weights of the target network $\hat{\theta}$ is periodically synchronized with the weights of the evaluation network θ. In this way, the target Q-values can be fixed temporarily during the training of the evaluation network instead of changing in each training iteration, so the training stability is significantly improved.

1.3 Challenges of Real-World RL

In the last few years, RL has been shown to be effective on different fields, such as playing video games [2], playing chess [3], to robotics [4]. However, much of the progress in RL is difficult to be exploited in real-world systems due to some practical challenges. In this section, we will present some of these practical challenges that appear when applying RL to real-world systems.

1.3.1 Partial Observability

In many real-world sequential decision processes, the state is not fully observable by the agent, and thus a MDP is generalized to a partially observable Markov decision process (POMDP). POMDP model is characterized by a tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, R, Ω, $\mathcal{O}$, γ), where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the state transition probability, R is the reward function, Ω is the observation space, $\mathcal{O}$ is the conditional observation probability, and $\gamma \in [0, 1]$ is a discount factor for calculating the cumulative reward. To be specific, at time t, the state is $s_t \in \mathcal{S}$, the agent receives an observation $o_t \in \Omega$ with probability $\mathcal{O}(o_t|s_t)$ and selects an action $a_t \in \mathcal{A}$. Afterwards, the agent receives a reward $r_t = R(s_t, a_t)$, and the system shifts to the next state s_{t+1} based on the state transition probability $\mathcal{T}(s_{t+1}|s_t, a_t)$. In partially observable environments, an observation at single time step may not contain sufficient information to predict future rewards and future states. Therefore, a policy that depends on observation histories has a beneficial effect in POMDP environments.

1.3.2 Non-stationary Environment

In many real-world scenarios, the non-stationary environments degrade the performance of DRL-based strategies drastically over time. The non-stationary environment poses a big challenge to learning stability and prevents the direct use of experience replay because the training data become obsolete as the environment changes [5]. On the other hand, experience replay is an important technique for DQN [2] to stabilize the training of neural networks and improve sample efficiency. Therefore, we design an online algorithm that is able to remove outdated training data and continue updating the policy in order to adapt to the non-stationary environment.

1.3.3 Limited Training Data

Most existing DRL algorithms require a large number of samples for the training to converge. This may not be an issue for applications such as computer games where samples can be easily obtained. However, in many other real-world problems, data collection is usually expensive making it challenging to directly apply DRL algorithms. Furthermore, the environments are usually non-stationary in real-world problems where the collected past data may become obsolete and no longer reflect the current environment as time goes. As a result, developing efficient DRL algorithms requiring limited training data with fast convergence will be critical to make DRL accessible to a wide range of real-world applications.

1.4 Echo State Network (ESN)

Recurrent neural network (RNN) is a powerful neural network structure that can learn the temporal behavior for a time sequence. Specifically, RNN has memory that stores the previous neuron activation to process sequential input data. To deal with the partial observability in many real-world environments, we can utilize RNNs in the DRL to capture the temporal correlation of observation sequences, which is referred to as the deep recurrent Q-network (DRQN) [6]. DRQN can better approximate actual Q-values from sequences of observations, leading to better policies in partial observable environments. Even though DRQN is a powerful machine learning tool, it faces serious issues related to training due to two reasons: 1) The kernel of DRQN, the RNN, has issues on vanishing and exploding gradients that make the underlying training computationally inefficient [7]; and 2) DRL requires a relatively large amount of training data to ensure the learning agent converges to an appropriate policy. In many real-world problems, the environments are non-stationary, where the rewards and the transition probabilities between states change with time. The training has to be computationally efficient to allow a DRL agent to quickly adapt its policy to the changing environment. Meanwhile, generating an enormous amount of training data is only feasible in artificial problems such as playing computer games. Obtaining training data from the environment is usually costly in real-world problems. Therefore, the difficulties in training DRQN prevent it from being widely adopted in real-world applications [8].

In light of training challenges of DRQNs, we utilize a special type of RNNs, echo state networks (ESNs), to reduce the training time and the required training data [9]. The network architecture of ESN is shown in Figure 1.1. Given a sequence of inputs $(x_1, \cdots, x_t)$, the

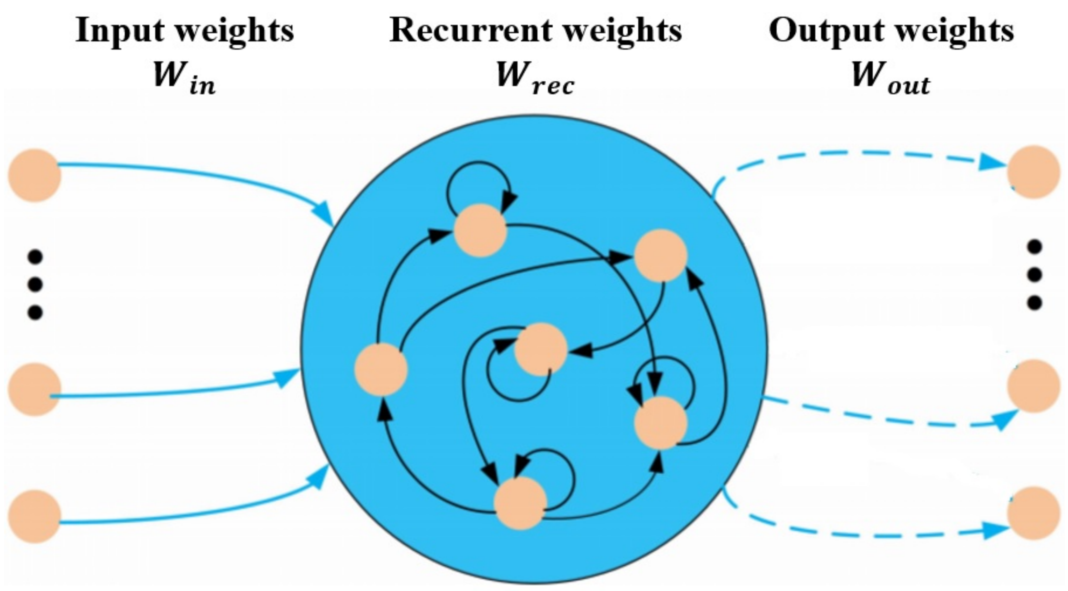


Figure 1.1: The network architecture of ESN.

update equations of RNN/ESN have the same form, which are written as:

$$\begin{aligned} h_t &= \tanh\left(W_{\text{in}} x_t + W_{\text{rec}} h_{t-1}\right), \\ y_t &= W_{\text{out}} h_t, \end{aligned} \tag{1.6}$$

where h_t is the hidden state, W_{in} is the input weights, W_{rec} is the recurrent weights, and W_{out} is the output weights. The hidden state h_t represents a summary of the past sequence of inputs up to t, and we set the initial hidden state $h_0 = \mathbf{0}$. The standard RNN training, backpropagation through time (BPTT), unfolds the network in time into a computational graph that has a repetitive structure to train input weights, recurrent weights, and output weights. On the other hand, ESNs simplify the underlying RNNs training by only training the output weights while leaving input weights and recurrent weights untrained. To be specific, the input weights and the recurrent weights of ESNs are initialized randomly according to the constraints specified by the Echo State Property [10], and then they remain untrained. Only the output weights of ESNs are trained so the training is extremely fast. The main idea of ESNs is to generate a large reservoir that contains the necessary summary of past input sequences for predicting targets. The fixed recurrent connections of the reservoir provide a high dimensional dynamics that is able to create all possible spatial and temporal combinations of the input history, which are analogous to cortical dynamics in human brain [11]. The learned output weights determine the best linear combination of the reservoir's state and the input signal to perform the desired task. This approach largely reduces the computation time because only the output weights are trained. Existing research shows that ESNs can achieve comparable performance with RNNs, especially in some applications requiring fast learning [12].

1.5 Summary of Contributions

Motivated by the success of DRL techniques in multiple domains, we utilize the power of the deep neural network to help the development of next generation wireless networks. In this dissertation, we first identify the practical challenges of applying DRL techniques to real-world wireless networks, where the wireless environments are usually non-stationary and partially observable. Meanwhile, the available training data that can be obtained from the wireless environment is extremely limited. To address the aforementioned practical challenges that appear when applying DRL to real-world wireless communication systems, we focus on designing an efficient DRL framework under non-stationary and partially observable environments with limited training data. To be specific, we introduced an efficient DRL method by adopting ESN as the Q-network in the DRL framework, which is referred to as the deep echo state Q-network (DEQN). DEQN can efficiently capture the temporal correlation of the underlying time-dynamic wireless communication systems.

We evaluate the performance of the introduced DEQN method under the dynamic spectrum sharing (DSS) scenario, which is a key enabling technology in 5G and future 6G wireless networks. First, we developed a single-user spectrum access strategy in the DSS networks using the DEQN approach. Next, we investigate multi-user spectrum access strategies in the DSS networks using two learning approaches: independent DEQN learning approach [13, 14] and federated ESN-based policy gradient (Fed-EPG) approach. To be specific, in the independent DEQN learning approach, each secondary user (SU) trains its DEQN agent locally to update its spectrum access policy under limited control information between the primary system and the secondary system. On the other hand, the Fed-EPG approach jointly learns a shared spectrum access policy for all SUs that can perform well across all SUs' local environments.

The main contributions of this dissertation can be summarized as the following:

- An efficient DRL method called DEQN is introduced to quickly adapt to the non-stationary and partially observable environment with limited training data. Meanwhile, we designed a specialized online training algorithm for DEQNs that decrease the training overheads significantly.
- Convergence analysis of the introduced DEQN approach is conducted to demonstrate the faster convergence of DEQN compared to that of DRQN.
- The bias-variance tradeoff of DEQN is characterized to identify the spectrum norm constraint of generating ESN's recurrent weights.
- The introduced Fed-EPG approach enables multiple users to collaboratively learn a shared policy in the multi-agent reinforcement learning (MARL) environment without sharing their private data. ESN-based policy network is suitable for the federated learning framework because the communication overheads can be largely decreased by only sharing the output weights of the ESN-based policy network.

- Convergence analysis of the Fed-EPG approach is conducted to show the tradeoff between the convergence rate and the communication period.
- We evaluate the performance of DEQN and Fed-EPG in the critical problem of DSS, which is a promising technology to improve the spectrum utilization for 5G and future 6G DSS networks.
- We built a new network simulator to evaluate the performance of DSS in a near real-world wireless environment. The simulator strictly follows the 3rd Generation Partnership Project (3GPP) standards based on field measurements.

1.6 Organization of the Dissertation

The rest of the dissertation is organized as follows. Chapter 2 presents the DEQN framework for the single-user spectrum access strategy in the DSS network. Section 2.1 describes the background knowledge of the DSS system and its challenges. Section 2.2 reviews the existing literature on conventional DSS strategies and DRL-based DSS strategies. Section 2.3 describes the system model for the considered DSS problem with single-SU and multi-PU, and Section 2.4 describes the DRL problem formulation of the corresponding system model. Section 2.5 presents the network architecture and the training algorithm of the introduced DEQN method. Section 2.6 provides the theoretical analysis of DEQN. Simulation results for the single-user DSS strategy are presented in Section 2.7. Chapter 2 is concluded in Section 2.8.

Chapter 3 presents the independent DEQN framework for the multi-user spectrum access strategy in the DSS network. Section 3.1 describes the spectrum multiple access methods in other wireless technologies and lists the requirements for designing spectrum multiple access in the DSS network. Section 3.2 describes the system model for the considered DSS problem with multi-SU and multi-PU, and Section 3.3 describes the corresponding DRL problem formulation. Section 3.4 presents the training algorithm of the independent DEQN framework. Simulation results for the multi-user DSS strategy using independent DEQN are presented in Section 3.5. Chapter 3 is concluded in Section 3.6.

Chapter 4 presents the collaborative learning framework called Fed-EPG for the multi-user spectrum access strategy in the DSS network. Section 4.1 introduces the background knowledge of multi-agent reinforcement learning, and Section 4.2 introduces the background knowledge federated learning. Section 4.3 describes the system model for the considered DSS problem in the CBRS system with multi-SU and multi-PU, and Section 4.4 describes the corresponding DRL problem formulation. Section 4.5 presents the network architecture and the training algorithm of the introduced Fed-EPG method. Section 4.6 provides the convergence analysis of Fed-EPG. Simulation results for the multi-user DSS strategy using Fed-EPG are presented in Section 4.7. Chapter 4 is concluded in Section 4.8.

Finally, we summarize the dissertation in Chapter 5.

Chapter 2

Dynamic Spectrum Sharing via ESN-based RL

2.1 Dynamic Spectrum Sharing (DSS)

In wireless communication, spectrum is a precious resource due to its scarcity. Only few MHz frequency band is extremely expensive. For example, in the FCC's AWS-3 spectrum auction, the revenues of total winning bids were $44.9 billion for 65 MHz bandwidth frequency [15]. With the development of new wireless technologies and applications, the demand for wireless access has increased remarkably in recent years. Meanwhile, the popularity of smartphones and widespread applications of the Internet of Things (IoT) has spurred explosive growth of the number of mobile devices. According to Cisco's annual internet report, the number of wireless devices is expected to grow at a compound annual growth rate (CAGR) of 10% between 2018 and 2023, reaching 29.3 billion wireless devices by 2023 [16]. To cope with this unprecedented high demand of wireless connections, extending the radio spectrum for commercial use is critical for the fifth-generation (5G) mobile broadband networks. However, the online table of frequency allocation published by the Federal Communications Commission (FCC) demonstrates the extremely congested frequency allocations. Such a crowded frequency allocation makes obtaining new licensed spectrum bands for developing wireless applications costly and challenging. Although the radio spectrum is a precious resource, experimental tests and investigations from both academia and industries show that the spectrum utilization ratio is actually very low in certain areas [17, 18]. This paradox is caused by the traditional static spectrum management policy that allocates a fixed spectrum band to a single system for exclusive use. The spectrum is underutilized because unlicensed users cannot operate on the licensed spectrum even when licensed users are idle. Therefore, dynamic spectrum sharing (DSS) has emerged as a promising technique by adopting a hierarchical spectrum access structure with primary users (PUs) and secondary users (SUs) [19]. To be specific, SUs are allowed to access the licensed spectrum when PUs are idle or PUs only receive little interference from SUs. In this way, the spectrum can be utilized more efficiently, and thus the spectrum utilization ratio can be increased significantly. In fact, DSS has been announced as the key technology for 5G by many companies and operators around the world including Qualcomm, Ericsson, AT&T, and Verizon [20, 21].

There are some differences between DSS networks and other wireless networks. First, compared to the scheduled resource allocation of cellular networks, the available spectrum resource for the secondary system in the DSS network depends on PUs' activities. Second, avoiding interference from SUs to PUs is extremely important since PUs are the licensed users. DSS techniques aim to increase the spectrum utilization by allowing a secondary system to use the licensed spectrum under the constraint of not interfering the primary system. A common technique is to construct a static protection zone around each PU where co-channel SU-transmissions are not allowed. Even though dynamic protection zones are introduced in recent works [22], the boundary estimation relies on accurate interference estimation [23], which is difficult to achieve in real-world deployment. Therefore, the protection zone technique is not flexible enough to enable efficient utilization of spectrum by SUs. Another common technique is through spectrum sensing, where a SU has to perform spectrum sensing to detect the available spectrum holes before accessing a wireless channel [24]. To be specific, spectrum sensing allows SUs to detect the existence of PUs in a particular channel which utilizes energy detection [25], cyclo-stationary feature detection [26], matched filter based detection [27], and covariance-based detection [28]. The range of spectrum sensing can be divided into narrowband sensing and wideband sensing, which monitors single channel and multiple channel at a particular time, respectively. Wideband sensing divides the spectrum into multiple narrowbands and either senses theses narrowbands sequentially or in a parallel manner. Although wideband sensing can let a SU achieve high data throughput by accessing the best channel among multiple sensed narrowband channels, it is impractical since sequential sensing suffers long sensing time and parallel sensing suffers from high hardware cost and the requirement of sensors synchronization. The pros and cons of narrowband and wideband sensing and more details can be found in [29].

DRL is a suitable framework for developing DSS strategies because the DRL agent can adapt to an unknown environment without modeling the complex wireless systems. However, the real deployment of DRL techniques in DSS systems have been hindered by many practical issues. First, the mobile wireless environments are non-stationary by nature due to many factors, such as user locations, fading, and user traffics [30], which makes allocating spectrum resources to users challenging. Second, obtaining control information from the DSS system is costly. For example, a SU usually cannot detect the existence of all PUs simultaneously because performing spectrum sensing is energy-consuming. Meanwhile, exchanging control information between wireless devices imposes a control overhead on wireless operations. It is challenging to optimize the system performance because collecting training data under limited control information is time-consuming. Furthermore, the non-stationary wireless environment largely decreases the number of effective training data that reflect the latest environment. Under these practical issues, the wireless environment of the DSS system is usually non-stationary and partially observable with extremely limited effective training data. Therefore, in this dissertation we designed an efficient learning framework for developing DSS strategies, which can quickly adapt to an unknown non-stationary environment and only require little training data.

2.2 Literature Review

2.2.1 Conventional DSS strategies

Many strategies have been introduced to achieve spectrum co-existence between SUs and PUs, and they can be classified into two schemes in general. The first scheme is called Listen-Before-Talk (LBT), also known as the interweaving scheme, where a SU can access a frequency band only if it is detected to be available [31]. Although this scheme can effectively avoid causing strong interference to primary users, spectrum opportunities for SUs to access shared frequency bands may be rather limited. This reason is that the spectrum access depends completely on current spectrum sensing outcome in LBT scheme. In reality, due to the dynamic random nature of wireless environments, limited/no cooperation among SUs, and other practical factors spectrum sensing can never be perfect. This will cause false alarm or miss detection of PUs' activities leading SUs to make inappropriate decisions regarding channel access [32]. The second scheme is called spectrum sharing, also known as the underlaying scheme [33]. In this scheme, SUs coexist with PUs on shared frequency bands, and adjust their transmit power level so that the accumulated interference experienced at PUs is less than a tolerable interference threshold. This scheme requires a strong assumption that the channel state information between transmitters of SUs and the receivers of PUs are known as apirori in order to conduct power control. However, in reality, it is usually difficult to obtain these channel state information without a central controller. Even under the presence of a central controller, exchanging these channel state information may impose a heavy control overhead for the underlying network, which makes it difficult to be implemented in practice.

Most non-learning based DSS methods translate the operations in wireless networks into tractable mathematical formulations and then find solutions based on optimization theory. The non-learning based methods predict the spectrum accessibility by introducing statistical models such as Poisson processes [34], linear regressions [35] and Bayesian prediction [36]. However, with the proliferation of wireless applications in 5G networks, the underlying management of spectrum resources become more complicated, hence leading to more complex mathematical formulations. Finding a closed-form solution via classical optimization approaches in such a complex wireless system becomes extremely challenging if not possible. Even if a comprehensive and tractable mathematical formulations can be obtained, the developed algorithms may have high computational complexity making them impractical. Furthermore, obtaining accurate network information for building the mathematical model is costly in the wireless system because the background noise in wireless environments largely decreases the accuracy of the obtained network data. Therefore, many machine learning (ML) approaches have been applied to the DSS problem because of their abilities to adapt to unknown environment without modeling the complex wireless network and requiring accurate network information. ML helps the network to dynamically manage the spectrum resources by analyzing network data, such as user spectrum sensing results and user behavior information.

2.2.2 DRL-based DSS strategies

RL stands out from many ML approaches due to its model-free data analysis characteristic, which substantially improves the spectrum resources allocation in real-world scenarios [37]. DRL-based methods have recently been applied in the DSS networks [38, 39, 40] because their ability to adapt to unknown environments from the partial observation of the system by improving performance on a specific task. However, there are some over-simplified network assumptions in these works. In [38], only one SU is considered in the network with perfect spectrum sensing outcomes (i.e., there is no error in spectrum sensing), but spectrum sensing usually contains lots of noise in reality. On the other hand, [39] assumes that the available spectrum channels are known as a priori and develops a centralized spectrum access algorithm for multi-user access, so it does not consider the interference to PUs. Furthermore, both [38] and [39] focus on the ”access” part of the DSS problem by assuming that one channel can only be utilized by one user at any particular time. Although [40] allows that multiple SUs can access a channel at the same time, a SU cannot access a channel that a PU is using. Meanwhile, [40] assumes that each SU can sense all channels simultaneously and the collision between a PU and a SU can be perfectly detected. In this work, in order to provide a comprehensive study for the impact DSS strategies, we consider practical situations of DSS where 1) a SU cannot conduct spectrum sensing perfectly. 2) the SU cannot sense multiple channels at a particular time. 3) A channel can be shared by the primary system and the secondary system. To handle the partial observations in wireless environments due to the incomplete and noisy spectrum sensing results, we utilize RNNs for the introduced DRL-based DSS strategies, which is also adopted by [41]. Furthermore, we utilize a special type of RNNs called ESNs to reduce the training time and the required training data.

Unlike most previous works utilize binary ACK/NACK feedback as the reward function, we calculate the practical reward based on the spectral-efficiency of the primary system. To reflect the real wireless environment, we built a new network simulator strictly following the 3GPP specifications, where field data are used to construct the underlying wireless environment and the PU traffic. In this new network simulator, the PU activity patterns are affected by multiple factors, including the scheduling algorithm, the PU traffic, and the data transmission procedure. In this way, we can train and evaluate the DSS strategies in realistic wireless application scenarios.

2.3 System Model

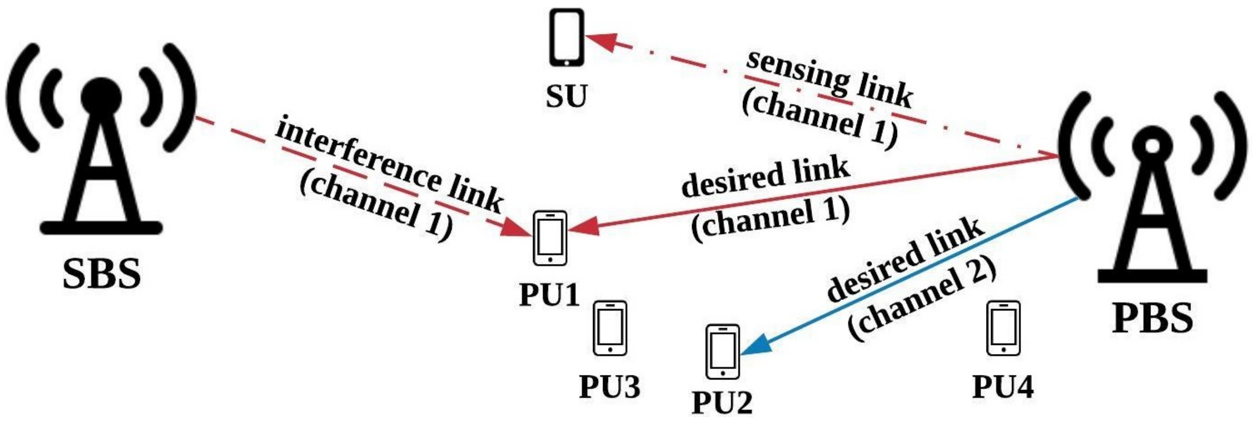


Figure 2.1: The DSS scenario where there are 1 SU, 4 PUs, and 2 wireless channels.

In this section, we describe the system model of the DSS problem and discuss its challenges. DSS is a promising technology to improve the utilization of radio spectrum. DSS allows a SU to access the licensed radio spectrum if PUs only receive tolerable interference. A DSS scenario is illustrated in Figure 3.1, where the primary system consists of a primary base station (PBS) and M PUs, and the secondary system consists of a secondary base station (SBS) and a SU. The primary system has a license to operate on C wireless channels. Our goal is to develop a spectrum access strategy of the SU to increase the overall spectrum utilization without generating intolerable interference to PUs. Each user can only transmit on one channel at a particular time. It is assumed that cross-channel interference is negligible so two users transmit on different wireless channels do not interfere with each other. Without loss of generality, we use a discrete time model to represent the dynamics of the DSS environment, where changes of the environment happen at discrete time slot k (k is a natural number). The PBS is responsible for scheduling the spectrum resources to PUs. To be specific, at time slot k, PBS allocates each channel to one of the PUs that require data transmission based on its scheduling algorithm.

A spectrum opportunity occurs on a channel for two cases: 1) There is no PU conducting data transmission on that channel. 2) The SU can share a channel with the PU if the interference to the PU is tolerable. Unfortunately, obtaining this control information is costly in mobile wireless networks. Furthermore, the control information is outdated quickly in the highly dynamic wireless networks. Therefore, it is impractical to design a DSS strategy by assuming that all the control information is known.

The quality of the wireless connection of a user is determined by its received signal-to-interference-plus-noise ratio (SINR), which is a ratio of the desired signal power to the sum of the interference power and the background noise power. A higher value of the SINR indicates the better quality of the wireless connection. For example, if PBS allocates channel c to PU m at time slot k, the received SINR at PU m is written as

$$\mathrm{SINR}_c^{\mathrm{PBS},m}[k] = \frac{P_c^{\mathrm{PBS}}[k] \cdot \left|H_c^{\mathrm{PBS},m}[k]\right|^2}{P_c^{\mathrm{SBS}}[k] \cdot \left|H_c^{\mathrm{SBS},m}[k]\right|^2 + N_c[k]}, \tag{2.1}$$

where $P_c^{\mathrm{PBS}}[k]$ and $P_c^{\mathrm{SBS}}[k]$ are the transmit power of PBS and SBS on channel c, respectively, $H_c^{\mathrm{PBS},m}[k]$ is the channel gain of the desired link between PBS and PU m on channel c, $H^{\mathrm{SBS},m}$ is the channel gain of the interference link between SBS and PU m on channel c, and $N_c[k]$ is the background noise power on channel c. SBS causes interference to PU m if the SU conducts data transmission on channel c at time slot k ($P_c^{\text{SBS}}[k] > 0$), otherwise there is no interference to PU m at time slot k. Note that the channel gain is dynamic over time and channel, which is affected by many factors such as user locations and fading.

To enable the protection to the primary system, we assume that the primary system will broadcast a warning signal if it experiences a low SINR. The warning signal contains information related to which PU may be interfered so that the SU is aware of the issue. In fact, this kind of warning signal is similar to the control signals used in current 4G and 5G networks. It is common to assume that the control signals are received perfectly by the SU, otherwise the underlying network will not even work. This is the only control information from the primary system to the secondary system to enable the protection for PUs.

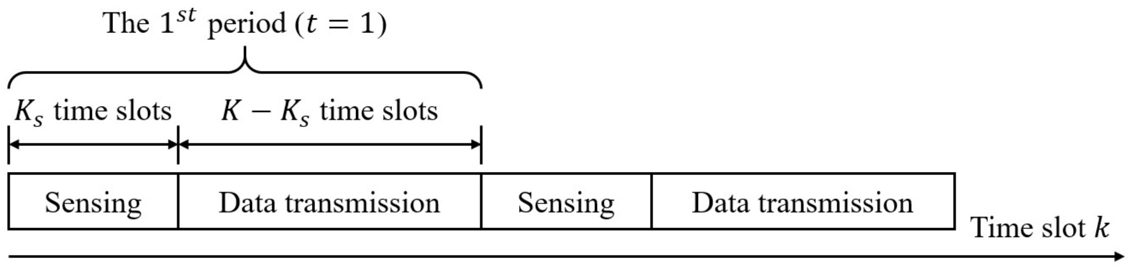


Figure 2.2: Time structure of sensing and data transmission.

The SU performs spectrum sensing to detect the existence of a PU before accessing a channel. Considering the power and complexity constraints, the SU can only sense one channel at a particular time. The energy detector is adopted as the underlying spectrum sensing method, which is the most common one due to its low complexity and cost. We consider the half-duplex spectrum sensing scheme, where a SU cannot transmit data and performs spectrum sensing at the same time. We assume a periodic time structure of spectrum sensing and data transmission as shown in Figure 2.2. To be specific, the t^{th} period contains K time slots (from time slots $(t-1)K+1$ to tK), where the spectrum sensing part contains the first K_s time slots (from time slots $(t-1)K+1$ to $(t-1)K+K_s$), and the data transmission parts contains the subsequent $K - K_s$ time slots (from time slots $(t-1)K + K_s + 1$ to tK). Accordingly, in the t^{th} period, the SU performs spectrum sensing on channel c by computing the energy of received signals as follows:

$$E_c[t] = \sum_{k=(t-1)K+1}^{(t-1)K+K_s} \left| \sqrt{P_c^{\text{PBS}}[k]} \cdot H_c^{\text{PBS,SU}}[k] + \omega_c[k] \right|^2, \tag{2.2}$$

where $\omega_c[k] \sim \mathcal{CN}(0, N_c[k])$ is a circularly-symmetric Gaussian noise on channel c, $P_c^{\text{PBS}}[k]$ is the transmit power of PBS on channel c, and $H_c^{\text{PBS,SU}}[k]$ is the channel gain of the sensing link between PBS and the SU on channel c. Note that if PBS allocates channel c to one PU at time slot k, then $P_c^{\text{PBS}}[k]$ is a positive value, otherwise $P_c^{\text{PBS}}[k]$ is zero.

If the computed energy is lower than a threshold, conventional energy detectors regard channel c as a spectrum opportunity in the t^{th} period. However, setting the threshold is challenging because it requires the sensing link's channel gain information, which is difficult to obtain in the real wireless environment. Furthermore, setting a threshold is difficult in some cases because of the relative locations of the base station and users. As shown in Figure 3.1, the sensing link is between PBS and the SU, while the interference link is between SBS and the PU. The discrepancy of the sensing link and the interference link may cause the problem that the sensing link is weak but the interference link is strong. For example, the SU and PBS are far away from each other while SBS is close to the PU. In this case, the SU cannot detect the existence of PBS, but SBS will cause strong interference to the PU. The warning signals from PUs are designed to provide additional protection to the primary system for the case where the SU cannot detect the existence of PBS. Therefore, instead of making the spectrum access decision solely based on outcomes of the energy detector, we developed a DRL framework to construct a new spectrum access policy: The DRL agent will use the sensed energy as the input to learn a spectrum access strategy to maximize the spectrum utilization while enabling the protection to PUs.

2.4 DRL Problem Formulation

We now formulate the DSS problem using the DRL framework. To be specific, we assume that the SU has a DRL agent that takes its observation as the input and learns how to perform spectrum sensing and access actions in order to maximize its cumulative reward. The reward for the SU is designed to maximize the spectrum utilization while preventing harmful interference to PUs. The observation of the SU in the t^{th} period is denoted by

$$o_t = (E[t], Q[t]) , \tag{2.3}$$

where $E[t]$ is the energy of received signals, and $Q[t]$ is a one-hot C-dimensional vector indicating the sensed channel in the t^{th} period. If the index of the sensed channel is c, then the c^{th} element of $Q[t]$ is equal to one while other elements of $Q[t]$ are zeros, and $E[t]$ is equal to $E_c[t]$ that is calculated by Equation (2.2). Accordingly, the dimension of the observation space is $(C + 1)$.

The action of the SU in the t^{th} period is denoted by

$$a_t = (q[t], z[t]) , \tag{2.4}$$

where $q[t] \in \{0, 1\}$ represents the SU will either access the current sensed channel $(q[t] = 1)$ or be idle $(q[t] = 0)$ during the data transmission part of the t^{th} period, $z[t] \in \{1, ..., C\}$ represents the SU will sense the channel $z[t]$ during the sensing part of the $(t+1)^{\text{th}}$ period. In other words, the SU makes two decisions: $q[t]$ decides whether to conduct data transmission in the current sensed channel of the t^{th} period and $z[t]$ decides which channel to sense in the

$(t+1)^{\text{th}}$ period. Accordingly, the dimension of the action space is $2C$. Note that the sensed channel in the t^{th} period may be different from that in the $(t+1)^{\text{th}}$ period

Table 2.1: The SINR and CQI mapping to modulation and coding rate.

CQI	SINR ($\geq$)	modulation	code rate ($\times$1024)	spectral-efficiency (bits/symbol)
0			out of range	
1	-6.7	QPSK	78	0.1523
2	-4.7	QPSK	120	0.2344
3	-2.3	QPSK	193	0.3770
4	0.2	QPSK	308	0.6016
5	2.4	QPSK	449	0.8770
6	4.3	QPSK	602	1.1758
7	5.9	16QAM	378	1.4766
8	8.1	16QAM	490	1.9141
9	10.3	16QAM	616	2.4063
10	11.7	64QAM	466	2.7305
11	14.1	64QAM	567	3.3223
12	16.3	64QAM	666	3.9023
13	18.7	64QAM	772	4.5234
14	21.0	64QAM	873	5.1152
15	22.7	64QAM	948	5.5547

In our work, we use a discrete reward function which is similar to the existing DRL-based DSS methods. Compared to a simple binary reward (0 and +1, -1 and +1) in [38] and [39], we consider a more relevant and comprehensive reward design that is based on the underlying achieved modulation and coding strategy (MCS) adopted in the 3GPP LTE/LTE-Advanced standard [42]. To be specific, a receiver measures SINR to evaluate the quality of the wireless connection and feedback the corresponding Channel Quality Indicator (CQI) to the transmitter [43]. We follow the method presented in [44] to map the received SINR to the CQI. After receiving the CQI, the transmitter determines the MCS for data transmission based on the CQI table specified in the 3GPP standard [42]. The SINR and CQI mapping to MCS is given in Table 2.1 for reference. Accordingly, the achieved spectral-efficiency can be calculated by (bits/symbol) = (modulation's power of 2) $\times$ (code rate) representing the average information bits per symbol. This critical metric is utilized as the reward function of our design.

The reward function corresponding to the SU accessing channel c in the t^{th} period depends on the average spectral-efficiency of the primary system. To be specific, we calculate the average spectral-efficiency of PUs $\bar{e}_c^{\text{PU}}[t]$ during the t^{th} data transmission period of the SU as follows

$$\bar{e}_c^{\text{PU}}[t] = \frac{1}{|\Psi_c^{\text{PU}}[t]|} \sum_{k \in \Psi_c^{\text{PU}}[t]} e_c^{\text{PU}}[k] \tag{2.5}$$

where $\Psi_c^{\text{PU}}[t]$ is the set of time slots when PBS allocates channel c to one PU in the SU's t^{th} data transmission period (from time slots $(t-1)K + K_s + 1$ to tK), and $e^{\text{PU}}[k]$ is the spectral-efficiency of the PU that conducts data transmission on channel c at time slot k. Note that PBS allocates each channel to only one PU at a particular time slot.

The reward of the SU in the t^{th} period is defined as

$$r_t = \begin{cases} 0, & \text{SU is idle} \\ -1, & \text{SU accesses channel } c \text{ and } \bar{e}_c^{\text{PU}}[t] < 3 \\ 1, & \text{SU accesses channel } c \text{ and } \bar{e}_c^{\text{PU}}[t] \geq 3 \end{cases} \tag{2.6}$$

The reward r_t is 0 if the SU is idle in the t^{th} transmission period. To enable the protection for the primary system, the primary system will broadcast a warning signal if its average spectral-efficiency is below 3, and then the reward received by the SU is -1. If the primary system does not suffer from strong interference (the average spectral-efficiency of PUs is larger than 3), the reward r_t is 1.

2.5 Deep Echo State Network (DEQN)

DRQN has been introduced to capture the temporal correlation of observation sequences in POMDP environments. However, the training of RNNs is known to be difficult that suffers from vanishing and the exploding gradients problems. The standard training technique for RNNs, backpropagation-through-time (BPTT), is to unfold the network in time into a computational graph that has a repetitive structure. BPTT suffers from the slow convergence rate and needs many training samples. Furthermore, both the required amount of training data and required training time for achieving convergence are large in the DRL framework, since the DRL agent has to find a good policy by exploring the environment with different potential policies. Therefore, training DRQNs under non-stationary environments with limited available training data is extremely challenging.

In light of training challenges of DRQNs, we introduce DEQN by adopting ESN as the kernel of deep Q-network to reduce the required training time and the training data [9]. ESNs simplify the BPTT by only training the output weights while leaving input weights and recurrent weights untrained. Existing research shows that ESNs can achieve comparable performance with RNNs, especially in some applications requiring fast learning [12]. Since the hidden states are fixed in the ESNs, our DEQN training algorithm can pre-store hidden states. In this way, DEQNs can substantially decrease the amount of training data by avoiding recalculating hidden states in every training iteration. Compared to DRQNs, DEQNs will have a much faster training rate with better utilization of the training data.

The action-value function in DRQN/DEQN is approximated by the Q-network Q_θ with parameter $\theta = (W_{\text{in}}, W_{\text{rec}}, W_{\text{out}})$, where $W_{\text{in}} \in \mathbb{R}^{d_h \times d_o}$ is the input weights, $W_{\text{rec}} \in \mathbb{R}^{d_h \times d_h}$ is the recurrent weights, and $W_{\text{out}} \in \mathbb{R}^{d_y \times d_h}$ is the output weights. A sequence of observations $o_1, \ldots, o_t \in \mathbb{R}^{d_o}$ is input to the Q-network to generate a sequence of outputs $y_1, \ldots, y_t \in \mathbb{R}^{d_y}$

as follows:

$$\begin{aligned} h_t &= \tanh\left(W_{\text{in}} o_t + W_{\text{rec}} h_{t-1}\right) \in \mathbb{R}^{d_h}, \\ y_t &= W_{\text{out}} h_t \in \mathbb{R}^{d_y}, \end{aligned} \tag{2.7}$$

where h_t is the hidden state. The hidden state h_t is used to represent a summary of the past sequence of observations up to t and we set $h_0 = \mathbf{0}$. Note that the a^{th} element of y_t is equal to the estimated Q-value of selecting action a, i.e., $y_t^a = Q_\theta\left(o_{\leq t}, a\right)$, where $o_{\leq t} = (o_1, \ldots, o_t)$. The loss function for training θ is

$$\left(r_t + \gamma \max_a Q_{\theta^-}\left(o_{\leq t+1}, a\right) - Q_\theta(o_{\leq t}, a_t)\right)^2, \tag{2.8}$$

where θ^- is the parameter of target Q-network. To stabilize the training targets, θ^- is periodically synchronized with θ instead of being updated in each training iteration.

2.5.1 DEQN Training Algorithm

BPTT involves unfolding the network in time into a computational graph that has a repetitive structure, which suffers from the slow convergence rate due to vanishing and exploding gradients [7]. Furthermore, DRQN requires a large amount of training data to ensure the learning agent converges to an appropriate policy. Unfortunately, in the DSS problem, the DRL agent can only collect limited training data and the wireless environment is usually non-stationary. To quickly adapt to the non-stationary wireless environment using limited training data, we design an online training algorithm for DEQN.

The designed DEQN training algorithm is stated in Algorithm 1. First, the input weights and the recurrent weights of ESNs are initialized randomly, and then they remain untrained. Only the output weights of ESNs are trained so the training is extremely fast by avoiding BPTT. Second, given $(o_1, \ldots, o_t)$, we can observe that the hidden states $(h_1, \ldots, h_t)$ are

Algorithm 1 The online training algorithm for DEQN.

```
Initialize the number of episodes $L$, the number of samples in one episode $T$, the exploration probability $\epsilon$, and the discount factor $\gamma$.
Initialize an evaluation network $\theta$, a target network $\hat{\theta}$, and a memory buffer $D$ for the agent.
Initialize a Q-network $\theta$ and a target Q-network $\theta^-$.
for $l = 1, ..., L$ do
    Set $\theta^- = \theta$ and empty the replay buffer $D$.
    Observe $o_1$ from the environment.
    for $t = 1, ..., T$ do
        Input $o_t$ and $h_{t-1}$ to $Q_\theta$ to calculate $h_t$ and $y_t$.
        Select action $a_t$ based on $\epsilon$-greedy policy.
        Execute $a_t$ based on $y_t$ and then receive $r_t$.
        Observe $o_{t+1}$ from the environment.
        Input $o_{t+1}$ and $h_t^-$ to $Q_{\theta^-}$ to calculate $h_{t+1}^-$.
        Store $(h_t, a_t, r_t, h_{t+1}^-)$ in replay buffer $D$.
    end for
    Sample $(h_j, a_j, r_j, h_{j+1}^-)$ from replay buffer $D$.
    Update $\theta$ by performing gradient descents on
        $\Big(r_j + \gamma \max_a Q_{\theta^-}(h_{j+1}^-, a) - Q_\theta(h_j, a_j)\Big)^2.$
end for
```

unchanged during the training process from Equation (2.7) because the input weights and recurrent weights are fixed. In contrast to DRQNs that waste some training samples and computational resources to recalculate the hidden states in every training iteration, DEQNs can pre-store the hidden states in the replay buffer and use them for training. Therefore, the DEQN training is much more sample and computationally efficient than the DRQN training. Third, the DEQN training can randomly sample from the replay buffer to create a training batch because the hidden states are unchanged and pre-stored. On the other hand, the DRQN training has to sample continuous sequences to create a training batch. Breaking the temporal correlations of sampled data during training is crucial for reducing generalization error, as the stochastic optimization algorithms usually assume i.i.d. data. Lastly, to deal with the non-stationary environment, the outdated training data in the replay buffer will be removed and the policy will be updated continually by the effective data collected from the latest environment.

2.6 Theoretical Analysis for DEQN

In this section, we first provide the convergence analysis for both DRQN and DEQN to demonstrate the faster convergence of DEQN over DRQN. Second, we characterize the bias-variance tradeoff of DEQN to select the maximum spectrum norm constraint for ESNs' recurrent weights. In the following analysis, we denote $\|\cdot\|$ as the l_2 norm of vectors and the spectral norm of matrices. The proofs of theorems in this section are presented in the supplementary file due to the page limitation.

To simplify the theoretical analysis while providing sufficient insights, we will make several assumptions. We make Assumption 2.1 because if $o_{\leq t}$ does not contain enough information for predicting reward and future state, no RL algorithms will be able to work.

Assumption 2.1. The observation sequence up to time t, $o_{\leq t} = (o_1, \dots, o_t)$, contains the sufficient information for determining the optimal action at time t.

For DRQN/DEQN, we can define a policy π that depends on the observation history. Under Assumption 2.1, the optimal action-value function Q^* is defined as

$$Q^*(o_{\leq t}, a_t) = \sup_{\pi} Q^{\pi}(o_{\leq t}, a_t), \ \ \forall (o_{\leq t}, a_t) \in \Omega_{\leq t} \times \mathcal{A}, \tag{2.9}$$

where $\Omega_{\leq t} = \prod_{i=1}^{t} \Omega$ and $Q^{\pi}(o_{\leq t}, a_t)$ is the action-value function under the policy π.

In Algorithm 2, we simplify the DRQN/DEQN training algorithm as a Neural Fitted Q-iteration algorithm [45], which is a common framework for RL theoretical analysis using function approximations [46, 47]. The output of Algorithm 2 is the greedy policy π_L with

Algorithm 2 Neural Fitted Q-Iteration Algorithm.

```
Initialize the number of episodes $L$.
Initialize the number of samples in one episode $T$.
Initialize a Q-network $\theta_0 \in \Theta$ and a memory buffer $Z$.
for $l = 1, ..., L$ do
    Sample $T$ transitions $(o_t, a_t, r_t, o_{t+1})$ from the environment and stores in buffer $Z$.
    Calculate the targets
    $z_t = r_t + \gamma \max_{a'} Q_{\theta_{l-1}}(o_1, ..., o_{t+1}, a')$.
    Update the action-value function:
    $\theta_l \leftarrow \underset{\theta\in\Theta}{\operatorname{argmin}} \frac{1}{T}\sum_{t=1}^{T}(z_t - Q_\theta(o_1, ..., o_t, a_t))^2$
end for
Define $\pi_L$ as the greedy policy with respect to $Q_{\theta_L}$.
```

respect to the learned Q-network Q_{θ_L}. To simplify the underlying theoretical analysis, we make the following assumption.

Assumption 2.2. The optimal weights θ_l can be characterized in each Q-iteration of Algorithm 2.

Note that Assumption 2.2 can be achieved relatively easily for DEQNs with a reasonably large number of training iterations. This is because only the output weights of ESNs are trainable, while it is much more challenging to be achieved for DRQNs due to the underlying training issues of RNNs. However, we still let Assumption 2.2 hold for both DRQN and DEQN even though it is biased towards DRQNs.

Theorem 2.1. *(Theorem 6.1 in [48]) Let σ be the sampling distribution over $\Omega_{\leq t} \times \mathcal{A}$ in Algorithm 2, μ be a fixed probability distribution over $\Omega_{\leq t} \times \mathcal{A}$, and R_{max} be the maximum reward value. Then we have*

$$\mathbb{E}_\mu\left[\left|Q^* - Q^{\pi_L}\right|\right] \leq \frac{2\phi_{\mu,\sigma}\gamma}{(1-\gamma)^2} \cdot \eta_{max,t} + \frac{4\gamma^{L+1}}{(1-\gamma)^2} \cdot R_{max}, \tag{2.10}$$

where $\phi_{\mu,\sigma}$ is the concentration coefficient of μ and σ, π_L is the output policy of Algorithm 2, *and $\eta_{max,t}$ is the maximum one-step approximation error.*

Theorem 2.2. *(Theorem 6.2 in [48]) Let $\epsilon > 0$ and $V_{max} = R_{max}/(1-\gamma)$. From Theorem 6.2 in [48], the upper-bound of $\eta_{max,t}$ can be written as*

$$\begin{aligned}\eta^2_{max,t} \leq & 4 \sup_{\theta' \in \Theta} \inf_{\theta \in \Theta} \mathbb{E}_\sigma \left[(\mathcal{B}Q_{\theta'} - Q_\theta)^2 \right] \\ & + C_1 \cdot V^2_{max}/T \cdot log N^{ext}_{\epsilon,t} + C_2 \cdot V_{max} \cdot \epsilon,\end{aligned} \tag{2.11}$$

where C_1 and C_2 are constants, $\mathcal{B}$ is the Bellman optimality operator, and $N^{ext}_{\epsilon,t}$ is the exterior ϵ-covering number of RNN/ESN with respect to the Euclidean norm.

The first term in Equation (2.10) represents a statistical error while the second term represents an algorithmic error. It is clear that the algorithmic error converges to zero as the Neural Fitted Q-iteration algorithm proceeds. On the other hand, the statistical error term is bounded by Equation (2.11), which characterizes the bias and variance of estimating the action-value function using neural networks. The first term in Equation (2.11) corresponds to the bias incurred by approximating the target $\mathcal{B}Q_{\theta'}$ using RNN/ESN. It can be viewed as a measure of the completeness of RNN/ESN with respect to the Bellman operator $\mathcal{B}$. The second term and the third term in Equation (2.11) correspond to the variance of estimating action-value functions. Since the variance represents the model's sensitivity to small fluctuations in the training set, the convergence rate of DRQN/DEQN scales with the estimator variance [49].

In this paper, we analyze the covering numbers of RNNs and ESNs to compare the underlying convergence rates of DRQNs and DEQNs. To provide an analytical characterization of the covering numbers for RNN/ESN, we will make some mild assumptions to bound inputs and spectral norms of weight matrices of DRQN/DEQN. Note that these assumptions are commonly used for the analysis of neural networks [50, 51].

Assumption 2.3. The input is bounded: $\|o_t\| \leq B_o$.

Assumption 2.4. The spectral norms of weight matrices are bounded:

$$\|W_{\text{in}}\| \leq \sigma_{\text{in}}^{\max}, \quad \|W_{\text{rec}}\| \leq \sigma_{\text{rec}}^{\max}, \quad \|W_{\text{out}}\| \leq \sigma_{\text{out}}^{\max}. \tag{2.12}$$

Furthermore, we make $\sigma_{\text{rec}}^{\max} < 1$ to satisfy the Echo State Property [9].

A function class constructed by RNN/ESN represents a family of functions that maps $o_{\leq t}$ to y_t. To begin our analysis, we specify two types of function classes where functions are represented by Equation (2.7) with different weight matrices. The first type of function class, $A_t(a, b)$, contains a family of functions with weights satisfying $\|W_{\text{in}}\| \leq a, \|W_{\text{rec}}\| \leq b, \|W_{\text{out}}\| \leq \sigma_{\text{out}}^{\max}$. The second function class, $B_t(a, b)$, contains a family of functions with weights satisfying $\|W_{\text{in}}\| = a, \|W_{\text{rec}}\| = b, \|W_{\text{out}}\| \leq \sigma_{\text{out}}^{\max}$. Since all weights are trainable in RNNs, the function class constructed by RNNs is $A_t(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max})$ based on Assumption 2.4. On the other hand, the function class constructed by ESNs is random because input weights and recurrent weights of ESNs are initialized randomly and are untrained. Let S_{in} and S_{rec} be two random variables representing the spectral norm of random input weights and random recurrent weights, respectively. According to Assumption 2.4, the interval of S_{in} is $(0, \sigma_{\text{in}}^{\max}]$ and the interval of S_{rec} is $(0, \sigma_{\text{rec}}^{\max}]$. The probability density function (PDF) of S_{in} and S_{rec} are denoted by $f_{S_{\text{in}}}(\cdot)$ and $f_{S_{\text{rec}}}(\cdot)$, respectively. Then the random function class constructed by ESNs is denoted as $B_t(S_{\text{in}}, S_{\text{rec}})$.

Let the exterior ϵ-covering number of RNNs and ESNs be $N_{\text{RNN},\epsilon,t}^{\text{ext}}$ and $\mathcal{N}_{\text{ESN},\epsilon,t}^{\text{ext}}$, respectively. Note that $\mathcal{N}_{\text{ESN},\epsilon,t}^{\text{ext}}$ is a random variable because the function class constructed by ESNs is random. To analyze the theoretical properties of DEQN, we define $N_{\text{ESN},\epsilon,t}^{\text{ext}} = \mathbb{E}[\mathcal{N}_{\text{ESN},\epsilon,t}^{\text{ext}}]$ and use $N_{\text{ESN},\epsilon,t}^{\text{ext}}$ for analysis without loss of generality. Theorem 2.3 and Theorem 2.4 contain the comparison of ϵ-exterior covering numbers of RNNs and ESNs.

Theorem 2.3. *$\mathcal{N}^{ext}_{ESN,\epsilon,t}$ is less than or equal to $N^{ext}_{RNN,\epsilon,t}$ almost surely.*

Theorem 2.4. *The lower-bounds and the upper-bounds of $N^{ext}_{RNN,\epsilon,t}$ and $N^{ext}_{ESN,\epsilon,t}$ are written as follows.*

$$N^{ext}_{RNN,\epsilon,t} \geq L\left(\sigma^{max}_{in}, \sigma^{max}_{rec}\right), \tag{2.13}$$

$$N^{ext}_{RNN,\epsilon,t} \leq U(\sigma^{max}_{in}, \sigma^{max}_{rec}) \tag{2.14}$$

$$N^{ext}_{ESN,\epsilon,t} \geq \iint\limits_{\substack{0<a\leq\sigma^{max}_{in}\\0<b\leq\sigma^{max}_{rec}}} L\left(a,b\right) f_{S_{in}}(a) f_{S_{rec}}(b) \mathrm{d}a\mathrm{d}b, \tag{2.15}$$

$$N^{ext}_{ESN,\epsilon,t} \leq \iint\limits_{\substack{0<a\leq\sigma^{max}_{in}\\0<b\leq\sigma^{max}_{rec}}} U(a,b)\, f_{S_{in}}(a) f_{S_{rec}}(b) \mathrm{d}a\mathrm{d}b, \tag{2.16}$$

where

$$U(a,b) = \left[1 + \frac{2B_o\sigma^{max}_{out}a\left(1-b^t\right)}{\epsilon(1-b)}\right]^{d_y},$$

$$L\left(a,b\right) = \left[\frac{\sigma^{max}_{out}H_t(a,b)}{\epsilon}\right]^{d_y},$$

$$H_t(a,b) = tanh\left(aB_o + bH_{t-1}(a,b)\right),$$

$$H_1(a,b) = tanh\left(aB_o\right).$$

Furthermore, $N^{ext}_{RNN,\epsilon,t} - N^{ext}_{ESN,\epsilon,t}$ is lower bounded by

$$\iint\limits_{\substack{0<a\leq\sigma^{max}_{in}\\0<b\leq\sigma^{max}_{rec}}} \left[L\left(\sigma^{max}_{in}, \sigma^{max}_{rec}\right) - U\left(a,b\right)\right]^{+} f_{S_{in}}(a) f_{S_{rec}}(b) \mathrm{d}a\mathrm{d}b, \tag{2.17}$$

where $[x]^+ = \max(0,x)$.

It is important to note that Theorem 2.3 and Theorem 2.4 show that the exterior covering number of ESNs is smaller than that of RNNs. This means that DEQN has a smaller variance of estimating optimal Q-value compared to DRQN, suggesting a faster convergence rate of DEQN. From Equation (2.17), it can be seen that the lower-bound of the gap between $N^{\text{ext}}_{\text{RNN},\epsilon,t}$ and $N^{\text{ext}}_{\text{ESN},\epsilon,t}$ depends on $f_{S_{\text{in}}}(\cdot)$ and $f_{S_{\text{rec}}}(\cdot)$. Since W_{in} and W_{rec} of ESNs are randomly generated, we can identify $f_{S_{\text{in}}}(\cdot)$ and $f_{S_{\text{rec}}}(\cdot)$ using random matrix theory.

Algorithm 3 Initialize W_{in} and W_{rec} of ESNs.

```
Each entry of $W_{\text{in}}$ and $W_{\text{rec}}$ follows $\mathcal{N}(0,1)$.
$W_{\text{in}} \leftarrow W_{\text{in}} \cdot \sigma^{\max}_{\text{in}} / (3\gamma_{\text{in}} + \mu_{\text{in}})$, where
    $\mu_{\text{in}} = \left(\sqrt{d_o - 0.5} + \sqrt{d_h - 0.5}\right)^2,$
    $\gamma_{\text{in}} = \sqrt{\mu_{\text{in}}}\left(\frac{1}{\sqrt{d_o - 0.5}} + \frac{1}{\sqrt{d_h - 0.5}}\right)^{\frac{1}{3}}.$
$W_{\text{rec}} \leftarrow W_{\text{rec}} \cdot \sigma^{\max}_{\text{rec}} / (3\gamma_{\text{rec}} + \mu_{\text{rec}})$, where
    $\mu_{\text{rec}} = 4 \cdot (d_h - 0.5),$
    $\gamma_{\text{rec}} = \sqrt{\mu_{\text{rec}}} \cdot \left(\frac{2}{\sqrt{d_h - 0.5}}\right)^{\frac{1}{3}}.$
If $\|W_{\text{in}}\| > \sigma^{\max}_{\text{in}}$, then $W_{\text{in}} \leftarrow W_{\text{in}} \cdot \sigma^{\max}_{\text{in}} / \|W_{\text{in}}\|$.
If $\|W_{\text{rec}}\| > \sigma^{\max}_{\text{rec}}$, then $W_{\text{rec}} \leftarrow W_{\text{rec}} \cdot \sigma^{\max}_{\text{rec}} / \|W_{\text{rec}}\|$.
```

To satisfy the spectrum constraints on weights in Assumption 2.4, W_{in} and W_{rec} of ESNs are initialized using Algorithm 3. Specifically, each entry of W_{in} and W_{rec} of ESNs is generated from independent standard normal distribution, and then W_{in} and W_{rec} are scaled to satisfy the spectrum norm constraints. Note that the normalized spectrum norm distribution of large Gaussian random matrix follows Tracy-Widom distribution [52]. To be specific, if W is a large $m \times n$ Gaussian random matrix, then we can obtain a type-1 Tracy-Widom random variable T_W as the following:

$$T_W = \left(\|W\| - \mu_W\right) / \gamma_W, \tag{2.18}$$

where

$$\mu_W = \left(\sqrt{m-0.5} + \sqrt{n-0.5}\right)^2,$$
$$\gamma_W = \sqrt{\mu_W}\left(1/\sqrt{m-0.5} + 1/\sqrt{n-0.5}.\right)^{\frac{1}{3}}$$

Let the spectrum norm constraint be $\sigma^{\max}$. Since $\Pr\left(\|W\| \le 3\gamma_W + \mu_W\right) = \Pr\left(T_W \le 3\right) \approx 1$, we first multiply W by $\sigma^{\max}/(3\gamma_W + \mu_W)$ to have $\Pr\left(\|W\| \le \sigma^{\max}\right) \approx 1$. If $\|W\|$ is still larger than $\sigma^{\max}$, then we multiply W by $\sigma^{\max}/\|W\|$ to make $\|W\| = \sigma^{\max}$. After generating W_{in} and W_{rec} of ESNs using Algorithm 3, $f_{S_{\text{in}}}(a)$ and $f_{S_{\text{in}}}(b)$, are represented in Theorem 2.5.

Theorem 2.5. *Under Algorithm 3, $f_{S_{in}}(a)$ and $f_{S_{in}}(b)$ are*

$$f_{S_{in}}(a) = \delta\left(a - \sigma_{in}^{max}\right) \cdot \left[1 - F_{T_{in}}(3)\right] + \mathbb{1}\left(a < \sigma_{in}^{max}\right) \cdot K_{in} \cdot f_{S_{in}}\left(K_{in}a - \frac{\mu_{in}}{\gamma_{in}}\right), \tag{2.19}$$
$$f_{S_{rec}}(b) = \delta\left(b - \sigma_{rec}^{max}\right) \cdot \left[1 - F_{T_{rec}}(3)\right] + \mathbb{1}\left(b < \sigma_{rec}^{max}\right) \cdot K_{rec} \cdot f_{S_{rec}}\left(K_{rec}b - \frac{\mu_{rec}}{\gamma_{rec}}\right), \tag{2.20}$$

where

$$K_{in} = \frac{3\gamma_{in} + \mu_{in}}{\gamma_{in}\sigma_{in}^{max}},$$
$$K_{rec} = \frac{3\gamma_{rec} + \mu_{rec}}{\gamma_{rec}\sigma_{rec}^{max}},$$

T_{in} and T_{rec} are type-1 Tracy-Widom random variables, $F_{T_{in}}(\cdot)$ and $F_{T_{rec}}(\cdot)$ are cumulative distribution function (CDF) of T_{in} and T_{rec}, respectively, $\delta(\cdot)$ is the delta function, and $\mathbb{1}(\cdot)$ is the indicator function.

In order for ESNs to work properly, the reservoir has to satisfy the Echo State Property in Definition 2.6.

Definition 2.6. Let $h_{t+N}(o_t^{t+N}, h_{t-1})$ be the hidden state at time $t+N$ of Equation (2.7) given the hidden state at time $t-1$ and the input sequence $o_t^{t+N} = (o_t, \ldots, o_{t+N})$. An ESN is said to satisfy the Echo State Property whenever for any $h_{t-1} = a, h_{t-1} = b$, and for any

o_t^{t+N}, it holds that:

$$\left\| h_{t+N}(o_t^{t+N}, a) - h_{t+N}(o_t^{t+N}, b) \right\| \to 0 \ \text{as} \ N \to \infty.$$

The Echo State Property means that ESNs will asymptotically wash out any information from initial hidden states as time passes, which can be satisfied by $\sigma_{\text{rec}}^{\max} < 1$ [9]. However, it does not specify a sufficiently accurate design principle of determining $\sigma_{\text{rec}}^{\max}$. Therefore, we derive a new bias-variance decomposition by extending Theorem 2.2, which is presented in Theorem 2.7

Theorem 2.7. *The upper-bound of $\eta_{max,t}$ can be written as*

$$\eta_{max,t}^2 \leq 4 \max\left(0, \frac{\sqrt{d_y} R_{max}}{1-\gamma} - \epsilon (N_{\epsilon,t}^{ext})^{\frac{1}{d_y}}\right)^2 + C \cdot log N_{\epsilon,t}^{ext} + C', \tag{2.21}$$

where

$$C = \left(8\sqrt{2T} + \frac{256}{V_{max}}\right) \cdot \frac{V_{max}^2}{T},$$
$$C' = (16 + 4\sqrt{2n} + 36) \cdot V_{max} \cdot \epsilon.$$

Corollary 2.8. *Let $L_{\epsilon,t}^{ESN}$ be the lower-bound of $N_{ESN,\epsilon,t}^{ext}$ and $U_{\epsilon,t}^{ESN}$ be the upper-bound of $N_{ESN,\epsilon,t}^{ext}$. Then we can obtain*

$$\sum_{t=1}^{T} \eta_{max,t}^2 \leq \sum_{t=1}^{T} \left(C \cdot log U_{\epsilon,t}^{ESN} + C'\right) + \sum_{t=1}^{T} 4 \max\left(0, \frac{\sqrt{d_y} R_{max}}{1-\gamma} - \epsilon \left(L_{\epsilon,t}^{ESN}\right)^{\frac{1}{d_y}}\right)^2, \tag{2.22}$$

where the first term and the second term represent the variance and the squared bias of estimating the action-value function using ESNs, respectively.

Corollary 2.8 can provide a method to determine $\sigma_{\text{rec}}^{\max}$ by minimizing the sum of the squared bias and the variance in Equation (2.22) through numerical evaluation. To be specific, $f_{S_{\text{in}}}(a)$

and $f_{S_{\text{in}}}(b)$ in Theorem 2.5 are used to calculate the squared bias and the variance.

2.7 Performance Evaluation

In this section, we evaluate the performance of the introduced DEQN methodology in the DSS scenario of a relevant wireless network. We develop a network simulator to provide practical radio environments in wireless networks by incorporating field measurements obtained from a 10-cell ray-tracing area in a city and real-world spectrum occupancy database [53]. All users randomly move at 0.7m/s - 1m/s based on the random waypoint model [54], which is commonly used for modeling the movement of mobile users in mobility management. Figure 2.3 shows the DSS scenario in our experiment, where there are 1 SU and 12 PUs in a 400m×400m area. The primary system has a license to operate on 3 wireless channels ($d_o = 4$ and $d_y = 6$), while the SU has to access the channels without generating intolerable interference to the primary system. For each channel, the bandwidth is set to 10MHz and the variance of the Gaussian noise is set to -157.3dBm/Hz. The transmit power of PBS and SBS are both set to 400mW. We set the sensing and data transmission period K to 100 time slots and the sensing duration K_s to 20 time slots, where one time slot represents interval of 1ms. We list all the parameters to generate the wireless environment in Table 2.2.

2.7.1 Network Simulator

To provide realistic performance evaluation, we develop a network simulator to provide practical radio environments in a real wireless network by incorporating field measurements and real-world datasets. First, we utilize the field measurement data obtained from a 37-cell ray-tracing area in a city to build the channel gains between the base station and the users.

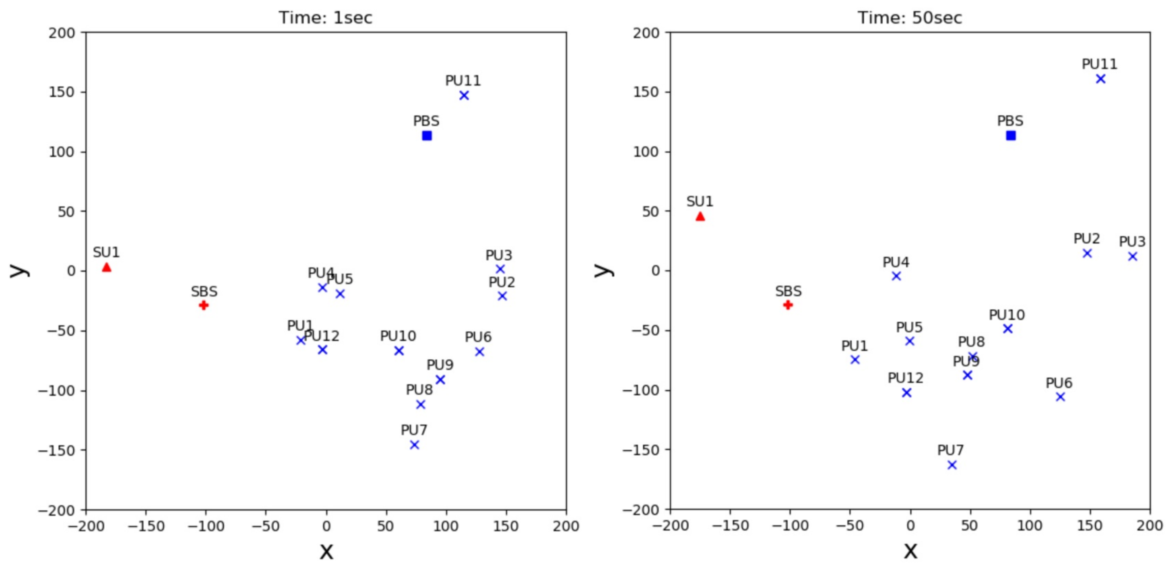


Figure 2.3: A snapshot of the DSS network geometry at 1s and 50s, where there are 1 SU and 12 PUs. PBS/SBS represent the base station of PU/SU.

As shown in Figure 2.4, the channel gains used in the simulation are extracted from cell 23-25 and cell 35-37, where the channel gains on 3 wireless channels from SBS and PBS are obtained from cell 23-25 and cell 35-37, respectively. Figure 2.5 shows the spectral-efficiency of a PU on channel 1-3 at different locations within the simulation area when the PU and the SU are operating on the same channel.

Second, we use real-world spectrum occupancy data for generating the data traffics of PUs. Most of the literature [55] assumes that the PU's activity follows a Markov chain which may not be realistic. In our evaluation, the PU's activity is determined by the scheduling algorithm of PBS and its data traffic reflecting practical and relevant wireless network operations. In realistic scenarios, the data traffic of a user usually consists of several sequences of packets with periods of inactivity in between [56]. Therefore, we extract the per-user data traffic of PU from RWTH Aachen University's spectrum occupancy database [53, 57] based on field measurements. Each PU has a transmit buffer for requiring unsent data from PBS. The PBS schedules spectrum resources to PUs that require data transmission based on the

Table 2.2: The values of parameters for generating the wireless environment.

Parameter	**Value**
number of PUs M	12
number of channels C	3
simulation area	400m×400m
transmit power of PBS	400mW
transmit power of SBS	400mW
variance of Gaussian noise	−157.3dBm/Hz
bandwidth of a channel	10MHz
interval of one time slot	1ms
sensing and transmission period K	100 time slots
sensing duration K_s	20 time slots

proportional fair scheduling algorithm [58], which is designed to maintain fairness among users while ensuring good spectral-efficiencies. Specifically, the PBS allocates channel c to PU z with the maximum of the priority function at time slot k as follows:

$$z = \underset{m \in \Phi[k]}{\operatorname{argmax}} \frac{R_c^m[k]}{T^m[k]}, \tag{2.23}$$

where $\Phi[k]$ is the set of PUs that require data transmission at time slot k, $R_c^m[k]$ is the instantaneous rate of PU m provided by channel c at time slot k, and $T^m[k]$ is the average throughput of PU m up to time slot k. Therefore, a balance between the maximum throughput and the average throughput can be achieved. The time-scale of scheduling is done every 1 millisecond. The transmit buffer of a PU is updated if the PU is allowed to conduct data transmission by the PBS, where the transmitted data size of the PU depends on the number of allocated channels and the spectral-efficiency of the wireless link. Once the transmit buffer of a PU is emptied, PBS will not allocate spectrum resources to this PU. Note that $R_c^m[k]$ is time-variant depending on the spectral-efficiency of PU m at time slot k.

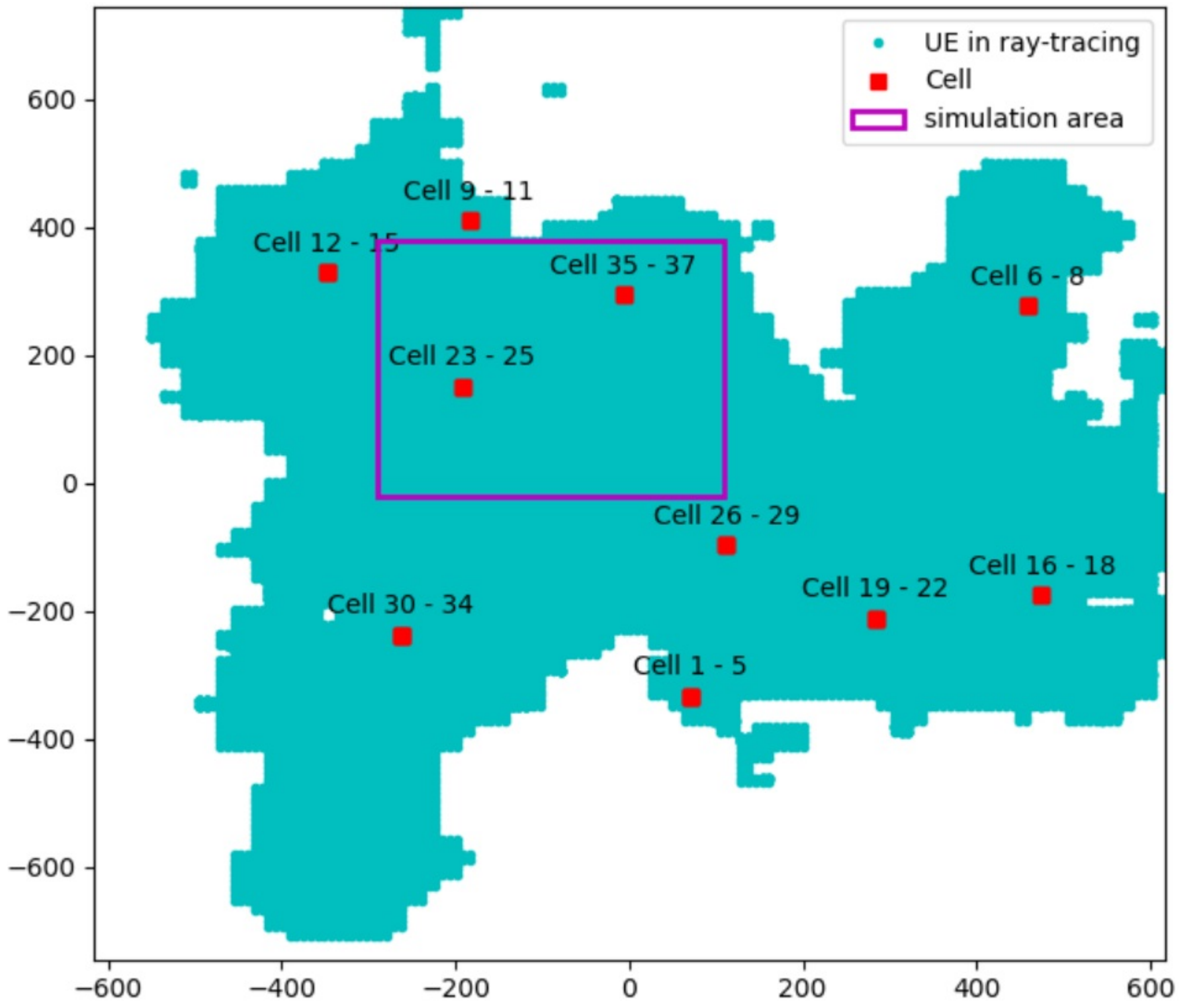


Figure 2.4: Ray-tracing data and simulation area.

2.7.2 Training Details

The SU adopts an online training algorithm that learns its spectrum access strategy continually to adapt to the non-stationary wireless environment. To be specific, the SU collects $T = 500$ training samples in one episode and stores them in its memory buffer. After collecting 500 training samples in one episode, the SU trains its DRQN/DEQN agent for 100 iterations to update its policy and then applies the updated policy in the next episode. Since

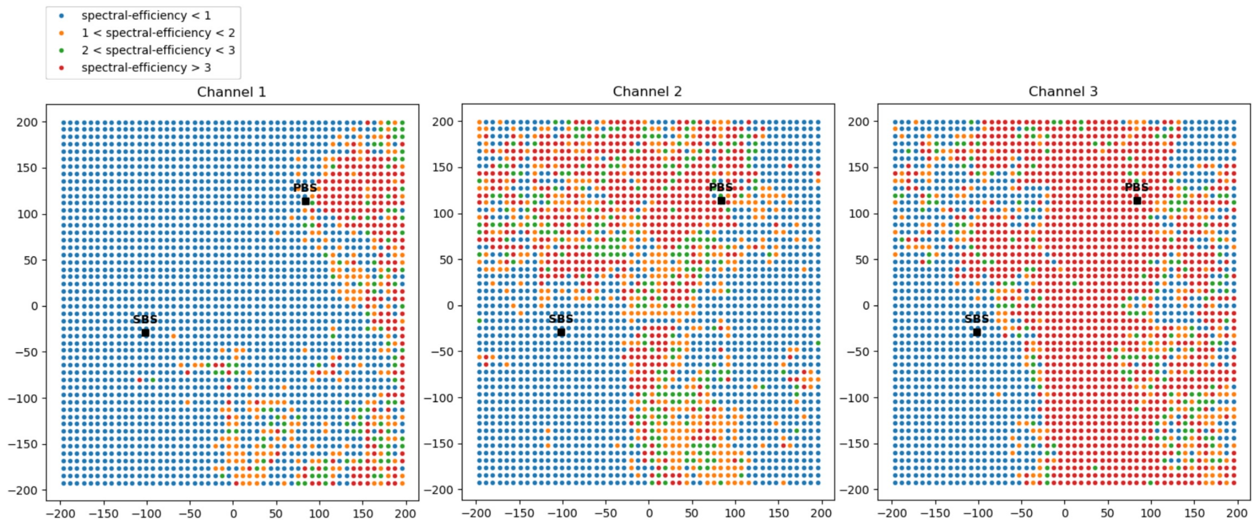


Figure 2.5: Spectral-efficiency of PU on different channels.

the environment gradually changes over time, the SU removes the outdated training samples from the memory buffer and collects another T training samples in the new episode.

Because we aim to evaluate the effect of using ESN as the recurrent Q-network, we compare with DRQN [6] that uses RNN as its recurrent Q-network structure. Although we assume vanilla RNN in theoretical analysis for DRQN to simplify the analysis, Long Short Term Memory (LSTM) is usually preferred to be used in DRQN. Therefore, LSTM is used as the Q-network for DRQN. To have a fair comparison, the LSTM of DRQN and the ESN of DEQN have the same number of neurons $d_h = 64$ and the same learning rate 0.001. Furthermore, we let $B_o = 1$, $\sigma_{\text{in}}^{\max} = 0.5$, $\sigma_{\text{out}}^{\max} = 10$, $\gamma = 0.9$, $R_{\max} = 1$, and $\epsilon = 0.001$. From Corollary 2.8, we calculate the sum of the squared bias and the variance under different $\sigma_{\text{rec}}^{\max}$ in Figure 2.6, and we set $\sigma_{\text{rec}}^{\max}$ to 0.7, which corresponds to the lowest sum. Since the number of channels is 3, the dimensions of input weights, recurrent weights, and output weights are 4×64, 64×64, and 64×6, respectively.

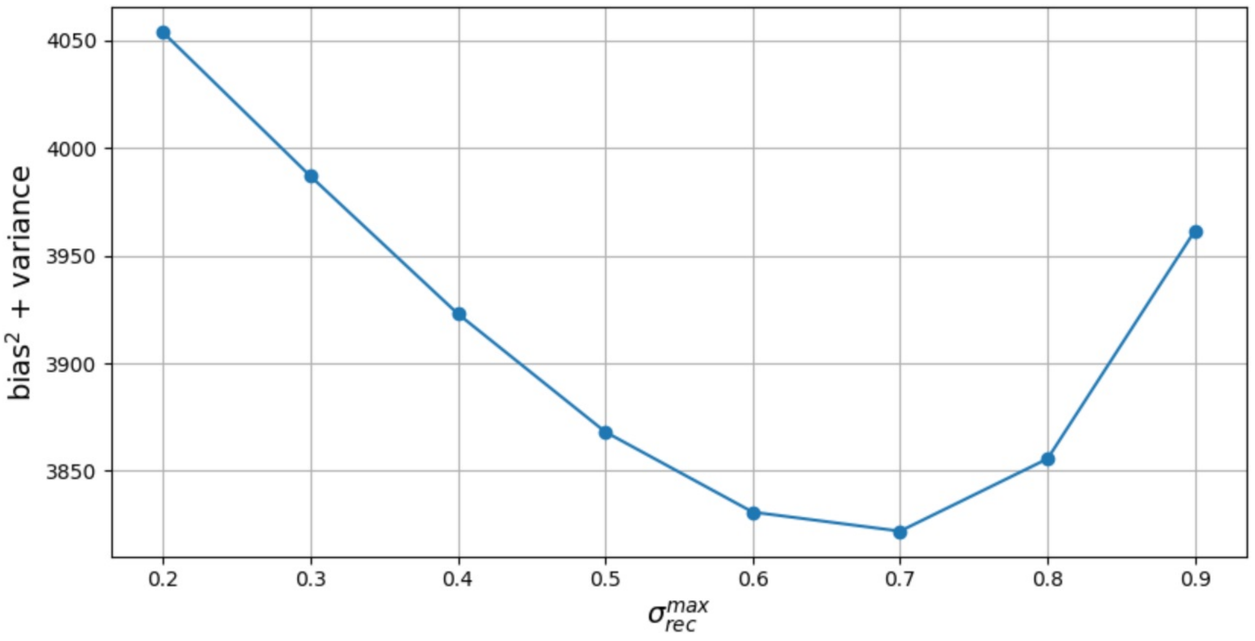


Figure 2.6: The sum of the squared bias and the variance vs. $\sigma_{\text{rec}}^{\text{max}}$.

2.7.3 Results

We run all our experiments with 3 random seeds, which varied the user geometry and the neural network initialization. Each episode consists of 500 periods of spectrum sensing and data transmission, and the results are averaged over these 500 periods and 3 random seeds. The curves of mean reward of different methods are shown in Figure 2.7, and the curves of reward variance of different methods are shown in Figure 2.8. As expected, the training curves of DRQN is extremely unstable and cannot converge well due to insufficient training data and training time. On the other hand, DEQN has a stable training curve and the lowest reward variance. These two reward curves show that DEQN has more stable and better performance than DRQN, which empirically proves that DEQN can learn more efficiently with limited training data than DRQN.

Figure 2.9 (a) and Figure 2.9 (b) show the success probability and failure probability of the SU's access under 3 channels and 4 channels, respectively. The curves represent an average

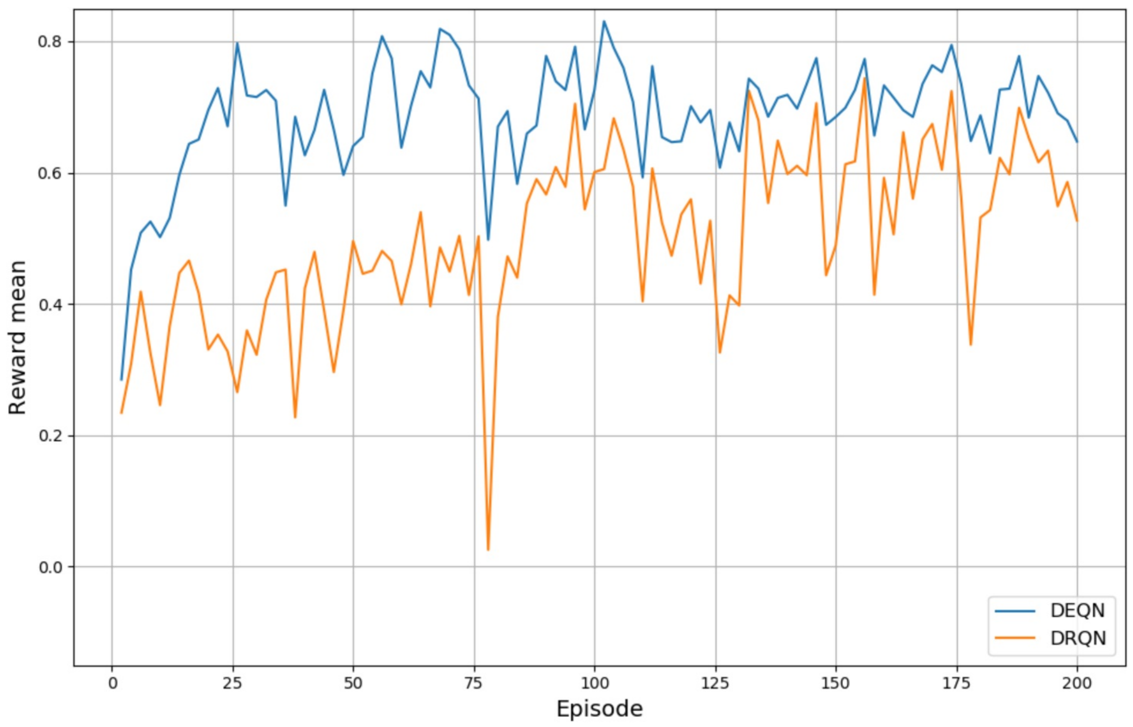


Figure 2.7: Mean of reward vs. episode.

over 3 random seeds, and the shaded areas show the 95% confidence interval. If the SU accesses a channel without generating intolerable interference to the primary system and receives a reward +1, then the SU finds a spectrum opportunity successfully, so we call it as a successful SU access. On the other hand, if the SU generates intolerable interference to the primary system and receives a reward -1. then we call it as a failed SU access. Note that a slight increase of the reward mean corresponds to a significant improvement in the DSS performance. Since the number of available channels decreases, the spectrum opportunities for the SU in 2 channels is less than that in 3 channels. Therefore, we can observe that the success probability of SU's access in 2 channels is lower than that in 3 channels. We observe that DEQN has the lowest failure probability and highest success probability of the SU's access in both scenarios. In sum, Table 2.3 shows the average success/failure probability of SU's access. Therefore, DEQN can find the most spectrum opportunities for the SU.

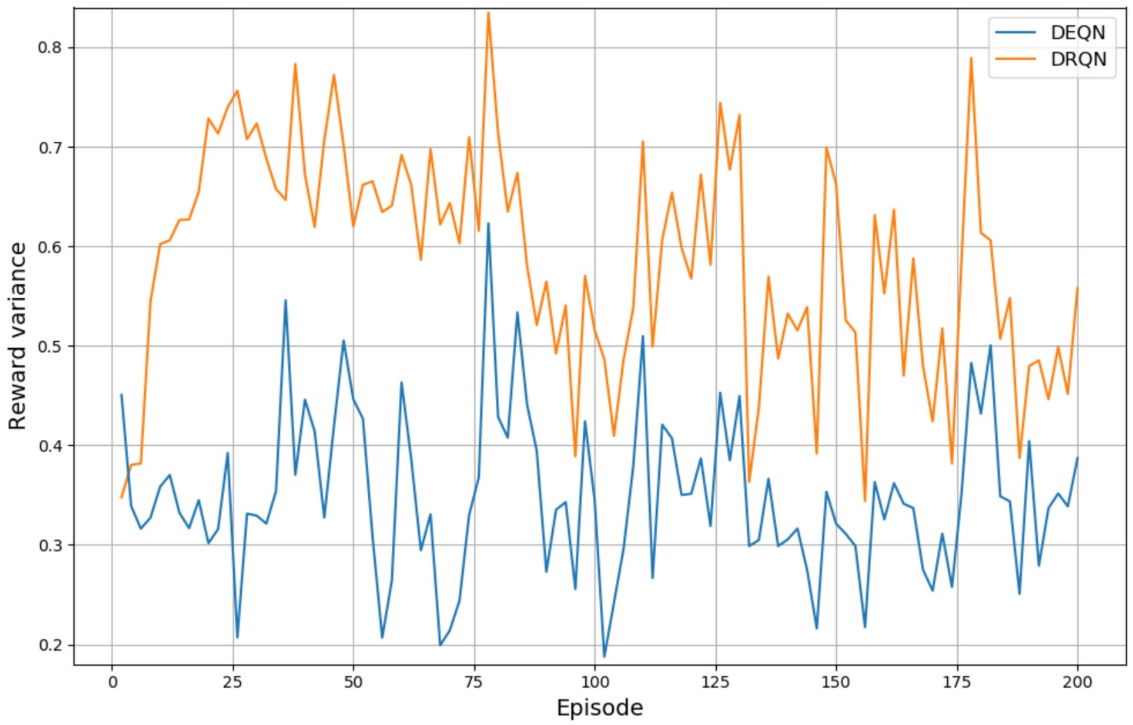


Figure 2.8: Variance of reward vs. episode.

Table 2.3: The average success/failure probability of SU's access.

Probability	DRQN	DEQN
Success (2 channels)	0.68	0.70
Failure (2 channels)	0.15	0.07
Success (3 channels)	0.68	0.76
Failure (3 channels)	0.18	0.07

Table 2.4 compares the average training time for 200 episodes when implemented and executed on the same machine with 2.71 GHz Intel i5 CPU and 12 GB RAM. The training time for DRQN is about 2.52 times more than the training time for DEQN. This huge difference shows the training speed advantage of our introduced DEQN method against the DRQN method. DRQN suffers from high training time because BPTT unfolds the network in time to compute the hidden states and gradients, while DEQN can be trained very efficiently because the hidden states can be pre-stored and only output weights are updated. Since all methods have the same number of training samples and training iterations, the evaluation

Table 2.4: Average Training Time

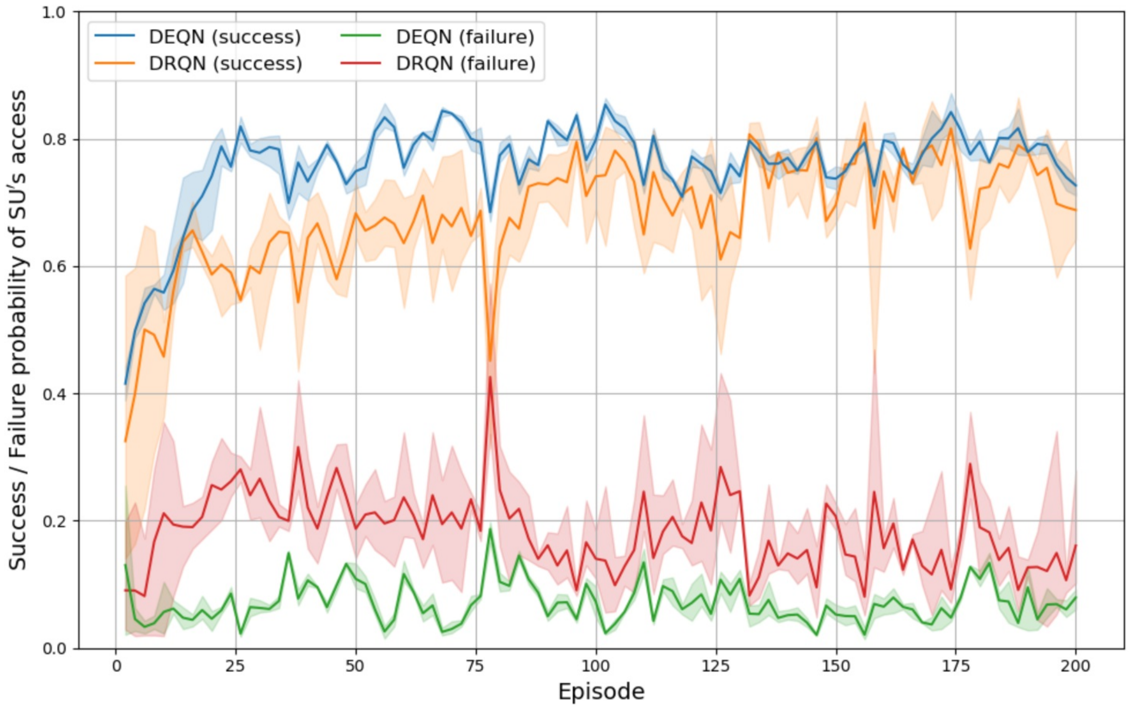
DEQN	DRQN
302.4sec	762.5sec

results confirm the theoretical findings and clearly demonstrate that DEQN has the benefit of faster learning under limited training data than DRQN in practice.

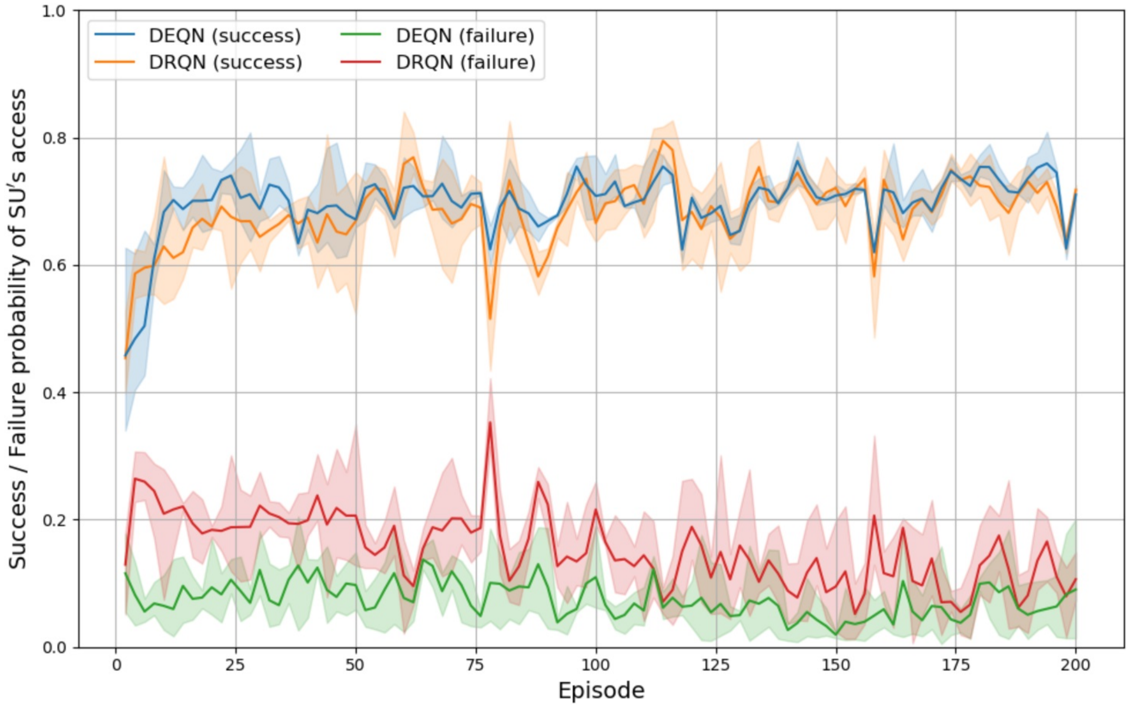
Figure 2.10 shows the reward mean curves of DEQN under different $\sigma_{\text{rec}}^{\max}$. We can observe that $\sigma_{\text{rec}}^{\max} = 0.7$ achieves the best performance, which confirms the numerical evaluation of the sum of the squared bias and the variance in Figure 2.6

2.8 Conclusion

In this work, we introduced a new RNN-based DRL strategy, DEQN, that provides a fast convergence rate to deal with partial observable and non-stationary environments under limited training data. An online learning algorithm is designed for DEQN with significantly reduced training overheads compared to DRQN. Upper-bounds and lower-bounds of exterior ϵ-covering numbers of RNN and ESN are analytically characterized to demonstrate the faster convergence rate of DEQN. Furthermore, we characterize the bias-variance tradeoff of DEQN to determine the spectrum norm constraint of generating ESN's recurrent weights. Experimental results in the DSS environment are obtained to validate our analytical claims. Both analytical and experimental results demonstrate the benefits of DEQN as an efficient DRL framework.



(a) 3 channels.



(b) 2 channels.

Figure 2.9: Success/Failure probability of SU's access vs. episode.

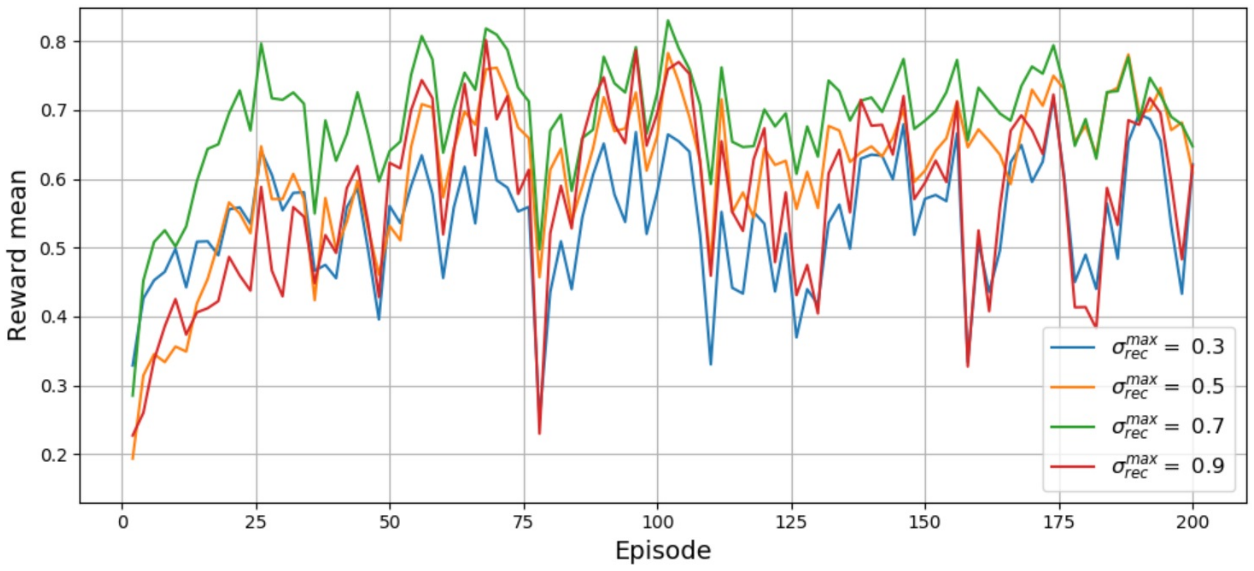


Figure 2.10: Reward mean under different $\sigma_{\text{rec}}^{\text{max}}$ vs. episode.

Chapter 3

Multi-user Dynamic Spectrum Sharing via ESN-based RL

3.1 Spectrum Multiple Access

When multiple SUs access the same group of wireless channels, a good multiple access strategy is needed to avoid collisions and contention among SUs. Multiple access techniques aims to utilize the spectrum resources more efficiently. There are mainly two approaches of spectrum multiple access techniques: random access approaches and channel partitioning approaches. Random access approaches include carrier sense multiple access with collision avoidance (CSMA/CA) and carrier sense multiple access with collision detection (CSMA/CD). The CSMA/CA protocol used by WiFi has two important rules to avoid collisions with other users: (1) Listen before talk. (2) Wait a random time period after colliding with other user. It is important to note that CSMA/CA does not detect collision during transmission. On the other hand, CSMA/CD stops transmission immediately when collision is detected during transmission. CSMA/CD is adopted by Ethernet, which is the dominant wired local area networks (LAN) technology in the world. Since WiFi adopts CSMA/CD multiple access approach, a WiFi user cannot terminate data transmission on the fly. The main reasons that WiFi adopts CSMA/CA are: (1) The strength of the received signal is much smaller than the strength of the transmitted signal, so it is costly to build hardware that can detect and send simultaneously. (2) The hidden terminal problem and fading make the WiFi users cannot detect all collisions. These two problems are resulted from the characteristic of wireless channels, and Ethernet does not have the same problems. Since there is no collision detection for WiFi users, WiFi has the drawback that the resource cannot be allocated efficiently and collisions may happen frequently, which degrades the performance significantly when there are many WiFi users.

Cellular networks adopt channel partitioning approach to achieve spectrum multiple access. Using the fact that the radio signal attenuates when it propagates in space, the available frequency spectrum for cellular networks is reused throughout the service area. To be specific, the service area of cellular networks is divided into cells, and a group of cells forms a cluster. The available frequency spectrum is reused in every cluster, and each cell has a base station that is responsible to control a portion of available frequency bands. The inverse of the number of cells in a cluster is defined as frequency reuse factor (FRF), which indicates how frequently cellular system uses a given amount of spectrum. The multiple access techniques used in commercial cellular networks are channel partitioning approaches, including frequency-division multiple access (FDMA), time-division multiple access (TDMA), code-division multiple access (CDMA), wideband code-division multiple access (WCDMA), and orthogonal frequency-division multiple access (OFDMA). 1G standards only support analog voice call and adopt FDMA for multiple users. 2G standards adopt TDMA+FDMA for Global System for Mobile communication (GSM) system, the most dominant cellular standard in the world, and adopt CDMA for IS-95 system, which is the first CDMA system. 3G standards use CDMA or WCDMA to support increasing data services demand, and 4G standards adopt OFDMA to provide more flexible frequency resource allocation for different users. In sum, these multiple access methods in cellular networks are based on scheduled allocation, which provides exclusive service to licensed users. If the traffic of licensed users is low or only a few licensed users locate in a cell, then some spectrum resources are unused and wasted.

We investigate artificial intelligence-enabled spectrum multiple access strategies in the DSS network. To reduce the control overhead of the underlying DSS network, we incorporate the powerful DRL technique for SUs to learn “appropriate” spectrum access strategies in a distributed fashion assuming NO knowledge of the underlying system statistics. DRL is a suitable framework for developing spectrum multiple access strategies because it is able to adapt the unknown environment without requiring labeled training data. DRL usually requires tons of training data and long training time to converge. However, obtaining training data from the wireless environment is costly in wireless networks because it may impose control overheads to obtain or exchange control information in wireless systems. Furthermore, wireless networks are dynamic due to factors such as path loss, shadow fading, and multipath fading [30], which largely decreases the number of effective training data that reflect the latest environment. Therefore, the major challenge of designing DRL-based spectrum multiple access strategies in the DSS network is how to optimize the system performance under limited control information exchange between the secondary system and the primary system. The performance of spectrum sharing depends on access strategies of multiple users. If one user changes its access strategy, then other users have to change their access strategies accordingly. As a result, designing an efficient DRL framework only requiring a small amount of training data will be critical for 5G and future 6G DSS networks. In this work, we utilize DEQNs to realize DRL by taking advantage of the underlying temporal correlation of the DSS network. To be specific, DEQN-based spectrum multiple access scheme is developed to facilitate DSS systems to perform appropriate channel access, aiming at protecting primary users from harmful interference and avoid collisions with other SUs. DEQN is utilized to learn a spectrum access strategy for each SU in a distributed fashion with limited training data and short training time in the highly dynamic DSS networks.

3.2 System Model

In this section, we introduce the DSS problem with multiple SUs. We consider a DSS system where the primary network consisting of M PUs and the secondary network consisting of N SUs. It is assumed that one wireless channel is allocated to each PU individually and cross-channel interference is negligible. We consider a discrete time model, where the dynamics of the DSS system, such as behaviors of users and changes of the wireless environment, are constrained to happen at discrete time slots t (t is a natural number). Our goal is to develop a distributive DSS strategy for each SU to increase the spectrum utilization without harming the primary network's performance.

The data of an user are transmitted over the wireless link between its transmitter and receiver. Signal-to-interference-plus-noise ratio (SINR) is a quality measure of the wireless connection that compares the power of a desired signal to the sum of the interference power and the power of background noise. The higher value of the SINR, the better quality of the wireless connection. The SINR of the user k's wireless connection on channel m at time slot t is written as

$$\mathrm{SINR}_m^k[t] = \frac{P^k \cdot \left|H^k[t]\right|^2}{\sum\limits_{z \in \Phi_m^k} P^z \cdot \left|H^{zk}[t]\right|^2 + N_m} \tag{3.1}$$

where P^k and P^z are the transmit power of the user k and the user z, respectively, Φ_m^k is the set containing all the users that are transmitting on channel m except for the user k, $H^k[t]$ is the channel gain of the desired link of the user k, $H^{zk}[t]$ is the channel gain of the interference link between the user z's transmitter and the user k's receiver, and N_m is the background noise power on channel m. Note that all channel gains are changing over time so SINR is

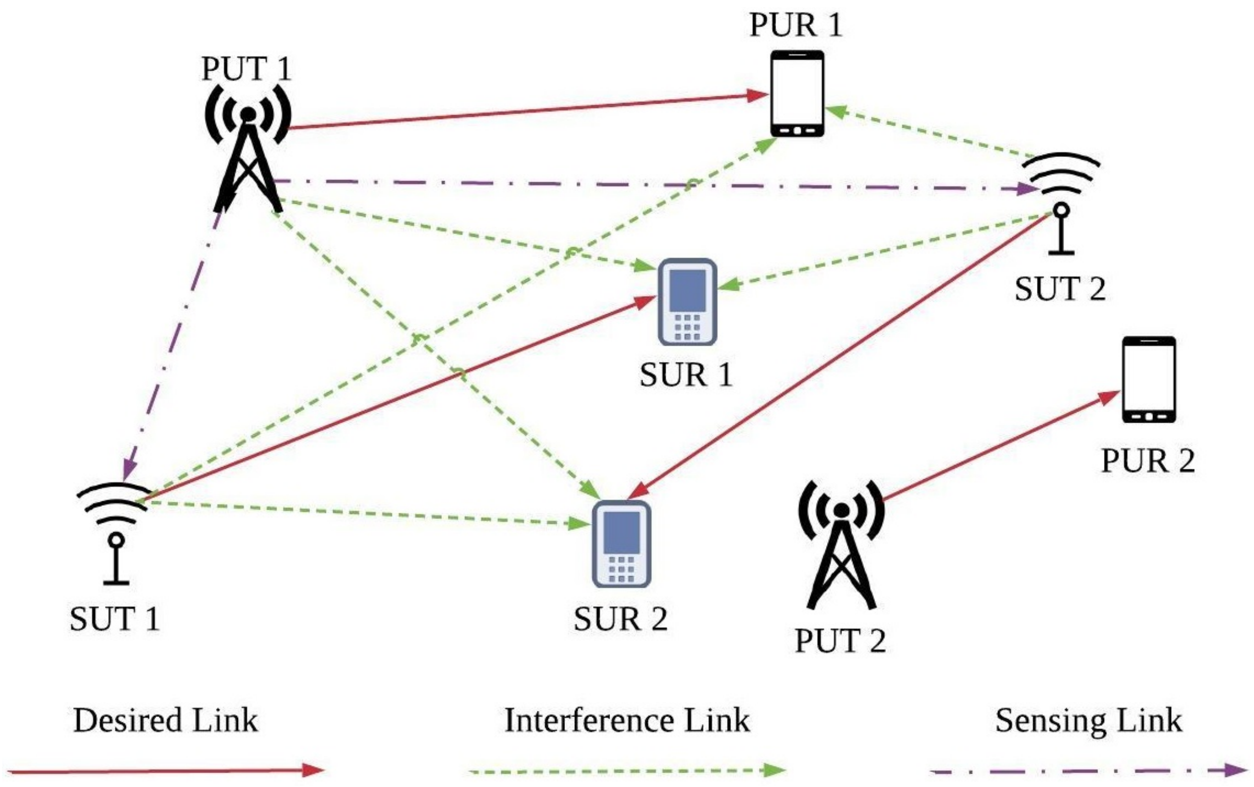


Figure 3.1: The desired links, the interference links, and the sensing links when PU1, SU1, and SU2 are operating on the same channel. PUT/SUT represent the transmitters of PU/SU and PUR/SUR represent the receivers of PU/SU.

also time-variant. The desired link is the link between the transmitter and the receiver of the same user. The interference link is the link between the transmitter and the receiver of two different users if these two users are transmitting on the same channel simultaneously. Figure 3.1 shows the complicated association of desired links and interference links when PU1, SU1, and SU2 are operating on the same channel. Since cross-channel interference is negligible, the interference link between two users operating on different channels is out of consideration.

The radio signal attenuates as it propagates through space between the transmitter and the receiver, which is referred to as the path loss. In addition to the path loss, the channel gain is affected by many factors such as shadow fading and multi-path fading. Shadow fading is caused by a large obstacle like a hill or a building obscuring the main signal path between the transmitter and the receiver. Multi-path fading occurs in any environment where multiple propagation paths exist between the transmitter and the receiver, which may be caused by reflection, diffraction, or scattering. In telecommunication society, the channel model is carefully designed to be consistent with wireless field measurements. We generate channel gains based on the WINNER II channel model [59], which is widely used in industry to make fair comparisons of telecommunication algorithms.

To enable the protection of the primary network, we assume that a PU will broadcast a warning signal if its data transmission experiences a low SINR. There are two possible causes for low SINR. First, the wireless connection of the desired link of the PU is in deep fade, which means the channel gain of the desired link is low. This leads to a small value of the numerator in Equation (3.1) so SINR is low. Second, the signals from one or more SUs cause strong interference to a PU when they are transmitting over the same wireless channel at the same time. This leads to a large value of the denominator in Equation (3.1), so SINR assumes a low value again. We called SUs ”collides” with the PU in this case. The warning signal contains information related to which PU may be interfered so that the SUs transmitting on the same channel are aware of the issue. In fact, this kind of warning signal is similar to the control signals (e.g. synchronization, downlink/uplink control) used in current 4G and 5G networks. It is common to assume that the control signals are received perfectly at receivers, otherwise the underlying network will not even work. In reality, the control signal can be transmitted through a dedicated control channel. According to this mechanism, a PU will broadcast a warning signal once the received SINR is low, and this is the only control information from the primary system to the secondary system to enable the protection for PUs under DSS. Note that a PU may send a warning signal even when no collisions happen because of deep fade.

The activity of a PU consists of two states: (1) *Active* and (2) *Inactive*. If a PU is transmitting data, it is in *Active* state, otherwise it is in *Inactive* state. A spectrum opportunity on a channel occurs when the licensed PU of that channel is in *Inactive* state or any SU can transmit on that channel with little interference to the *Active* licensed PU. Unfortunately, it is difficult for a SU to obtain the information of activity states of PUs or the interference that it will cause in the highly dynamic wireless networks. A SU has to perform spectrum sensing to detect the activity of a PU, but the accuracy of detection is based on the wireless link between the transmitters of the PU and the SU, the background noise, and the transmit power of the PU. On the other hand, the interference level caused by a SU is determined by the interference link from the SU to the PU, the desired link of the PU, transmit powers of the PU and the SU, and the background noise. Furthermore, all these factors for determining spectrum opportunities are time-variant so control information becomes outdated quickly. Since obtaining control information is costly in mobile wireless networks, it is impractical to design a DSS strategy by assuming that all the control information is known.

SUs should provide protection to prevent PUs from harmful interference since the primary system is the spectrum licensee. A commonly used method is that the transmitter of a SU performs spectrum sensing to detect the activity of a PU before accessing a channel. Due to the power and complexity constraints, a SU is unable to perform spectrum sensing across all channels simultaneously. Therefore, we assume that a SU can only sense one channel at a particular time. We adopt the energy detector as the underlying spectrum sensing method, which is the most common one due to its low complexity and cost. The energy detector of SU n first computes the energy of received signals on channel m as follows:

$$E_m^n[t] = \sum_{t'=t}^{t+T_s-1} \left| y_m^n[t'] \right|^2 \tag{3.2}$$

where t is the starting time slot of the spectrum sensing, $y_m^n[t']$ is the received signal at time slot t', and T_s is the number of time slots of the spectrum sensing. We consider the half-duplex SU system where a SU cannot transmit data and perform spectrum sensing at the same time. We assume a periodic time structure of spectrum sensing and data transmission as shown in Figure 3.2. To be specific, the k^{th} sensing and transmission period contains T time slots from $kT+1$ to $(k+1)T$, the spectrum sensing contains the first T_s time slots in the period from $kT+1$ to $kT+T_s$, and the data transmission contains the subsequent $T-T_s$ time slots in the period from $kT+T_s+1$ to $(k+1)T$.

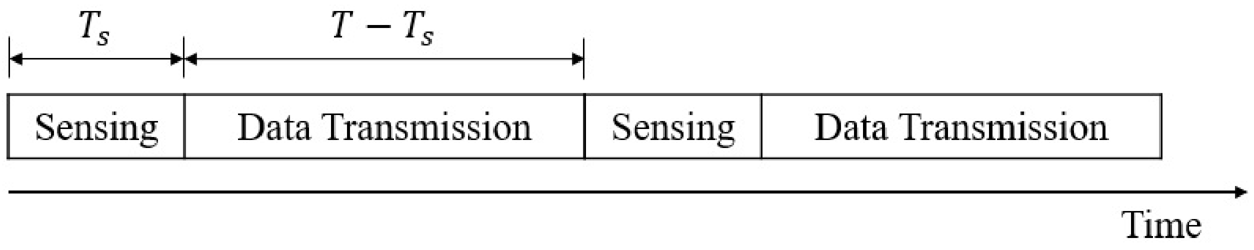


Figure 3.2: The time structure of spectrum sensing and data transmission.

The received signal $y_m^n[t']$ depends on the activity state of PU m, the power of PU m, the background noise, and the sensing link between the transmitters of PU m and SU n. When PU m is in the *Inactive* state, the received signal is represented as

$$y_m^n[t'] = \omega_m[t'] \tag{3.3}$$

When PU m is in the *Active* state, the received signal is represented as

$$y_m^n[t'] = \sqrt{P^m} \cdot H^{mn}[t'] + \omega_m[t'] \tag{3.4}$$

where $\omega_m[t'] \sim \mathcal{CN}(0, N_m)$ is a circularly-symmetric Gaussian noise with zero mean and variance N_m, P^m is the transmit power of PU m, and $H^{mn}[t]$ is the channel gain of the sensing link between the transmitters of PU m and SU n.

If the energy computed in Equation (3.2) is higher than a threshold, the PU is considered in the *Active* state, otherwise the PU is considered in the *Inactive* state. The challenge of designing an energy detector is how to set the threshold properly. The value of the threshold is actually a trade-off between the detection probability and the false alarm probability. However, setting the threshold for achieving a good trade-off is related to many factors, including the channel gain of the sensing link, the transmit power of the PU, the noise variance, the number of received signals, etc. This information is difficult to obtain before deploying in the real environment and is time-variant. Furthermore, setting a threshold is difficult in some cases because of the relative positions of transmitters and receivers. As shown in Figure 3.1, the sensing link is between the transmitters of the PU and the SU, but the interference link is between the transmitter of the SU and the receiver of the PU. The discrepancy between the sensing link and the interference link may cause the hidden node problem, where the sensing link is weak but the interference link is strong. For example, the transmitters of a SU and a PU are far away from each other while the SU transmitter is close to the receiver of the PU. In this case, the transmitters of the SU and the PU are hidden nodes with respect to each other. For example, a SU's transmitter is far away from a PU's transmitter but is close to a PU's receiver, then this SU may not detect the existence of this PU but still causes strong interference to this PU. These SU and PU are said to be hidden nodes with respect to each other in this case. On the other hand, a SU should access the channel more aggressively if the interference link is weak. The warning signals from PUs are designed to provide additional protection to the primary system for the case where the SU cannot detect the activity of the PU, thereby mitigating the issues caused by the hidden nodes. Meanwhile, instead of making the spectrum access decision solely based on the outcomes of the energy detector, we developed a DRL framework to construct a novel spectrum access policy: The DRL agent will use the sensed energy as the input to learn a spectrum access strategy to maximize the cumulative reward. The reward is designed to maximize the spectral-efficiencies of SUs while enabling the protection for PUs with the help of warning signals from PUs.

3.3 DRL Problem Formulation

We now formulate the multi-user DSS problem using the DRL framework, where all SUs in the secondary system learn their spectrum access strategies in a distributed fashion through the interactions with the mobile wireless environment. To be specific, we assume that each SU has a DRL agent that takes its observed state as the input and learns how to perform spectrum sensing and access actions in order to maximize its cumulative reward. The reward for each SU is designed to maximize its spectrum efficiency and to prevent harmful interference to PUs.

The observation of SU n in the k^{th} sensing and transmission period is denoted by

$$s^n[k] = (E^n[k], Q^n[k]), \tag{3.5}$$

where k is a non-negative integer, $E^n[k]$ is the energy of received signals, and $Q^n[k]$ is a one-hot M-dimensional vector indicating the sensed channel from time slots $kT+1$ to $kT+T_s$. If the index of the sensed channel is m, then the m^{th} element of $Q^n[k]$ is equal to one while other elements of $Q^n[k]$ are zeros. On the other hand, $E^n[k]$ is equal to $E^n_m[kT]$ that is calculated by Equation (3.2).

The action of SU n in the k^{th} sensing and transmission period is denoted by

$$a^n[k] = (q^n[k], z^n[k]), \tag{3.6}$$

where $q^n[k] \in \{0, 1\}$ represents SU n will either access the current sensed channel ($q^n[k] = 1$) or be idle ($q^n[k] = 0$) during the data transmission part of the k^{th} period (from time slots $kT + T_s + 1$ to $(k + 1)T$), $z^n[k] \in \{1, ..., M\}$ represents SU n will sense channel $z^n[k]$ during the sensing part of the $(k + 1)^{\text{th}}$ period (from time slots $(k + 1)T + 1$ to $(k + 1)T + T_s$). In other words, SU n makes two decisions: $q^n[k]$ decides whether to conduct data transmission in the current sensed channel of the k^{th} period and $z^n[k]$ decides which channel to sense in the $(k + 1)^{\text{th}}$ period. Therefore, the dimension of each SU's action space is $2M$. Note that the sensed channel in the k^{th} period may be different from that in the $(k + 1)^{\text{th}}$ period

We use a discrete reward function which is similar to the reward function design in Section 2.4. To be specific, each user adopts Table 2.1 to map the received SINR to the achieved spectral-efficiency. Then the spectral-efficiency is utilized as the reward function design for the multi-user DSS system. To jointly consider the performance of the primary and the secondary systems, the reward function corresponding to SU n accessing channel m depends on both the spectral-efficiency of SU n and PU m. During time slots $kT + T_s + 1$ to $(k + 1)T$, the average spectral-efficiency of SU n, $\bar{e}^n[k]$, and the average spectral-efficiency of PU m, $\bar{e}^m[k]$, are calculated by

$$\begin{aligned} \bar{e}^n[k] &= \frac{1}{T - T_s} \sum_{t'=kT+T_s}^{(k+1)T-1} e_m^n[t'] \\ \bar{e}^m[k] &= \frac{1}{T - T_s} \sum_{t'=kT+T_s}^{(k+1)T-1} e_m^m[t'] \end{aligned} \tag{3.7}$$

where $e_m^n[t']$ and $e_m^m[t']$ represent the spectral-efficiency of SU n and PU m on channel m at time slot t', respectively.

The reward of SU n in the k^{th} transmission period is defined as

$$r^n[k] = \begin{cases} -2, & \text{if } \bar{e}^m[k] < 1.5 \\ -1, & \text{if SU } n \text{ is idle in the } k^{\text{th}} \text{ period} \\ 0, & \text{if } \bar{e}^m[k] \geq 1.5 \text{ and } \bar{e}^n[k] < 1 \\ 1, & \text{if } \bar{e}^m[k] \geq 1.5 \text{ and } 1 \leq \bar{e}^n[k] < 2 \\ 2, & \text{if } \bar{e}^m[k] \geq 1.5 \text{ and } 2 \leq \bar{e}^n[k] < 3 \\ 3, & \text{if } \bar{e}^m[k] \geq 1.5 \text{ and } \bar{e}^n[k] \geq 3 \end{cases} \tag{3.8}$$

To enable the protection for the primary system, PU m will broadcast a warning signal if its average spectral-efficiency is below 1.5, and then the reward received by SU n that accesses channel m is set to -2. To motivate SUs to explore spectrum opportunities, the reward $r^n[k]$ is set to -1 if SU n decides to be idle in the k^{th} transmission period. When PU m does not suffer from strong interference (the average spectral-efficiency of PU m is larger than 1.5), we increase the reward $r^n[k]$ from 0 to 3 as the average spectral-efficiency of SU n increases (see Equation (3.8)). Note that the low spectral-efficiency of a PU or a SU does not necessarily mean collisions because the underlying wireless channels are changing dynamically over time. If the channel gain of the wireless link is small, the spectral-efficiency of the user will be low even if there is no collision. Therefore, the reward function and the warning signal are introduced since it is impossible to detect collisions perfectly in practical wireless environments.

3.4 Independent DEQN

To capture the activity patterns of PUs, which are usually time-dependent, applying DRQNs is a natural choice. Although DQNs are able to learn the temporal correlation by stacking a history of states in the input, the sufficient number of stacked states is unknown because it depends on PUs' behavior patterns. RNNs are a family of neural networks for processing sequential data without specifying the length of temporal correlation.

However, the training of RNNs is known to be difficult that suffers from vanishing and the exploding gradients problems. Furthermore, the required amount of training data for achieving convergence is large in the DRL scheme, since there are no explicit labels to guide the training and the agents have to learn from interacting with its environment. In the wireless environment, the channel gain of a wireless link changes rapidly, which is shown in Figure 3.3. Note that the environment observed by a SU is affected by other SUs' access strategies because of possible collisions between SUs, and all SUs are dynamically adjusting their DSS strategies during their training processes. As a result, in the DSS problem, the duration for a learning environment being stable is short and the available training data is very limited.

The standard training technique for RNNs is to unfold the network in time into a computational graph that has a repetitive structure, which is called backpropagation through time (BPTT). BPTT suffers from the slow convergence rate and needs many training examples. DRQN also requires a large amount of training data because a learning agent finds a good policy by exploring the environment with different potential policies. Unfortunately, in the DSS problem, there are only limited training data for a stable environment due to dynamic channel gains, partial sensing, and the existence of multiple SUs. To address this issue, we use ESNs as the Q-networks in the DRQN framework to rapidly adapt to the environment.

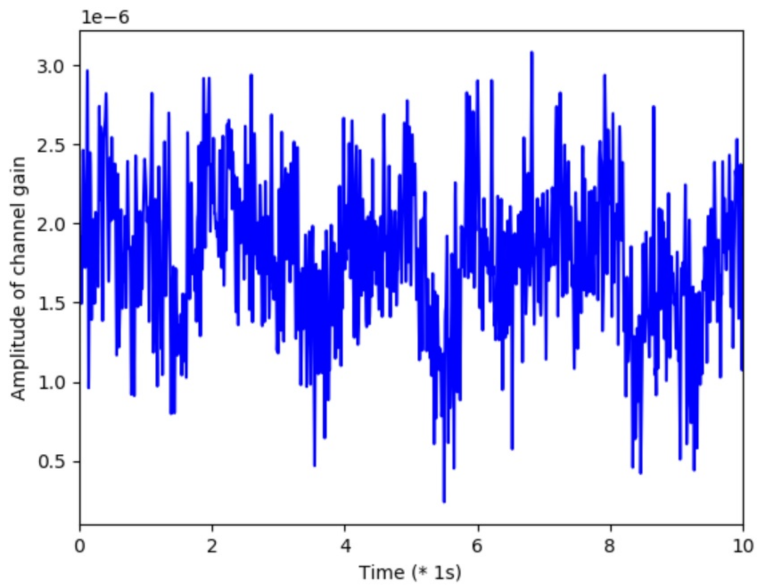


Figure 3.3: Time-variant channel gain of a wireless link.

ESNs simplify the training of RNNs significantly by keeping the input weights and recurrent weights fixed and only training the output weights.

We denote the sequence of observations for SU n by $\{s^n[1], s^n[2], ...\}$. Accordingly, the sequence of hidden states, $\{h^n[1], h^n[2], ...\}$, is updated by

$$\begin{aligned} h^n[k] =&(1-\beta) \cdot h^n[k-1] \\ &+ \beta \cdot \tanh\left(W_{in}^n s^n[k] + W_{rec}^n h^n[k-1]\right), \end{aligned} \tag{3.9}$$

where W_{in}^n is the input weight, W_{rec}^n is the recurrent weight, $\beta \in [0,1]$ is the leaky parameter, and we let $h^n[0] = \mathbf{0}$. The output sequence, $\{o^n[1], o^n[2], ...\}$, is computed by

$$o^n[k] = W_{out}^n u^n[k] \tag{3.10}$$

where $u^n[k]$ is a concatenated vector of $s^n[k]$ and $h^n[k]$, and W_{out}^n is the output weight. Note that the output vector $o^n[k]$ is a $2M$-dimensional vector, where each element of $o^n[k]$ corresponds to the estimated Q-value of selecting one of all possible actions given the observation $s^n[1], ..., s^n[k]$.

The double Q-learning algorithm [60] is adopted to train the underlying DEQN agent of each SU. Each DEQN agent has two Q-networks: the evaluation network and the target network. Let the output sequence from the evaluation network and the target network be $\{o^n_\theta[1], o^n_\theta[2], ...\}$ and $\{o^n_{\theta^-}[1], o^n_{\theta^-}[2], ...\}$, respectively. The loss function for training the evaluation network of SU n is written as

$$\left(r^n[k] + \gamma o^n_{y,\theta^-}[k+1] - o^n_{y,\theta}[k]\right)^2, \tag{3.11}$$

where $o^n_{y,\theta^-}[k+1]$ and $o^n_{y,\theta}[k]$ are the y^{th} element of $o^n_{\theta^-}[k+1]$ and $o^n_\theta[k]$, respectively, y is the index of the maximum element of $o^n_\theta[k+1]$, $r^n[k] + \gamma o^n_{y,\theta^-}[k+1]$ is the target Q-value. To stabilize the training targets, the target network is only periodically synchronized with the evaluation network.

The input weights and the recurrent weights of ESNs are randomly initialized according to the constraints specified by the Echo State Property [10], and then they remain untrained. Only the output weights of ESNs are trained so the training is extremely fast. The main idea of ESNs is to generate a large reservoir that contains the necessary summary of past input sequences for predicting targets. From Equation (3.9), we can observe that the hidden state $h^n[k]$ at any given time slot k is unchanged during the training process if the input weights and recurrent weights are fixed. In contrast to conventional RNNs that usually initialize the hidden states to zeros and waste some training examples to set them to appropriate values in one training iteration, the benefit of ESNs is that the hidden states do not need to be reinitialized in every training iteration. Therefore, the training process becomes extremely efficient, which is especially suitable for learning in a high dynamic environment. Compared to storing $(s[k], a[k], r[k], s[k+1])$ in conventional DRQN framework, we also store hidden states $(h[k], h[k+1])$ because hidden states are unchanged. In this way, we do not have to waste lots of training time and data to recalculate hidden states in every training iteration. It largely boosts the training efficiency in the highly dynamic environment since we can avoid using BPTT and only update the output weights of networks. Furthermore, we can randomly sample from the replay memory to create a training batch, while conventional DRQN methods have to sample continuous sequences to create a training batch. Thus the training data can be more efficiently used in our DEQN method. The training data stored in the buffer will be refreshed periodically in order to adapt to the latest environment. Therefore, our training method is an online training algorithm that keeps updating the learning agent. The training algorithm for DEQNs in the DSS problem is detailed in Algorithm 4.

3.5 Performance Evaluation

3.5.1 Experimental Setup

We set the number of PUs and SUs to 4 and 6, respectively, and the locations of PUs and SUs are randomly defined in a 2000m$\times$2000m area. The distance between the transmitter and the receiver of each desired link is randomly chosen from 400m-450m. Figure 3.4 shows the geometry of the DSS network, where PUT/SUT represent the transmitters of PU/SU and PUR/SUR represent the receivers of PU/SU. The channel gains of desired links, interference links, and sensing links are generated by the WINNER II channel model widely used in 3GPP LTE-Advanced and 5G networks [59]. In this case, there are 4 desired links for PUs, 6 desired links for SUs, 30 interference links between different SUs, 24 interference links between SUTs and PURs, 24 interference links between PUTs and SURs, and 24 sensing links between PUTs

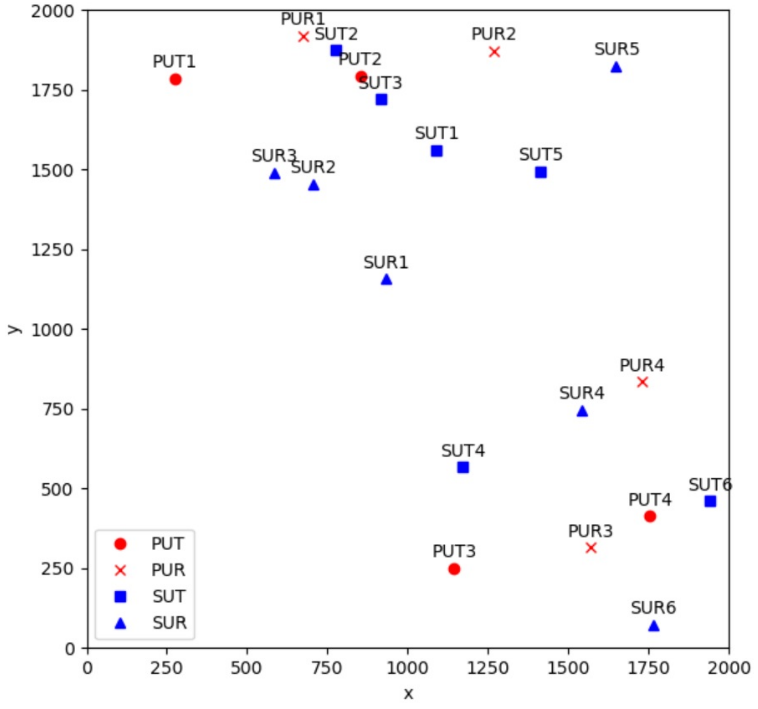


Figure 3.4: The DSS network geometry. PUT/SUT represent the transmitter of PU/SU. PUR/SUR represent the receiver of PU/SU.

and SUTs. Totally, 112 wireless links are generated in our simulation, which establishes a more complicated scenario than existing DRL-based DSS strategies [38, 39, 40]. Specifically, [38] considers each channel only has two possible states (good or bad) without modeling the true wireless environment; [39] assumes that the collision between users can be perfectly detected without considering the dynamics of interference links; [40] assumes that SUs are forbidden to access a channel when a PU is using without considering the actual interference links between PUs and SUs.

For each channel, the bandwidth is set to 5MHz and the variance of the Gaussian noise is set to -157.3dBm. The transmit power of PUs and SUs are both set to 500mW. We set the sensing and transmission period T to 10 time slots and the sensing duration T_s to 2 time slots, where one time slot represents interval of 1ms. For the activity pattern of PUs, we let two PUs be in *Active* state every $3T$ (PU1 and PU3) and two PUs be in *Active* state every $4T$ (PU2 and PU4). We list all the parameters to generate the wireless environment in Table 3.1.

Table 3.1: The values of parameters for generating the wireless environment.

Parameter	**Value**
number of PUs M	4
number of SUs N	6
simulation area	2000m×2000m
distance between user pair	400m-450m
transmit power of PU	500mW
transmit power of SU	500mW
variance of Gaussian noise	-157.3dBm
bandwidth of a channel	5MHz
interval of one time slot	1ms
sensing and transmission period T	10 time slots
sensing duration T_s	2 time slots

3.5.2 Training Details

Each SU trains its DEQN agent and updates the policy accordingly after collecting 300 samples in the buffer. The buffer will be refreshed after training so we only use training data from the latest 3 sec. The total number of training data is 60000, which requires 600 sec to collect all the training data. The initial exploration probability ϵ is set to 0.3, and then it will gradually decrease until ϵ is 0. We first train the Q-network with learning rate 0.01, and then the learning rate decreases to 0.001 when ϵ is less than 0.2.

As shown in Figure 3.5, our DEQN network consists of L reservoirs for extracting the necessary temporal correlation to predict targets. The number of neurons in each reservoir is set to 32 and the leaky parameter β is set to 0.7 in Equation (3.9). During the training process, the input weights $\{W_{in}^{(1)}, ..., W_{in}^{(L)}\}$ and the output weights $\{W_{rec}^{(1)}, ..., W_{rec}^{(L)}\}$ are untrained. To find a good policy, only the output weight W_{out} is trained to read essential temporal

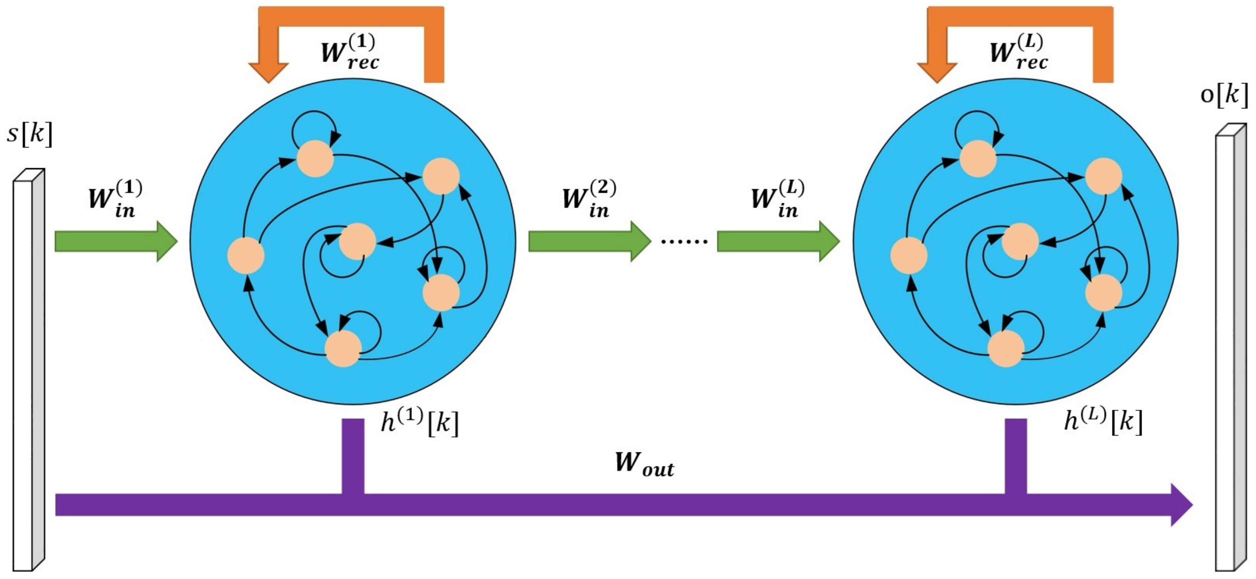


Figure 3.5: The network architecture of DEQN.

information from the input observations and the hidden states stored in the experience replay buffer. Existing research shows that stacking RNNs automatically creates different time scales at different levels, and this stacked architecture has better ability to model long-term dependencies than single layer RNN [61, 62, 63]. We also find that stacking ESNs can indeed improve the performance in our experiment.

3.5.3 Results

We evaluate our introduced DEQN method with three performance metrics: 1) The system throughput of PUs. 2) The system throughput of SUs. 3) The required training time. The throughput represents the number of transmitted bits per second, which is calculated by (spectral-efficiency) $\times$ (bandwidth), and the system throughput represents the sum of users' throughput in the primary system or secondary system. A good DSS strategy should increase the throughput of SUs as much as possible, while the transmissions of SUs do not harm the throughput of PUs. Therefore, each SU has to access an available channel by predicting activities of other mobile users. We compare with conventional DRQN method that uses Long Short Term Memory (LSTM) [64] as the Q-network. For a fair comparison, we also set the number of neurons in each LSTM layer to 32. The training algorithm of DRQNs is BPTT and double Q-learning with the same learning rate as DEQNs. Since each SU updates its policy for every 300 samples, we show all of our curves in figures by calculating the moving average of 300 consecutive samples for clarity.

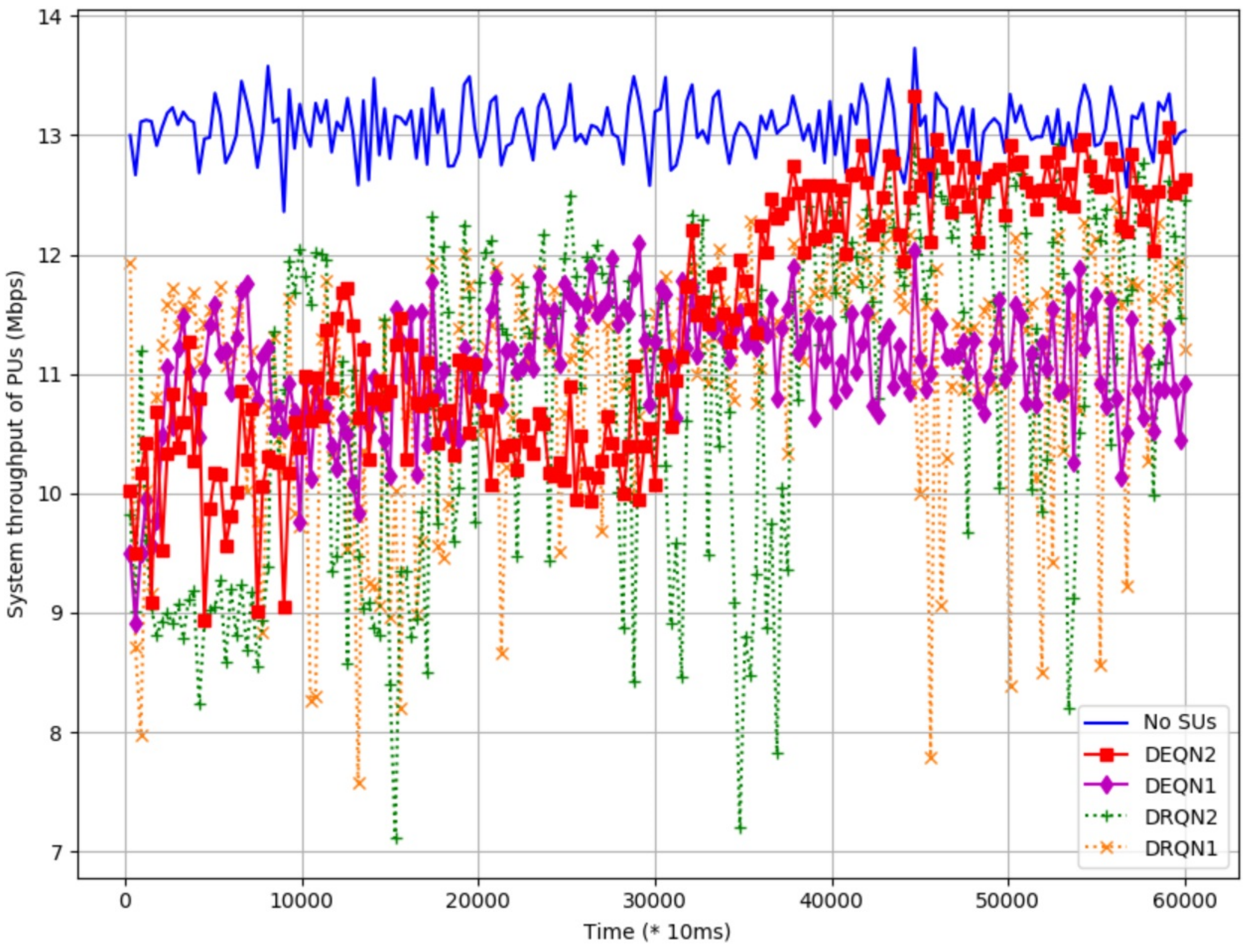


Figure 3.6: The system throughput of PUs.

DEQN1 and DEQN2 are our DEQN method with one and two layers, respectively, and DRQN1 and DRQN2 are the conventional DRQN method with one and two layers, respectively. The system throughput of PUs is shown in Figure 3.6 and the system throughput

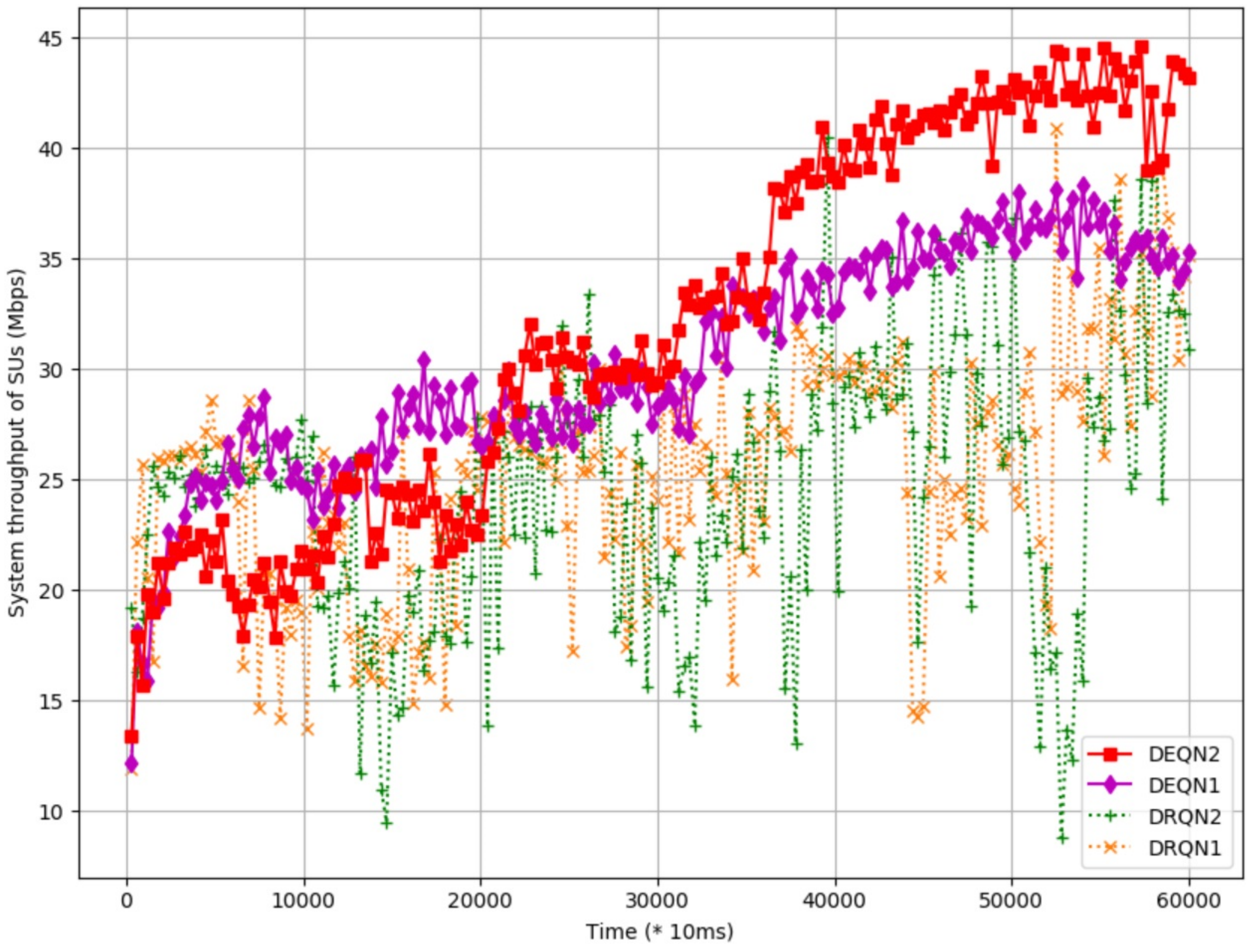


Figure 3.7: The system throughput of SUs.

of SUs is shown in Figure 3.7. We observe that DEQNs have more stable performance than DRQNs, which empirically proves that the DEQN method can learn efficiently with limited training data. Note that one experience replay buffer only contains 300 latest training samples. After updating the learning agent of each SU using the 300 data in the buffer, DSS strategy of each SU changes so the environment observed by one SU also changes. Therefore, we have to erase the outdated samples from the buffer and let SUs collect new training data from the environment. Figure 3.8 shows the average reward of SUs versus time. We observe extremely unstable reward curves of both DRQN1 and DRQN2 so it proves that DRQNs cannot adapt to this dynamic wireless scenario well with few training data.

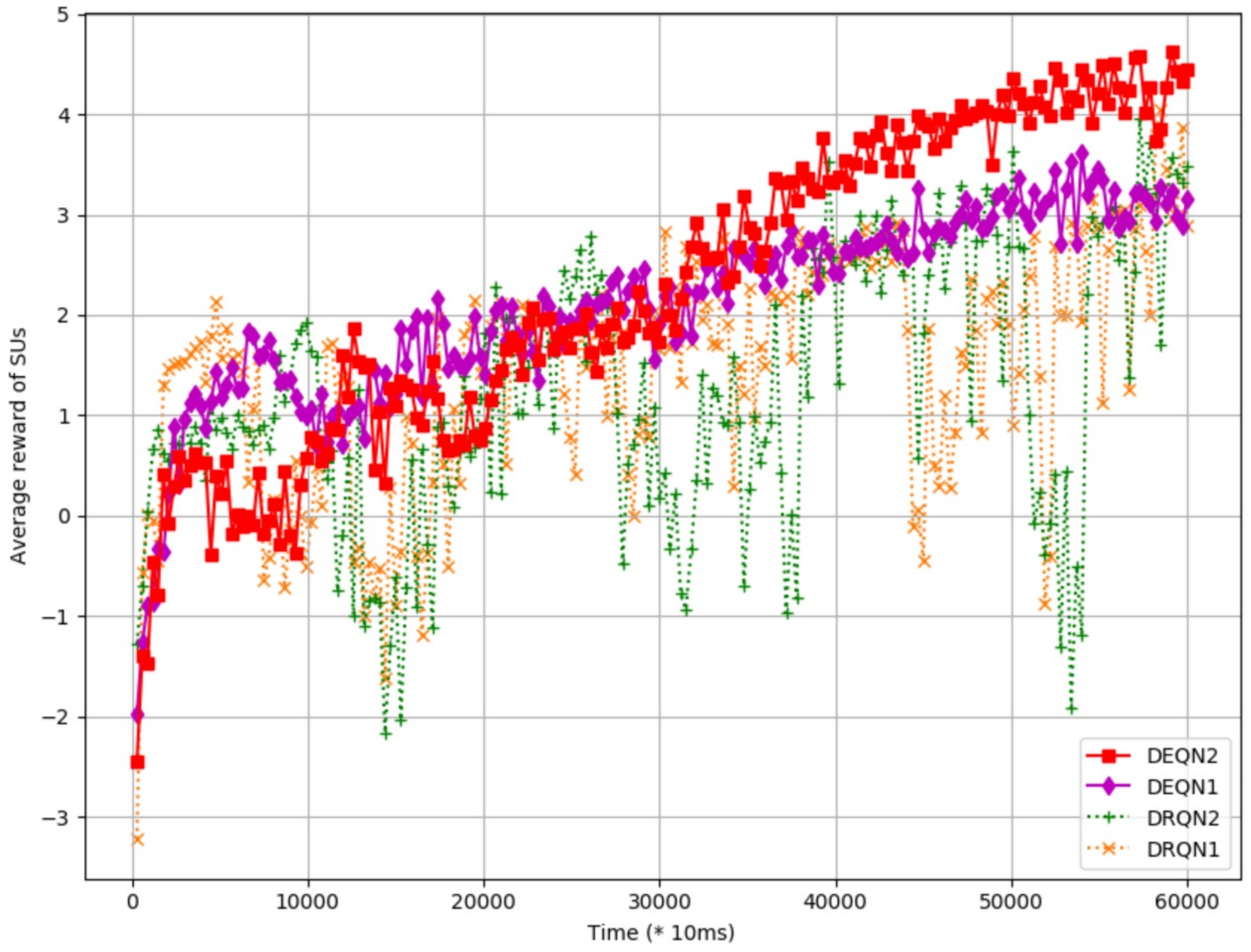


Figure 3.8: The average reward versus time.

We observe that DEQN2 has better performance than DEQN1 in both the system throughput of PUs and SUs, which shows that deep structure (stacking ESNs) indeed improves the capability of the DRL agent to learn long-term temporal correlation. As for DRQNs, we observe that DRQNs do not have improved performance as we increase the number of layers in the underlying RNN. The main reason is that more training data are needed for training a larger network but even DRQN with one layer cannot be trained well.

The top priority of designing a DSS network is to prevent harmful interference to the primary system. To analyze the performance degradation of the primary system after allowing the secondary system to access, we show the system throughput of PUs when there is no SU

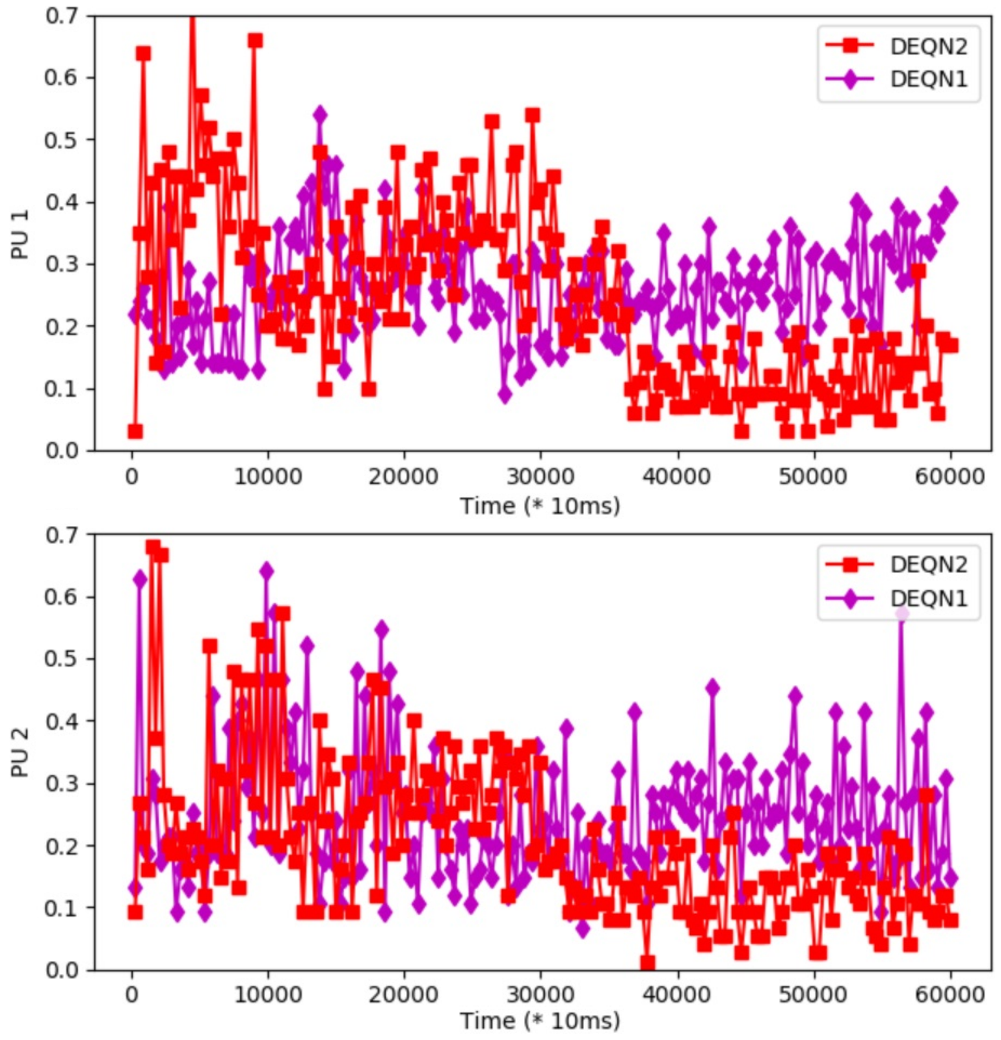


Figure 3.9: The average warning frequency of PU1 and PU2 versus time.

exist in Figure 3.6. We observe that DEQN2 can achieve almost the same performance of the system throughput of PUs. A PU broadcasts a warning signal if its spectral-efficiency is below a threshold. For each PU, we record the frequency of (the PU sends a warning signal and it is received by some SUs) / (number of the PU's access), which is called as the warning frequency. Figure 3.9 and Figure 3.10 show the average warning frequency of each PU versus time. We observe that the every PU decreases its warning frequency over time, meaning that each SU learns not to access the channel that will cause harmful interference

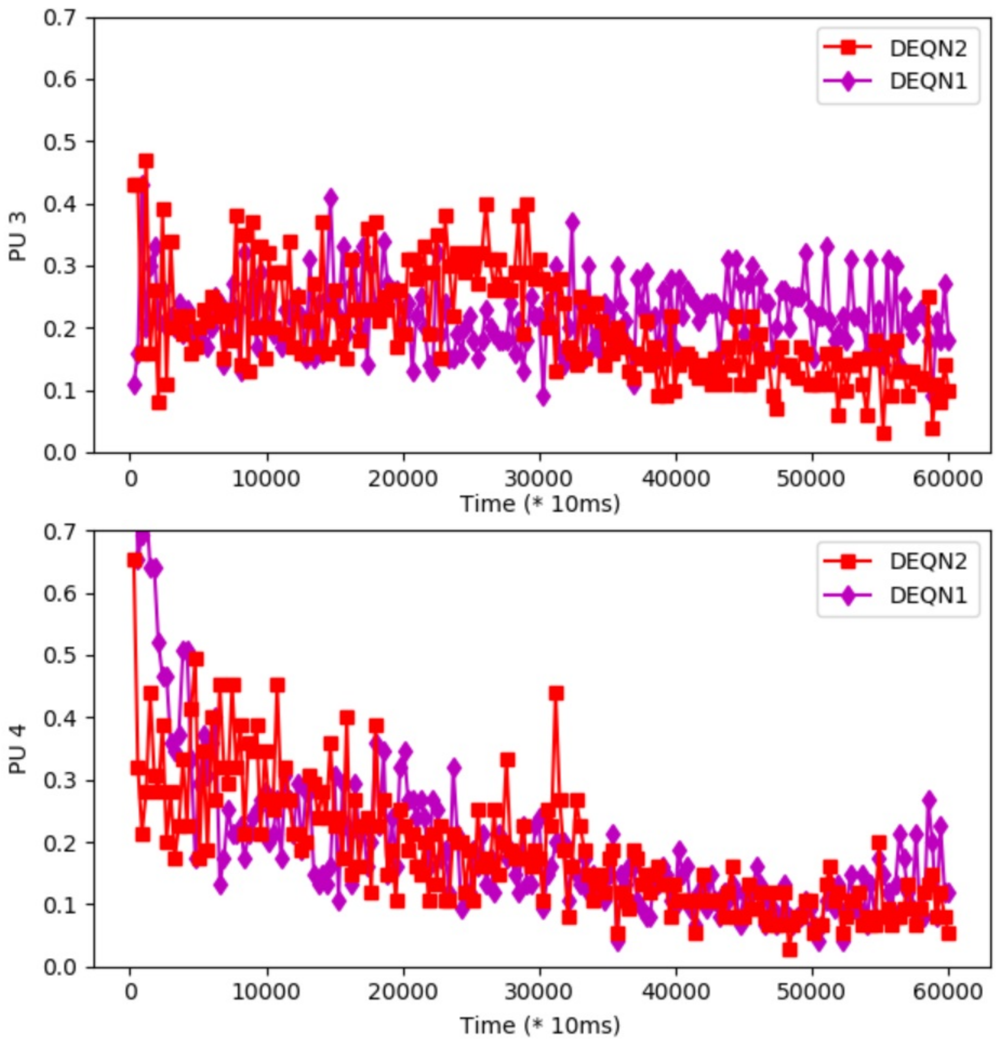


Figure 3.10: The average warning frequency of PU3 and PU4 versus time.

to PUs.

We compare the training time of different approaches in Table 3.2 when implemented and executed on the same machine with 2.71 GHz Intel i5 CPU and 12 GB RAM. The required training time for DRQN1 is 23.4 times the training time for DEQN1, and the required training time for DRQN2 is 42.8 times the training time for DEQN2. This huge difference shows the training speed advantage of our introduced DEQN method against the conventional DRQN method. DRQN suffers from high training time because BPTT unfolds the network

Table 3.2: The comparison of training time of different network architectures.

Network	**Training time (sec)**
DEQN1	161
DEQN2	178
DRQN1	3776
DRQN2	7618

in time to compute the gradients, but DEQN can be trained very efficiently because the hidden states can be pre-stored for many training iterations.

3.6 Conclusion

In this work, we study spectrum multiple access strategies in a distributive DSS network under the condition of imperfect spectrum sensing and no centralized controllers. The independent DEQN learning approach can efficiently capture the temporal correlation of the underlying time-dynamic environment requiring very limited amount of training data. Equipped with the DEQN agent, each SU is able to make proper spectrum access decisions distributively relying only on minimal warning information from the PUs, their own spectrum sensing outcomes, and the learning outcomes. Compared to the DRQN-based approach, the DEQN-based approach largely increase the convergence rate. Experimental results verify the performance of the independent DEQN learning framework, showing significant performance improvements over state-of-the-art DRQN-based approaches. This provides strong evidence for adopting DEQN for real-time and time-dynamic applications.

Algorithm 4 The training algorithm for independent DEQN.

```
Initialize the wireless environment with $M$ PUs and $N$ SUs.
Set the sensing and transmission period to $T$ time slots and the sensing duration to $T_s$ time slots.
Set the buffer size to $Z$, the training iteration to $I$, and the exploration probability to $\epsilon$.
Randomly initialize an evaluation network $\mathrm{DEQN}^n_\theta$ and a target network $\mathrm{DEQN}^n_{\theta^-}$ with the same weights for each SU $n$.
Each SU $n$ randomly selects one channel ($= z^n[0]$) to sense for $T_s$ time slots and then computes the state $s^n[1]$.
for $q = 1, ...$ do
    Initialize an empty buffer $B^n_q$ for each SU.
    for $z = 1, ..., Z$ do
        Let $k = (q-1)Z + z$.
        Each SU $n$ inputs $s^n[k]$ to $\mathrm{DEQN}^n_\theta$, calculates the hidden state $h^n_\theta[k]$, and outputs $o^n_\theta[k]$.
        Each SU $n$ decides action $a^n[k] = (q^n[k], z^n[k])$ based on $\epsilon$-greedy policy, where $a^n[k]$ is the index of the maximum element of $o^n_\theta[k]$ with probability $1-\epsilon$ and $a^n[k]$ is chosen randomly with probability $\epsilon$.
        Each SU $n$ accesses channel $z^n[k-1]$ if $q^n[k] = 1$ or does not access if $q^n[k] = 0$ for $T - T_s$ time slots.
        Each SU $n$ obtains the reward $r^n[k]$.
        Each SU $n$ senses channel $z^n[k]$ for $T_s$ time slots and then computes the state $s^n[k+1]$.

        Each SU $n$ inputs $s^n[k+1]$ to $\mathrm{DEQN}^n_{\theta^-}$, calculates the hidden state $h^n_{\theta^-}[k]$, and outputs $o^n_{\theta^-}[k]$.
        Each SU $n$ stores $(s^n[k], h^n_\theta[k], a^n[k], r^n[k], s^n[k+1], h^n_{\theta^-}[k])$ in $B^n_q$.
    end for
    for iteration $= 1, ..., I$ do
        Each SU $n$ samples random training batch $(s^n[k], h^n_\theta[k], a^n[k], r^n[k], s^n[k+1], h^n_{\theta^-}[k])$ from $B^n_q$.
        Each SU $n$ inputs $s^n[k]$ and $h^n_\theta[k]$ to $\mathrm{DEQN}^n_\theta$ to calculate $o^n_\theta[k]$
        Each SU $n$ inputs $s^n[k+1]$ and $h^n_\theta[k+1]$ to $\mathrm{DEQN}^n_{\theta^-}$ to calculate $o^n_{\theta^-}[k]$
        Each SU $n$ updates $\mathrm{DEQN}^n_\theta$ by performing gradient descent step on

            $$\left(r^n[k] + \gamma o^n_{y,\theta^-}[k+1] - o^n_{y,\theta}[k]\right)^2,$$

        where $y$ is the index of the maximum element of $o^n_\theta[k+1]$.
    end for
    Each SU $n$ synchronizes $\mathrm{DEQN}^n_{\theta^-}$ with $\mathrm{DEQN}^n_\theta$.
end for
```

Chapter 4

Multi-user Dynamic Spectrum Sharing via ESN-based MARL

4.1 Multi-agent Reinforcement Learning (MARL)

We can treat the spectrum multiple access in the DSS network as a problem of multi-agent reinforcement learning (MARL) problem, where multiple RL agents share a common environment. In fact, many of the successful DRL applications such as the games of Go and Poker [3], robotic control [65], and autonomous driving [66], naturally fall into the realm of MARL. MARL is a challenging problem because both the local observation and the local reward received by each agent are influenced by other agents' actions. In other words, an agent not only interacts with the environment but also interacts with each other, resulting in a non-stationary environment from each local agent's viewpoint. To be specific, MARL is characterized by N tuples $(\mathcal{S}^n, \mathcal{A}^n, \mathcal{T}^n, R^n, \Omega^n, \mathcal{O}^n, \gamma)_{n\in N}$, where N is the number of agents, $\mathcal{S}^n$ is the state space of agent n, $\mathcal{A}^n$ is the action space of agent n, Ω^n is the observation space of agent n, and $\gamma \in [0,1]$ is the discount factor. At time t, the state of agent n is $s^n[t] \in \mathcal{S}^n$, the observation of agent n is $o^n[t] \in \Omega^n$, the action of agent n is $a^n[t] \in \mathcal{A}^n$, and the reward of agent n is $r^n[t]$. Note that $\mathcal{T}^n$ is the state transition probability of agent n providing $\Pr\left(s^n[t+1]|s^n[t], a[t]\right)$, $\mathcal{O}^n$ is the observation probability of agent n providing $\Pr\left(o^n[t+1]|s^n[t+1], a[t]\right)$, R^n is the reward function of agent n providing

$r^n[t] = R^n\left(\cdot|s^n[t], a[t]\right)$. In addition, we denote $\mathcal{S} = \cup_{n=1}^{N}\mathcal{S}^n$ as the joint state space, $\mathcal{A} = \cup_{n=1}^{N}\mathcal{A}^n$ as the joint action space, and $R = \sum_{n=1}^{N} R^n$ as the joint reward function. The goal of the considered MARL problem is to optimize the joint reward $r[t]$, which is defined as

$$r[t] = \sum_{n=1}^{N} r^n[t]. \tag{4.1}$$

It is important to note that $r^n[t]$ depends on the joint action, $a[t] = \left(a^1[t], \ldots, a^N[t]\right)$, in the MARL setting.

Most MARL algorithms assume that a joint reward is received by all agents, or each agent receives an individual reward but shares it with other agents. However, this assumption may not be practical in some real-world applications because agents do not share their received observations and rewards for data privacy and security issues. In this paper, we assume that each agent does not share its local observations and rewards with other agents, where each agent updates its policy to maximize its own long-term local reward. We leverage federated learning to jointly learn a shared policy that maximizes the joint reward by combining all agents' local policies.

4.2 Federated Learning

The training of ML algorithms can be centralized or distributed. To ensure the training accuracy, most learning based methods require a centralized server to collect all the training data and perform the centralized training. Since the training data are usually distributed over wireless users in different locations, these wireless users have to send their collected data to the centralized server frequently to maintain the accurate decision making in the highly dynamic 5G network. However, the frequent data exchange between the centralized server and wireless users result in high communication overhead in network operations. Furthermore, the user data sent to the server may contain sensitive information such as user location, which may be eavesdropped by hackers and cause privacy issues.

Therefore, it triggers the idea of the distributed learning framework that does not require users to send their private data to the centralized server. In the entirely distributed learning scheme called independent learning, each wireless user only optimizes its own training task without communicating with one another. Authors in [39] utilizes Aloha based protocol DRL method to solve multi-user spectrum access problem in a independent learning manner, which has a DQN architecture with lower computational complexity. Although the independent learning can maintain the efficiency and scalability when handling a large number of users, the system performance of the independent learning is much worse than the centralized training due to a lack of cooperation among users.

To address the aforementioned problems, we utilize a special distributed learning framework called federated learning (FL). FL enables multiple devices to collaboratively learn a shared model while keeping the training data local. To be specific, each device downloads the shared model from the centralized server and updates the model using its local training data. These locally trained models are then sent to the centralized server, and the centralized server aggregates the information from these models to obtain an updated shared model. Then the updated shared model is downloaded by distributed devices to start next training cycle. In this way, all the training data remains on local devices, so data privacy can be ensured. Furthermore, the communication efficiency can be improved because only model updates are exchanged between the centralized server and local devices. Lastly, FL enables the cooperation among devices via the adequate model aggregation process in the centralized server.

4.3 System Model

The Federal Communications Commission (FCC) has released the 3.5 GHz (3550-3700 MHz) band, termed Citizens Broadband Radio Service (CBRS), for shared spectrum usage of federal and commercial users [67]. To be specific, the CBRS system has been opened for spectrum sharing across three tiers of users: Incumbent User (IU), Priority Access Licenses (PAL) users, and General Authorized Access (GAA) users. IUs include federal users such as military radars and satellite ground stations, which are the highest tiers and should be protected from possible interference from the lower tiers such as PAL or GAA users. The second tier PAL users are commercial users that are protected from the interference caused by GAA users. PALs are licensed based on spectrum auction, and each PAL consists of a 10 MHz channel for a 10-year term. Finally, the lowest tier GAA users must not cause harmful interference to IUs or PAL users and must accept interference from them. Under these constraints, GAA provides free access to the available spectrum, so GAA users act as unlicensed users in the CBRS system. FCC requires that the operations among users in these three tiers should be managed through an automated frequency coordinator called Spectrum Access System (SAS). When managing spectrum access, SAS may incorporate information from an Environmental Sensing Capability (ESC), which is a sensor network that detects the transmissions of IUs. In this work, we focus on designing the spectrum access algorithm for GAA users in the CBRS system.

In this section, we describe the DSS problem in the CBRS system, which is shown in Figure 4.1 The CBRS system has been opened for shared access under a three-tiered spectrum access model: IUs, PAL users, and GAA users. IUs have the highest priority in this three-tiered spectrum access model, so the SAS ensures that they only receive negligible interference from PAL users and GAA users. Next, the spectrum is allocated to commercial users who

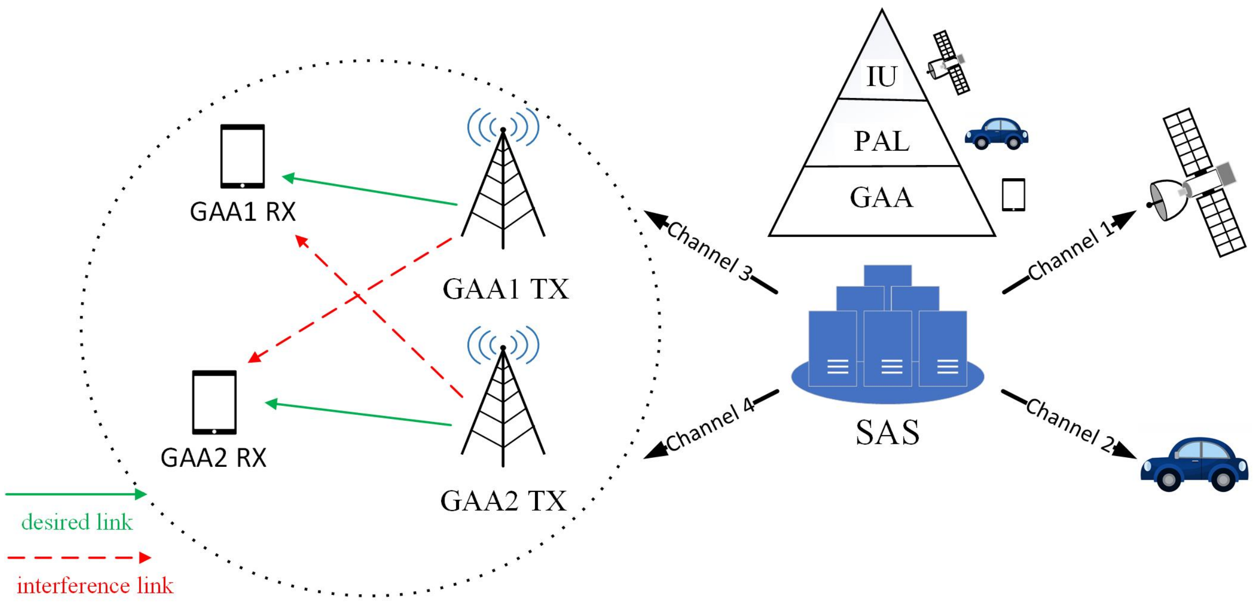


Figure 4.1: The CBRS system model.

buy PALs for a specified location and period of time. Lastly, the remaining spectrum is allocated to GAA users. It is important to note that both PAL users and GAA users should stop using the spectrum immediately if IUs need the spectrum. Similarly, GAA users have to stop using the spectrum if PAL users need the spectrum.

In this paper, we focus on designing the spectrum access strategies for GAA users to efficiently utilize the spectrum resources. We assume that there are N GAA users sharing M wireless channels, where $1, \cdots, N$ represent the index set of GAA users and $1, \cdots, M$ represent the index set of wireless channels. Without loss of generality, we assume that a GAA user can access at most one channel at a particular time. We use a one-hot vector $a^n[t] \in \{0,1\}^M$ to denote the accessed channel index of GAA user n at time t. If GAA user n accesses channel m at time t, we let $a^n[t] = \delta_m$, where δ_m is an M-dimensional vector with its m^{th} element is equal to 1. It is important to note that GAA users do not receive interference protection from the SAS, so they may interfere with each other if multiple GAA users access the same channel at the same time. To be specific, the channel capacity of GAA user n on channel m at time t is defined as

$$c_m^n[t] = B_m \log_2 \left(1 + \frac{\mathrm{SINR}_m^n[t]}{\Gamma^n}\right), \tag{4.2}$$

where

$$\mathrm{SINR}_m^n[t] = \frac{\mathbb{1}\left(a^n[t] = \delta_m\right) P^n \cdot h_m^{n,n}[t]}{\sum\limits_{z=1, z\neq n}^{N} \mathbb{1}\left(a^z[t] = \delta_m\right) P^z \cdot h_m^{z,n}[t] + N_m[t]}, \tag{4.3}$$

B_m is the bandwidth of channel m, P^n and P^z are transmit power of GAA user n and GAA user z, respectively, $\mathbb{1}(\cdot)$ is an indicator function, $h_m^{n,n}[t]$ is the channel gain of the desired link between user n's transmitter and receiver at time t, $h_m^{z,n}[t]$ is the channel gain of the interference link between user z's transmitter and user n's receiver at time t, $N_m[t]$ is the noise on channel m at time t, and Γ^n is the SNR gap corresponding to the modulation and coding strategy of user n. From the interference term in Equation (4.3), it can be observed that the channel capacity of each GAA user on a channel depends on other GAA users that transmit on the same channel. We can calculate the channel capacity of GAA user n at time t is

$$c^n[t] = \sum_{m=1}^{M} c_m^n[t]. \tag{4.4}$$

According to the spectrum management rules in the CBRS system, the SAS ensures that a GAA user accesses a channel that is not utilized by IUs or PAL users. The channel activity state, $q_m[t] \in \{0, 1\}$, is used to denote the existence of IUs and PAL users on channel m at time t. If $q_m[t]$ is equal to 0, GAA users cannot access channel m because IUs or PAL users are using channel m. On the other hand, if $q_m[t]$ is equal to 1, GAA users are allowed to access channel m. The dynamics of each channel activity state are modeled as a two-state Markov chain as shown in Figure 4.2. The transition probability of the two-state Markov

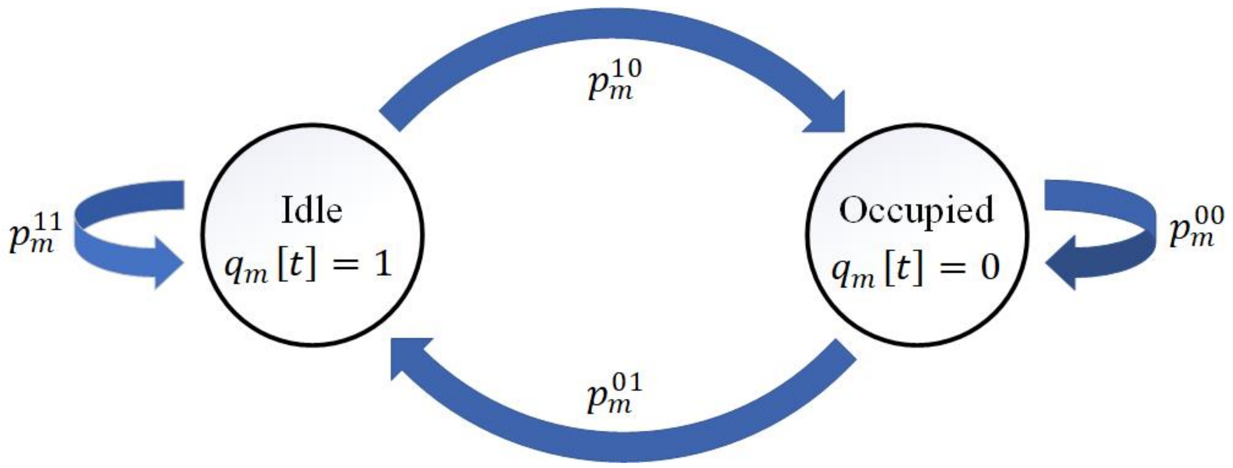


Figure 4.2: Channel activity model.

chain on channel m is denoted as

$$p_m = \begin{bmatrix} p_m^{00} & p_m^{01} \\ p_m^{10} & p_m^{11} \end{bmatrix}, \tag{4.5}$$

where p_m^{ij} $(i, j \in \{0, 1\})$ represents the probability of next activity state of channel m is j given that current activity state of channel m is i.

4.4 DRL Problem Formulation

In this section, we formulate the DSS problem using the DRL framework. To be specific, each GAA user n has a DRL agent with policy network parameters θ^n that determines its spectrum access decisions based on its observation. The local state of each agent n at time t is written as

$$s^n[t] = \left(P^n, \Gamma^n, h_m^{n,n}[t], P^z, h_m^{z,n}[t], a^z[t], N_m[t]\right), \forall m \in \mathcal{M}, \forall z \in \mathcal{N} \setminus n. \tag{4.6}$$

The joint state at time t is written as

$$s[t] = \left(P^n, \Gamma^n, h_m^{z,n}[t], a^z[t], N_m[t]\right), \forall n, z \in \mathcal{N}, \forall m \in \mathcal{M}. \tag{4.7}$$

It can be observed that the joint state space is the union of all agents' local states. The received reward of GAA user n at time t is the achieved channel capacity, which is written as

$$r^n[t] = \sum_{m=1}^{M} q_m[t] B_m \log_2 \left(1 + \frac{\mathrm{SINR}_m^n[t]}{\Gamma^n}\right). \tag{4.8}$$

We let the received reward be zero if GAA user n accesses a channel m that is currently utilized by IUs or PAL users, i.e., $q_m[t] = 0$.

Let V_π^n be the value function that the agent n follows policy π in its environment, which is defined as

$$V_\pi^n(s_i^n) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^n\left(s^n[t], a^n[t]\right) \mid s^n[0] = s_i^n, a^n[t] \sim \pi\left(s^n[t]\right)\right], \tag{4.9}$$

where s_i^n is the initial state of agent n. Let π_{θ^n} be the policy obtained from the agent n's policy network with parameters θ^n and let ρ^n be the initial state distribution of the agent n. Then we can represent the value function of agent n as a function of θ^n, which is written as

$$\begin{aligned} f^n(\theta^n) &= \mathbb{E}_{s_i^n \sim \rho^n}\left[V_{\pi_{\theta^n}}^n(s_i^n)\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^n\left(s^n[t], a^n[t]\right) \mid s^n[0] \sim \rho^n, a^n[t] \sim \pi_{\theta^n}\left(s^n[t]\right)\right]. \end{aligned} \tag{4.10}$$

The goal is to jointly learn a policy that can perform well across all agents' environments. To be specific, the sum of all agents' value functions that all agents follow the same policy

π_θ can be written as a function of θ, which is represented as:

$$f(\theta) = \sum_{n=1}^{N} f^n(\theta). \tag{4.11}$$

Accordingly, the goal is to find a joint policy π_{θ^*} with parameters θ^* that maximizes the sum of all agents' value functions:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, f(\theta). \tag{4.12}$$

In many real-world applications, it is impossible for an agent to perfectly observe the environment and obtain the complete state information. Instead, an agent receives a partial observation from the environment. In the DSS problem of the CBRS system, we define the observation received by GAA user n at time t is

$$o^n[t] = \left(c_m^n[t-1], \bar{c}_m^n[t-1]\right), \forall m \in \mathcal{M}, \tag{4.13}$$

where $\bar{c}_m^n[t-1]$ represents the average channel capacity at time $t-1$, which is defined as

$$\bar{c}_m^n[t] = \alpha \cdot c_m^n[t-1] + (1-\alpha) \cdot c_m^n[t], \tag{4.14}$$

where α is a positive number between 0 and 1.

4.5 Federated Training Algorithm

To enable the collaboration among GAA users, it is natural to design a centralized spectrum sharing strategies for GAA users. In the centralized approach, all GAA users need to share their sensory data, such as spectrum access histories, detection of interference, and transmit powers. However, gathering all GAA users' information in a spectrum access database may not always be appropriate due to privacy and security concerns. Furthermore, it may increase control overheads significantly for the GAA users to send their spectrum access information. To decrease the communication overheads and increase the data privacy, we utilize FL to design a distributed spectrum sharing strategy that does not require GAA users to share their private data.

4.5.1 Distributed Federated Policy Gradient

Algorithm 5 Federated policy gradient training algorithm.

```
Let $Q(\cdot)$ be the quantized function, $\beta \in (0,1]$, and $\eta \in (0,1]$, where $\beta > \eta$.
Initialize a shared policy network with parameters $\overline{\theta}_0$
for $k = 1, \cdots, K$ do
    Set each agent $n$'s policy network as the shared policy network:
        $\theta^n_{k-1,0} = \overline{\theta}_{k-1}, \forall n \in \mathcal{N}$.
    for $c = 1, \cdots, \tau$ do
        Each user $n$ empties its memory buffer $D^n$.
        for $t' = 1, \cdots, T$ do
            $t = \tau T(k-1) + T(c-1) + t'$
            Each agent $n$ receives observation $o^n[t]$ from the environment.
            Each agent $n$ determines $a^n[t]$ based on its policy network.
            Each agent receives reward $r^n[t]$ from the environment after executing $a^n[t]$.
            Each agent $n$ stores the training sample $(o^n[t], a^n[t], r^n[t])$ in its memory buffer $D^n$.
        end for
        Each agent $n$ updates its policy network locally using the training samples from $D^n$:
            $\theta^n_{k-1,c} = \theta^n_{k-1,c-1} + \eta \nabla f^n(\theta^n_{k-1,c-1}).$    (4.15)
    end for
    The centralized controller updates the shared policy network via taking the average of
    all quantized local policy networks:
        $\overline{\theta}_k = (1-\beta) \cdot \overline{\theta}_{k-1} + \beta \cdot \frac{1}{N} \sum_{n=1}^{N} Q\left(\theta^n_{k-1,\tau}\right)$    (4.16)
end for
```

We utilized the distributed federated policy gradient as the training algorithm, which is formally stated in Algorithm 5. In this algorithm, the SAS serves as a centralized controller to learn a shared model by aggregating the information of all locally trained models. To be specific, each agent first downloads a shared policy network from the centralized controller, and then each agent updates the policy network using the collected data from its local environment. Next, each agent sends its local policy network to the centralized controller. It is important to note that only the policy network is sent to the centralized controller, while training data are kept in each local agent. The centralized controller aggregates the information of all received local policy networks. In this paper, we let the centralized controller update the shared policy network via taking the average of the parameters of local policy networks. Then the shared policy network will be downloaded by each local agent again to start a new trainig cycle. The aforementioned training method can be summarized as a Periodically Averaging Stochastic Gradient Descent (PASGD) as the following:

$$\theta_{k,c}^{n} = \begin{cases} \frac{1}{N} \sum\limits_{n'=1}^{N} \theta_{k-1,\tau}^{n'}, & c = 1, \\ \theta_{k,c-1}^{n} + \eta \nabla f^{n}\left(\theta_{k,c-1}^{n}\right), & c = 2, \cdots, \tau, \end{cases} \tag{4.17}$$

where η is the learning rate, c is the iteration of SGD, k is the iteration of FL, $\theta_{1,1}^{n}$ is randomly initialized, $\theta_{k,c}^{n}$ is the local policy model parameters of agent n at SGD iteration c and FL iteration k. We can observed that the local policy networks are averaged after every τ iterations. If $\tau = 1$, then PASGD is equal to fully synchronous SGD as the following:

$$\theta_{k}^{n} = \theta_{k-1}^{n} + \eta \left[\frac{1}{N} \sum_{n=1}^{N} \nabla f^{n}\left(\theta_{k-1}^{n}\right) \right]. \tag{4.18}$$

For the fully synchronous SGD, the local policy networks are averaged for every SGD iteration.

4.5.2 Federated ESN-based Policy Gradient (Fed-EPG)

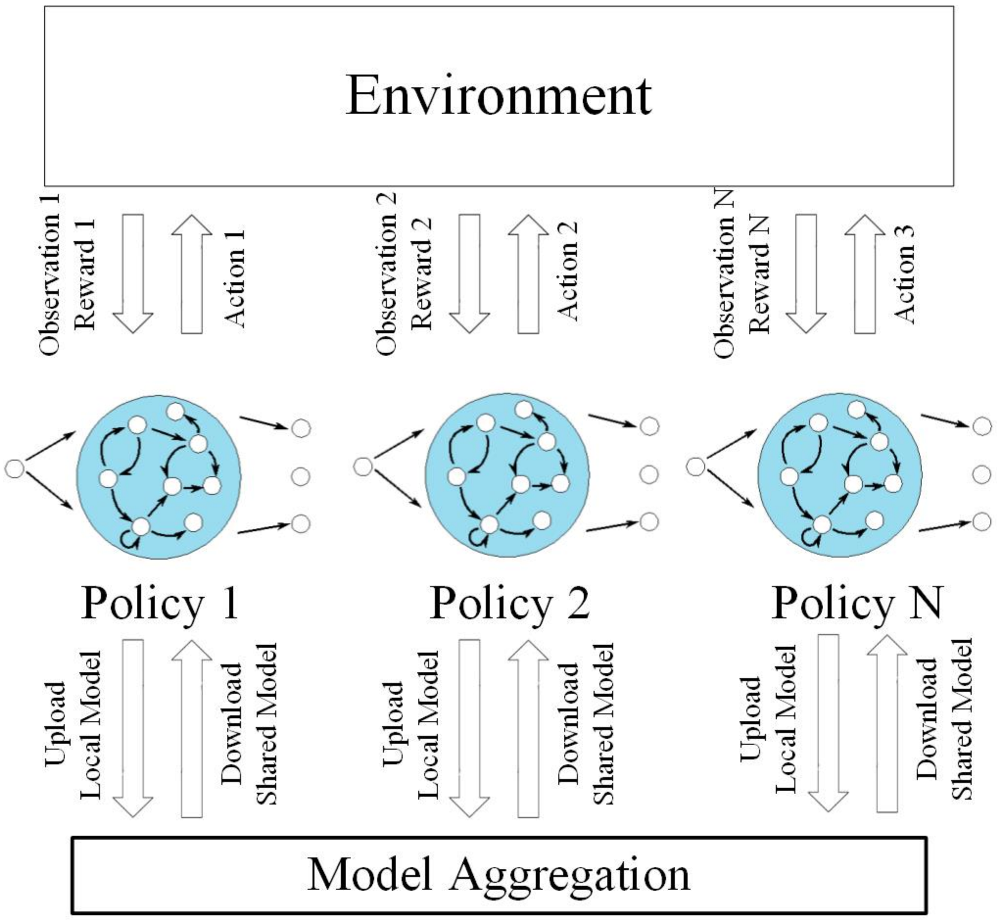


Figure 4.3: Federated ESN-based Policy Gradients.

DRL aims to solve the large state space problem in traditional reinforcement learning. Conventional RL techniques such as Q-learning have limited applications with low-dimensional state spaces. DRL utilizes deep neural network as a function approximator to accelerate the convergence time when the state space is large [68]. In our considered DSS problem, the state is continuous resulting in infinite state space, so we apply DRL as the underlying policy network.

To handle the partially observable environment and to accelerate the training, we utilize ESN as the underlying neural network structure of the policy network. Specifically, the softmax function is used as the last activation function of a ESN to generate a probability distribution over the action space. For each agent n, we let the input sequence of ESN be the observation history $o^n_{\leq t} = (o^n[1], \cdots, o^n[t])$, and the parameters of the ESN-based policy network of agent n are $\theta^n = (W^n_{\text{in}}, W^n_{\text{rec}}, W^n_{\text{out}})$. The update equations of the ESN-based policy network is written as:

$$\begin{aligned} y^n[t] &= W^n_{\text{out}} h^n[t], \\ h^n[t] &= \tanh\left(W^n_{\text{in}} o^n[t] + W^n_{\text{rec}} h^n[t-1]\right), \end{aligned} \tag{4.19}$$

where the dimension of $y^n[t]$ is equal to the action space. Accordingly, the resulting distribution over actions at time t is given as

$$\pi_{\theta^n}\left(a|o^n_{\leq t}\right) = \frac{\mathrm{e}^{y^n_a[t]}}{\sum\limits_{a' \in \mathcal{A}} \mathrm{e}^{y^n_{a'}[t]}}, \quad \forall a \in \mathcal{A}^n, \tag{4.20}$$

where $y^n_a[t]$ is the a^{th} element of $y^n[t]$.

From Algorithm 5, the data collected from agent n's environment are $(o^n[t], a^n[t], r^n[t])$ from $t = 1$ to $t = T$. According to the policy gradient algorithm, the loss function of the ESN-based policy gradient (EPG) is written as

$$\underset{W^n_{\text{out}}}{\operatorname{argmin}} \sum_{t=1}^{T} \log \pi_{\theta^n}\left(a^n[t]|o^n_{\leq t}\right) \cdot \left(\sum_{t'=1}^{t} \gamma^{t'-1} r^n[t']\right). \tag{4.21}$$

The system model of the introduced federated ESN-based policy gradient (Fed-EPG) is shown in Figure 4.3. Each agent first downloads a shared ESN-based policy network from the centralized controller, and then each agent updates the ESN-based policy network locally. Next, each agent only uploads the output weights of the ESN-based policy to the centralized controller because only weights of the ESN-based policy network are trainable. Therefore, ESN-based policy network is suitable for the FL framework because the communication overheads can be largely decreased.

4.6 Convergence Analysis

The convergence analysis is conducted under the following assumptions:

Assumption 4.1. Each agent n can compute the exact gradient $\nabla f^n(\theta)$.

Assumption 4.2. The value function $f^n(\theta)$ is Lipschitz smooth with constant L:

$$\left\|\nabla f^n(\theta_1) - \nabla f^n(\theta_2)\right\| \leq L \left\|\theta_1 - \theta_2\right\|.$$

Assumption 4.3. $\nabla f^n(\theta)$ is lower-bounded: $\left\|\nabla f^n(\theta)\right\| \leq Z$.

We can rewrite the PASGD update equations in Equation (4.17) as follows:

$$\theta_{k,c}^n = \theta_{k,c-1}^n + \eta \nabla f^n\left(\theta_{k,c-1}^n\right), \quad \forall c \in \{2, \cdots, \tau\}, \tag{4.22}$$

where $\theta_{k,1}^n = \bar{\theta}_k$ and

$$\bar{\theta}_k = \frac{1}{N}\sum_{n=1}^{N} \theta_{k-1,\tau}^n, \quad \forall k \in \{2, \cdots, K\}. \tag{4.23}$$

It is easily seen that

$$\theta_{k,\tau}^n = \bar{\theta}_k + \eta \sum_{c=1}^{\tau} \nabla f^n\left(\theta_{k,c-1}^n\right). \tag{4.24}$$

By plugging Equation (4.24) into Equation (4.23), we have

$$\bar{\theta}_k = \bar{\theta}_{k-1} + \frac{\eta}{N}\sum_{n=1}^{N}\sum_{c=1}^{\tau} \nabla f^n\left(\theta_{k-1,c-1}^n\right). \tag{4.25}$$

From Equation (4.23) and Equation (4.22), we can obtain

$$\theta_{k,c}^{n} = \bar{\theta}_k + \eta \sum_{c'=1}^{c} \nabla f^n \left(\theta_{k,c'-1}^{n}\right), \quad \forall c \in \{1, \cdots, \tau\}. \tag{4.26}$$

By using the fact that $\nabla f^n(\theta)$ is lower-bounded, we can obtain the following upper-bounds:

$$\left\|\bar{\theta}_k - \bar{\theta}_{k-1}\right\| = \left\| \frac{\eta}{N} \sum_{n=1}^{N} \sum_{c=1}^{\tau} \nabla f^n \left(\theta_{k-1,c-1}^{n}\right) \right\| \leq \eta \tau Z \tag{4.27}$$

and

$$\left\|\theta_{k,c}^{n} - \bar{\theta}_k\right\| = \left\| \eta \sum_{c'=1}^{c} \nabla f^n \left(\theta_{k,c'-1}^{n}\right) \right\| \leq \eta c Z \leq \eta \tau Z. \tag{4.28}$$

We can rewrite Equation (4.24) as

$$\theta_{k,\tau}^{n} = \bar{\theta}_k + \eta \tau \nabla f\left(\bar{\theta}_k\right) + \eta \sum_{c=1}^{\tau} \left(\nabla f^n \left(\theta_{k,c-1}^{n}\right) - \nabla f\left(\bar{\theta}_k\right)\right) \tag{4.29}$$

By plugging Equation (4.29) into Equation (4.23), we have

$$\bar{\theta}_k = \bar{\theta}_{k-1} + \eta \tau \nabla f\left(\bar{\theta}_{k-1}\right) + \frac{\eta}{N} \sum_{n=1}^{N} \sum_{c=1}^{\tau} \left(\nabla f^n \left(\theta_{k-1,c-1}^{n}\right) - \nabla f\left(\bar{\theta}_{k-1}\right)\right). \tag{4.30}$$

According to the Lipschitz smooth assumption, we have:

$$f^n\left(\bar{\theta}_k\right) \geq f^n\left(\bar{\theta}_{k-1}\right) + \left\langle \nabla f^n\left(\bar{\theta}_{k-1}\right), \bar{\theta}_k - \bar{\theta}_{k-1} \right\rangle - \frac{L}{2}\left\|\bar{\theta}_k - \bar{\theta}_{k-1}\right\|^2. \tag{4.31}$$

By plugging Equation (4.30) into Equation (4.31), we can obtain

$$f^n\left(\bar{\theta}_k\right) \geq f^n\left(\bar{\theta}_{k-1}\right) + \left\langle \nabla f^n\left(\bar{\theta}_{k-1}\right), \eta \tau \nabla f\left(\bar{\theta}_{k-1}\right) \right\rangle + A^n + B, \tag{4.32}$$

where

$$A^n = \left\langle \nabla f^n\left(\bar{\theta}_{k-1}\right), \frac{\eta}{N} \sum_{n=1}^{N} \sum_{c=1}^{\tau} \left(\nabla f^n\left(\theta_{k-1,c-1}^n\right) - \nabla f\left(\bar{\theta}_{k-1}\right)\right) \right\rangle \tag{4.33}$$

$$B = -\frac{L}{2}\left\|\bar{\theta}_k - \bar{\theta}_{k-1}\right\|^2. \tag{4.34}$$

Using the Lipschitz smooth assumption and Equation (4.28), we have

$$\left\|\nabla f^n\left(\theta_{k-1,c-1}^n\right) - \nabla f\left(\bar{\theta}_{k-1}\right)\right\| \leq L\left\|\theta_{k-1,c-1}^n - \bar{\theta}_{k-1}\right\| \leq L\eta\tau Z. \tag{4.35}$$

We use the Cauchy–Schwarz inequality to obtain the following upper-bound of A^n.

$$\begin{aligned} |A^n| &\leq \left\|\nabla f^n\left(\bar{\theta}_{k-1}\right)\right\| \cdot \left\| \frac{\eta}{N} \sum_{n=1}^{N} \sum_{c=1}^{\tau} \left(\nabla f^n\left(\theta_{k-1,c-1}^n\right) - \nabla f\left(\bar{\theta}_{k-1}\right)\right) \right\| \\ &\leq Z \cdot \frac{\eta}{N} N\tau L\eta\tau Z = L\eta^2\tau^2 Z^2. \end{aligned} \tag{4.36}$$

Thus we have

$$\frac{1}{N}\sum_{n=1}^{N} A^n \geq -L\eta^2\tau^2 Z^2. \tag{4.37}$$

In addition, we have $B \geq -\frac{L\eta^2\tau^2Z^2}{2}$ from Equation (4.27). Summing up both sides of Equation (4.32) over n, we obtain

$$\begin{aligned} f^n\left(\bar{\theta}_k\right) &\geq f^n\left(\bar{\theta}_{k-1}\right) + \frac{1}{N}\sum_{n=1}^{N}\left\langle \nabla f^n\left(\bar{\theta}_{k-1}\right), \eta\tau\nabla f\left(\bar{\theta}_{k-1}\right)\right\rangle + \frac{1}{N}\sum_{n=1}^{N} A^n + B \\ &\geq f^n\left(\bar{\theta}_{k-1}\right) + \left\langle \nabla f\left(\bar{\theta}_{k-1}\right), \eta\tau\nabla f\left(\bar{\theta}_{k-1}\right)\right\rangle - L\eta^2\tau^2Z^2 - \frac{L\eta^2\tau^2Z^2}{2} \\ &= f^n\left(\bar{\theta}_{k-1}\right) + \eta\tau\left\|\nabla f^n\left(\bar{\theta}_{k-1}\right)\right\|^2 - \frac{3L\eta^2\tau^2Z^2}{2} \end{aligned} \tag{4.38}$$

Thus, we have

$$\eta\tau\left\|\nabla f^n\left(\bar{\theta}_k\right)\right\|^2 \leq f^n\left(\bar{\theta}_{k+1}\right) - f^n\left(\bar{\theta}_k\right) + \frac{3L\eta^2\tau^2Z^2}{2}. \tag{4.39}$$

From Equation (4.39), we can obtain

$$\begin{aligned}\eta\tau\sum_{k=0}^{K-1}\left\|\nabla f^n\left(\bar{\theta}_k\right)\right\|^2 &\leq f^n\left(\bar{\theta}_K\right)-f^n\left(\bar{\theta}_0\right)+\frac{3L\eta^2\tau^2Z^2K}{2}\\ &\leq f^*-f^n\left(\bar{\theta}_0\right)+\frac{3L\eta^2\tau^2Z^2K}{2}.\end{aligned} \tag{4.40}$$

Then we have

$$\min_{k\in[0,K-1]}\left\|\nabla f^n\left(\bar{\theta}_k\right)\right\|^2\leq\frac{f^*-f^n\left(\bar{\theta}_0\right)}{\eta\tau K}+\frac{3L\eta\tau Z^2}{2}. \tag{4.41}$$

4.7 Performance Evaluation

4.7.1 Experimental Setup

In this section, we evaluate the introduced Fed-EPG method for the DSS problem in the CBRS system through simulations. There are 8 GAA users and 4 channels, where each channel is 10MHz. Each GAA user can request data transmission on one of the channels, but the final channel allocation is determined by the SAS. In other words, if a GAA user requests a wireless channel that is currently occupied by IUs or PAL users, then it cannot access that wireless channel. For each GAA user n, the distance of the transmitter and the receiver of GAA user n is denoted as d^n and the transmit power is set to P^n. As described in (), the dynamics of each channel activity state is modelled as a two-state Markov chain. For each channel m, we randomly choose p_m^{00} and p_m^{11} from a uniform distribution over $[0.8, 1]$ and $[0, 0.2]$, respectively, and then $p_m^{01} = 1 - p_m^{00}$ and $p_m^{10} = 1 - p_m^{11}$ can be calculated accordingly. To generate channel gains of desired links and interference links, we set the path loss model as $41 + 22.7\log_{10}(d)$ dB, where d is the distance between a transmitter and a receiver in meter. The small-scale channel gain follows a Rician distribution, where the ratio of the average power in the Line-of-Sight path to that in the Non-Line-of-Sight paths is set as 0.8. The noise spectral density N_0 is set to -164 (dBm/Hz).

4.7.2 Training Details

Each GAA user interacts with the environment to collect $T = 50$ training samples, and then each GAA user updates its ESN-based policy network locally using these collected samples. All ESN-based policy networks use ESNs with 32 neurons. After updating its policy network for τ iterations, each GAA user uploads the output weights of the ESN-based policy network to the centralized controller in the SAS. The centralized controller calculates a shared ESN-based policy network by aggregating the information from all local ESN-based policy networks. Accordingly, τ represents the communication overhead between the centralized controller and the local agents. If $\tau = 1$, then it is equal to the fully synchronous SGD approach. The learning rate of the local SGD and the FL are set to 0.8 and 0.95, respectively. We let $\gamma = 0.9$ and use 8 bits to represent the quantized value of each weight of the ESN-based policy network. The simulation parameters are listed in Table 4.1.

4.7.3 Results

We compare with two baselines: the fully synchronous SGD method ($\tau = 1$) and the independent learning method. Independent learning method represents that each agent aims to optimize its local reward without information aggregation in the centralized controller, so the training is totally independent. We run all our experiments with 300 random seeds, which varied the initial locations of users and the neural network initialization. Since all users are randomly moving in the experiment, user locations change over time, and the wireless environment changes accordingly. Each episode consists of 50 samples, and the reported

Table 4.1: Simulation Parameters.

Parameter	**Value**
Number of channels (M)	4
Number of GAA users (N)	8
Power of GAA user n (P^n)	50 mW
Transmit distance of GAA user n (d^n)	50m-150m
Simulation area	500m$\times$500m
Channel bandwidth (B_m)	10MHz
Path loss model (B_m)	$41 + 22.7 \log_{10}(d[m])$ dB
Small-scale fading	Rician distribution
Noise spectral density (N_0)	-164 (dBm/Hz),
Number of training data for each SGD iteration (T)	50
SGD period (τ)	5
Total number of training data	50000
FL period (K)	$1000/\tau$
SGD learning rate (η)	0.8
FL learning rate (β)	0.95
Entropy regularization parameter (λ)	0.01

curves are averaged over these 50 samples and 300 random seeds. The curves of the system throughput are shown in Fig. 4.4, where the shaded areas show the 95% confidence interval. As expected, the fully synchronous SGD achieves the best system performance because it performs federated learning every local SGD iteration. However, the communication overheads of the fully synchronous SGD is unbearable for the DSS network. On the other hand, Fed-EPG with $\tau = 5$ achieves comparable performance while maintaining reasonable communication overheads between the centralized controller and the local agents. Lastly, the independent learning method has the worst system performance due to no collaboration among agents. It is because if any GAA user updates its spectrum access policy, then the local environments of other agents change, which results in unstable environment from local agent's perspective in the independent learning method.

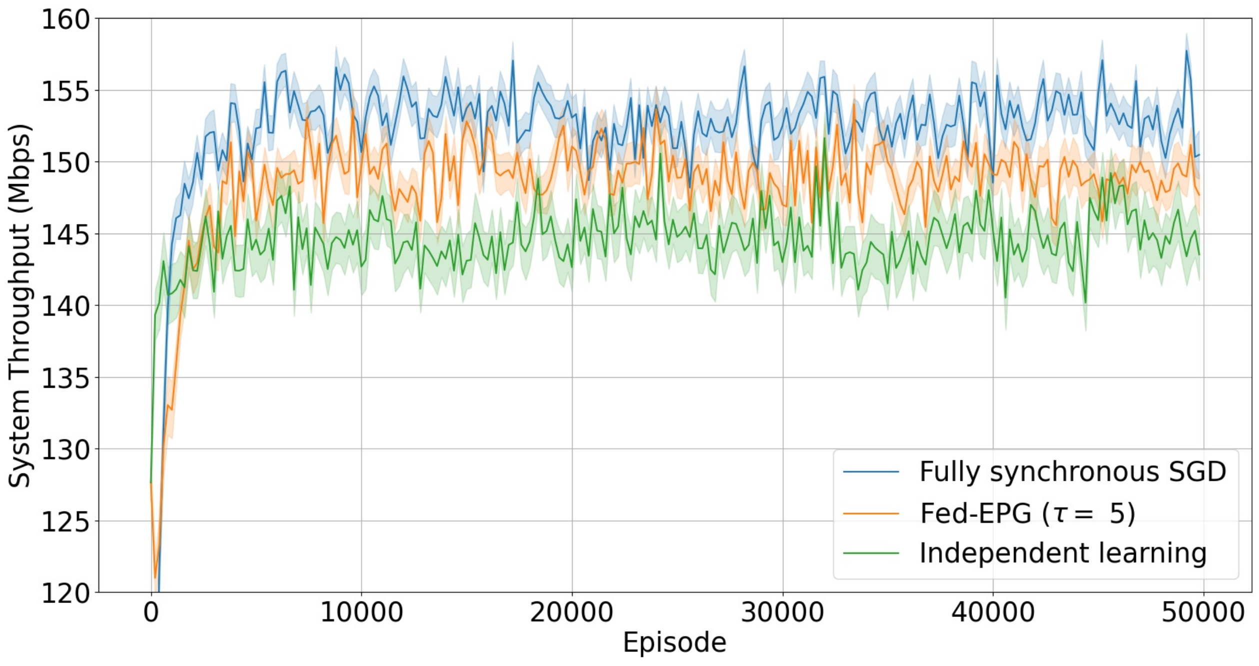


Figure 4.4: The system throughput versus time for fully synchronous SGD, independent learning, and Fed-EPG.

4.8 Conclusion

In this work, we introduce a new collaborative spectrum access strategies in the DSS network. To reduce the communication overheads and improve the data privacy, we utilize federated learning to design a distributed spectrum sharing strategy called Fed-EPG. Fed-EPG can learn a joint policy that achieves the system goal in the partially observable environment without requiring users to share their private data. We conduct theoretical analysis to show the tradeoff between the communication overhead and the convergence rate. Experimental results in the DSS network of the CBRS system are obtained to show that the introduced Fed-EPG method can achieve comparable performance with the fully synchronous method.

Chapter 5

Summary

DRL techniques are appealing to wireless communications society because they can provide a flexible architecture for solving many types of critical problems in next generation wireless networks. Although DRL shows tremendous empirical success in many other fields, applying DRL techniques to wireless systems is still difficult due to many practical constraints in the wireless systems. In the real-world wireless communication systems, the environment is always non-stationary and partially observable, and the available effective training data is extremely limited. Therefore, this dissertation aims to develop the efficient DRL methods that can be deployed in the real-world wireless applications with aforementioned properties, such as spectrum sharing, multi-user scheduling, and cell load balancing. To handle these practical challenges, we utilized the efficient training structures of ESNs to design the DEQN method and the Fed-EPG method. Furthermore, we provide theoretical analysis for DEQN and Fed-EPG to show the fast convergence properties and a guideline for designing hyper-parameters. We evaluate our methods in the DSS problem, which is a key technology in 5G and future 6G wireless systems. DSS allows the secondary system to access the licensed radio spectrum if the primary system only receive tolerable interference, which increases the utilization ratio of the precious spectrum resources. Simulation results show that our methods can enable the DRL agents to quickly update its spectrum access policy to adapt to the changing environment with limited training overheads. This work shed a light on realizing the powerful DRL techniques in the next generation wireless networks.

Bibliography

[1] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*. 2013.

[3] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[4] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[5] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 1146–1155, 2017.

[6] Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium Series*, 2015.

[7] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training

recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[8] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[9] Herbert Jaeger. The “echo state” approach to analysing and training recurrent neural networks-with an erratum note. Technical Report 34, German National Research Center For Information Technology, Jan. 2001.

[10] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade*, pages 659–686. Springer, 2012.

[11] Pierre Enel, Emmanuel Procyk, René Quilodran, and Peter Ford Dominey. Reservoir computing properties of neural dynamics in prefrontal cortex. *PLoS computational biology*, 12(6):e1004967, 2016.

[12] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: a review. *Neural Networks*, 2019.

[13] Hao-Hsuan Chang, Lingjia Liu, and Yang Yi. Deep echo state q-network (deqn) and its application in dynamic spectrum sharing for 5g and beyond. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[14] Hao-Hsuan Chang, Hao Song, Yang Yi, Jianzhong Zhang, Haibo He, and Lingjia Liu. Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach. *IEEE Internet of Things Journal*, 6(2):1938–1948, 2019.

[15] Peter Cramton and Pacharasut Sujarittanonta. Bidding and prices in the aws-3 auction. *Competitive Carrier Association*, 2015.

[16] Cisco public. Cisco annual internet report (2018–2023). Technical report, Cisco Systems Inc., San Jose, CA, USA, Mar. 2020.

[17] Roger Bacchus, Tanim Taher, Kenneth Zdunek, and Dennis Roberson. Spectrum utilization study in support of dynamic spectrum access for public safety. In *IEEE Symposium on New Frontiers in Dynamic Spectrum (DySPAN)*, pages 1–11, 2010.

[18] Václav Valenta, Roman Maršálek, Geneviève Baudoin, Martine Villegas, Martha Suarez, and Fabien Robert. Survey on spectrum utilization in europe: Measurements, analyses and observations. In *Proceedings of the fifth International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, pages 1–5, 2010.

[19] Sudeep Bhattarai, Jung-Min Jerry Park, Bo Gao, Kaigui Bian, and William Lehr. An overview of dynamic spectrum sharing: Ongoing initiatives, challenges, and a roadmap for future research. *IEEE Transactions on Cognitive Communications and Networking*, 2(2):110–128, 2016.

[20] Sue Marek. Marek's Take: Dynamic spectrum sharing may change the 5G deployment game. *Fierce Wireless*, Apr. 2019.

[21] Sean Kinney. Dynamic spectrum sharing is key to Verizon's 5G strategy. *RCR Wireless News*, Aug. 2019.

[22] Sudeep Bhattarai, Jung-Min Park, and William Lehr. Dynamic exclusion zones for protecting primary users in database-driven spectrum sharing. *IEEE/ACM Transactions on Networking*, 28(4):1506–1519, 2020.

[23] Mohd Shabbir Ali and Neelesh B Mehta. Modeling time-varying aggregate interference from cognitive radios and implications on primary exclusive zone design. In *IEEE Global Communications Conference (GLOBECOM)*, pages 3760–3765, 2013.

[24] Danda B Rawat and Gongjun Yan. Spectrum sensing methods and dynamic spectrum sharing in cognitive radio networks: A survey. *International Journal of Research and Reviews in Wireless Sensor Networks*, 1(1):1–13, 2011.

[25] Ashish Ranjan, Balwinder Singh, et al. Design and analysis of spectrum sensing in cognitive radio based on energy detection. In *2016 International Conference on Signal and Information Processing (IConSIP)*, pages 1–5. IEEE, 2016.

[26] Jarmo Lundén, Visa Koivunen, Anu Huttunen, and H Vincent Poor. Collaborative cyclostationary spectrum sensing for cognitive radio systems. *IEEE Transactions on Signal Processing*, 57(11):4182–4195, 2009.

[27] Xinzhi Zhang, Rong Chai, and Feifei Gao. Matched filter based spectrum sensing and power level detection for cognitive radio network. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1267–1270, 2014.

[28] Abhijeet Bishnu and Vimal Bhatia. Logdet covariance based spectrum sensing under colored noise. *IEEE Transactions on Vehicular Technology*, 67(7):6716–6720, 2018.

[29] Youness Arjoune and Naima Kaabouch. A comprehensive survey on spectrum sensing in cognitive radio networks: Recent advances, new challenges, and future research directions. *Sensors*, 19(1):126, 2019.

[30] David Tse and Pramod Viswanath. *Fundamentals of wireless communication.* Cambridge university press, 2005.

[31] Yun Liao, Tianyu Wang, Lingyang Song, and Zhu Han. Listen-and-talk: Full-duplex cognitive radio networks. In *IEEE Global Communications Conference (GLOBECOM)*, pages 3068–3073, 2014.

[32] Zhuoran Fu, Wenjun Xu, Zhiyong Feng, Xuehong Lin, and Jiaru Lin. Throughput analysis of LTE-licensed-assisted access networks with imperfect spectrum sensing. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2017.

[33] Stergios Stotas and Arumugam Nallanathan. Enhancing the capacity of spectrum sharing cognitive radio networks. *IEEE Transactions on Vehicular Technology*, 60(8):3768–3779, 2011.

[34] Juhyeon Lee and Hyung-Kun Park. Channel prediction-based channel allocation scheme for multichannel cognitive radio networks. *Journal of Communications and Networks*, 16(2):209–216, 2014.

[35] Vamsi Krishna Tumuluru, Ping Wang, and Dusit Niyato. Channel status prediction for cognitive radio networks. *Wireless Communications and Mobile Computing*, 12(10):862–874, 2012.

[36] Xiukui Li and Seyed A Zekavat. Cognitive radio based spectrum sharing: Evaluating channel availability via traffic pattern prediction. *Journal of Communications and Networks*, 11(2):104–114, 2009.

[37] Yonghua Wang, Zifeng Ye, Pin Wan, and Jiajun Zhao. A survey of dynamic spectrum allocation based on reinforcement learning algorithms in cognitive radio networks. *Artificial Intelligence Review*, 51(3):493–506, 2019.

[38] Shangxing Wang, Hanpeng Liu, Pedro Henrique Gomes, and Bhaskar Krishnamachari. Deep reinforcement learning for dynamic multichannel access in wireless networks. *IEEE Transactions on Cognitive Communications and Networking*, 4(2):257–265, 2018.

[39] Oshri Naparstek and Kobi Cohen. Deep multi-user reinforcement learning for distributed

dynamic spectrum access. *IEEE Transactions on Wireless Communications*, 18(1):310–323, 2018.

[40] Hao-Hsuan Chang, Hao Song, Yang Yi, Jianzhong Zhang, Haibo He, and Lingjia Liu. Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach. *IEEE Internet of Things Journal*, 6(2):1938–1948, 2018.

[41] Yue Xu, Jianyuan Yu, and R Michael Buehrer. The application of deep reinforcement learning to distributed spectrum access in dynamic heterogeneous environments with partial observations. *IEEE Transactions on Wireless Communications*, 19(7):4494–4506, 2020.

[42] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures. Technical Specification (TS) 36.213, 3rd Generation Partnership Project (3GPP), Jun. 2019. version 15.6.0.

[43] Lingjia Liu, Runhua Chen, Stefan Geirhofer, Krishna Sayana, Zhihua Shi, and Yongxing Zhou. Downlink MIMO in LTE-advanced: SU-MIMO vs. MU-MIMO. *IEEE Communications Magazine*, 50(2):140–147, 2012.

[44] Alessandro Chiumento, Mehdi Bennis, Claude Desset, Liesbet Van der Perre, and Sofie Pollin. Adaptive CSI and feedback estimation in LTE and beyond: a Gaussian process regression approach. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):168, Jun. 2015.

[45] Martin Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328, 2005.

[46] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. 2012.

[47] Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pages 1042–1051, 2019.

[48] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep Q-learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, pages 486–489, 2020.

[49] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alexander J Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems*, pages 2647–2655, 2015.

[50] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[51] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

[52] Marco Chiani. Distribution of the largest eigenvalue for real wishart and gaussian random matrices and a simple approximation for the tracy–widom distribution. *Journal of Multivariate Analysis*, 129:69–81, 2014.

[53] Matthias Wellens and Petri Mähönen. Lessons learned from an extensive spectrum occupancy measurement campaign and a stochastic duty cycle model. *Mobile Networks and Applications*, 15(3):461–474, 2010.

[54] Esa Hyytiä and Jorma Virtamo. Random waypoint mobility model in cellular networks. *Wireless Networks*, 13(2):177–188, 2007.

[55] Yasir Saleem and Mubashir Husain Rehmani. Primary radio user activity models for cognitive radio networks: A survey. *Journal of Network and Computer Applications*, 43:1–16, 2014.

[56] Pablo Ameigeiras, Yuanye Wang, Jorge Navarro-Ortiz, Preben E Mogensen, and Juan M Lopez-Soler. Traffic models impact on ofdma scheduling design. *EURASIP Journal on Wireless Communications and Networking*, 2012(1):61, 2012.

[57] Matthias Wellens, Janne RiihijäRvi, and Petri MäHöNen. Empirical time and frequency domain models of spectrum use. *Physical Communication*, 2(1-2):10–32, 2009.

[58] Lingjia Liu, Young-Han Nam, and Jianzhong Zhang. Proportional fair scheduling for multi-cell multi-user mimo systems. In *Annual Conference on Information Sciences and Systems*, pages 1–6, 2010.

[59] CEPT. WINNER II Channel Models. Technical Report D1.1.2, European Conference of Postal and Telecommunications Administrations (CEPT), Feb. 2008. version 1.2.

[60] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[61] Michiel Hermans and Benjamin Schrauwen. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198, 2013.

[62] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. Deep reservoir computing: a critical experimental analysis. *Neurocomputing*, 268:87–99, 2017.

[63] Zhou Zhou, Lingjia Liu, Jianzhong Zhang, and Yang Yi. Deep reservoir computing meets 5g mimo-ofdm systems in symbol detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[64] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[65] Pinxin Long, Tingxiang Fanl, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *International Conference on Robotics and Automation*, pages 6252–6259, 2018.

[66] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[67] Federal Communications Commission. Amendment of the commission's rules with regard to commercial operations in the 3550-3650 mhz band. *Report and Order and Second Further Notice of Proposed Rulemaking, GN Docket*, (12-354), 2015.

[68] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[69] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Appendices

Appendix A

Proofs of theorems in Section 2.6

In this supplementary material, we present detailed proofs of Theorem 2.2, 2.3, 2.4, 2.5, and 2.7. In the following analysis, we denote $\|\cdot\|$ as the ℓ_2 norm of vectors and the spectral norm of matrices. We first define the function class constructed by RNN/ESN. A function class, $F_t := \{f_t : o_{\leq t} \to y_t\}$, represents the family of functions that maps the first t observations to the t-th output according to update equations in Equation (2.7). In our analysis, the function class F_t can be a family of RNN/ESN satisfying Assumption 2.3 and Assumption 2.4. We now define the exterior ϵ-net and ϵ-covering number of a given function class F_t.

Definition A.1. (The exterior ϵ-net and ϵ-covering number) Given a function class F_t and let $\epsilon > 0$. We can define a function class, $C(F_t, \epsilon, d)$, as the exterior ϵ-net of F_t if for any $f_t \in F_t$, there exists $\hat{f}_t \in C(F_t, \epsilon, d)$ such that

$$\sup_{o_{\leq t}} d\left(f_t(o_{\leq t}), \hat{f}_t(o_{\leq t})\right) \leq \epsilon$$

where d is the distance metric, $f_t(o_{\leq t})$ and $\hat{f}_t(o_{\leq t})$ is the t-th output of RNN/ESN calculated by f_t and $\hat{f}_t$ given the input sequence $o_{\leq t}$, respectively. The smallest possible cardinality of $C(F_t, \epsilon, d)$ is called the exterior ϵ-covering number of F_t.

In our analysis, we consider d as the ℓ_2 distance and denote the exterior ϵ-covering number of F_t with respect to the ℓ_2 distance by $N_\epsilon^{\text{ext}}(F_t)$. To begin our analysis, we specify two types of function classes where functions are represented by Equation (2.7) with different constraints on weight matrices. The first type of function class, $A_t(a, b)$, contains a family of functions with weights satisfying

$$\|W_{\text{in}}\| \leq a, \|W_{\text{rec}}\| \leq b, \|W_{\text{out}}\| \leq \sigma_{\text{out}}^{\max}. \tag{A.1}$$

The second function class, $B_t(a, b)$, contains a family of functions with weights satisfying

$$\|W_{\text{in}}\| = a, \|W_{\text{rec}}\| = b, \|W_{\text{out}}\| \leq \sigma_{\text{out}}^{\max}. \tag{A.2}$$

In our analysis, we assume that the weights of both RNN and ESN satisfy Assumption 2.4. Since all weights are trainable in RNNs, the function class constructed by RNNs is denoted by $A_t(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max})$. On the other hand, the function class constructed by ESNs is random because input weights and recurrent weights of ESNs are initialized randomly and are untrained. Let S_{in} and S_{rec} be two random variables representing the spectral norm of random input weights and random recurrent weights, respectively. Then the random function class constructed by ESNs is denoted by $B_t(S_{\text{in}}, S_{\text{rec}})$, where the interval of S_{in} and S_{rec} are $(0, \sigma_{\text{in}}^{\max}]$ and $(0, \sigma_{\text{rec}}^{\max}]$, respectively. Therefore, the exterior ϵ-covering number of RNN and ESN with respect to the ℓ_2 norm are

$$N_{\text{RNN},\epsilon,t}^{\text{ext}} = N_{\epsilon}^{\text{ext}}\left(A_t\left(\sigma_{in}^{\max}, \sigma_{rec}^{\max}\right)\right) \tag{A.3}$$

and

$$\mathcal{N}_{\text{ESN},\epsilon,t}^{\text{ext}} = N_{\epsilon}^{\text{ext}}\left(B_t\left(S_{\text{in}}, S_{\text{rec}}\right)\right) \tag{A.4}$$

respectively. Note that $\mathcal{N}_{\text{ESN},\epsilon,t}^{\text{ext}}$ is a random variable because the function class $B_t(S_{\text{in}}, S_{\text{rec}})$ is a random function class. To analyze the theoretical properties of DEQN, we define the expected value of $\mathcal{N}^{\text{ext}}_{\text{ESN},\epsilon,t}$ as

$$N^{\text{ext}}_{\text{ESN},\epsilon,t} = \mathbb{E}[\mathcal{N}^{\text{ext}}_{\text{ESN},\epsilon,t}] = \iint\limits_{\substack{0<a\le\sigma^{\max}_{\text{in}}\\ 0<b\le\sigma^{\max}_{\text{rec}}}} N^{\text{ext}}_{\epsilon}\left(B_t\left(a,b\right)\right) f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b)\mathrm{d}a\mathrm{d}b, \tag{A.5}$$

where $f_{S_{\text{in}}}(\cdot)$ and $f_{S_{\text{rec}}}(\cdot)$ are the probability density function (PDF) of S_{in} and S_{rec}, respectively. We use $N^{\text{ext}}_{\text{ESN},\epsilon,t}$ for analysis without loss of generality.

A.1 Proof of Theorem 2.2

Theorem 6.2 in [48] shows that: For any $\delta \in (0,1]$ and $\epsilon' > 0$, we have

$$\begin{aligned} \eta^2_{\max,t} \le & (1+\delta)^2 \sup_{\theta'\in\Theta} \inf_{\theta\in\Theta} \mathbb{E}_{\sigma}\left[(\mathcal{B}Q_{\theta'} - Q_{\theta})^2\right] \\ & + C_1' \cdot \frac{V^2_{\max}}{T\cdot\delta} \cdot \log N_{\epsilon',t,\ell_\infty} + C_2' \cdot V_{\max} \cdot \epsilon', \end{aligned} \tag{A.6}$$

where

$$\begin{aligned} C_1' &= (1+\delta)^2 \left[(1+\delta)\sqrt{2T} + 64/V_{\max}\right] \\ C_2' &= (1+\delta)(4+\sqrt{2T}) + 18, \end{aligned}$$

$\mathcal{B}$ is the Bellman optimality operator, and $N_{\epsilon,t,\ell_\infty}$ is the ϵ-covering number of RNN/ESN with respect to the ℓ_∞ norm. Similar to Definition A.1, the ϵ-covering number of a given function class F_t is the smallest cardinality of the ϵ-net of F_t, but the ϵ-net of F_t is required to be a subset of F_t.

From Exercise 4.2.9 in [69], we have the following relation between the covering number and the exterior covering number:

$$N_{\epsilon',t,\ell_\infty} \le N^{\text{ext}}_{\epsilon,t,\ell_\infty}, \tag{A.7}$$

where $\epsilon = \epsilon'/2$ and $N^{\text{ext}}_{\epsilon,t,\ell_\infty}$ is the exterior ϵ-covering number of RNN/ESN with respect to the ℓ_∞ norm.

Let $N^{\text{ext}}_{\epsilon,t,\ell_2}$ and $C^{\text{ext}}_{\epsilon,t,\ell_2}$ be the exterior ϵ-covering number and the minimal exterior ϵ-net of RNN/ESN with respect to the ℓ_2 norm, respectively, where $\left|C^{\text{ext}}_{\epsilon,t,\ell_2}\right| = N^{\text{ext}}_{\epsilon,t,\ell_2}$. According to the exterior ϵ-net definition, for any function f_t in RNN/ESN, there exists $\hat{f}_t$ in $C^{\text{ext}}_{\epsilon,t,\ell_2}$ such that $sup_{o_{\le t}} \left\| f_t(o_{\le t}) - \hat{f}_t(o_{\le t}) \right\|_2 \le \epsilon$. Furthermore, $C^{\text{ext}}_{\epsilon,t,\ell_2}$ must be an exterior ϵ-net of RNN/ESN with respect to the ℓ_∞ norm because $\|x\|_\infty \le \|x\|_2$. Since the exterior ϵ-covering number is the cardinality of the minimal exterior ϵ-net, we have

$$N^{\text{ext}}_{\epsilon,t,\ell_\infty} \le N^{\text{ext}}_{\epsilon,t,\ell_2}. \tag{A.8}$$

Let $\delta = 1$ and $N^{\text{ext}}_{\epsilon,t} = N^{\text{ext}}_{\epsilon,t,\ell_2}$. By utilizing $N_{\epsilon',t,\ell_\infty} \le N_{\epsilon,t}$, we can extend Equation (A.6) to

$$\begin{aligned} \eta^2_{\max,t} \le & 4 \sup_{\theta' \in \Theta} \inf_{\theta \in \Theta} \mathbb{E}_\sigma \left[(\mathcal{B} Q_{\theta'} - Q_\theta)^2 \right] \\ & + C_1 \cdot \frac{V^2_{\max}}{T} \cdot \log N^{\text{ext}}_{\epsilon,t} + C_2 \cdot V_{\max} \cdot \epsilon, \end{aligned} \tag{A.9}$$

where

$$C_1 = 8\sqrt{2T} + \frac{256}{V_{\max}},$$
$$C_2 = 16 + 4\sqrt{2T} + 36.$$

A.2 Proof of Theorem 2.3

The function class constructed by RNNS is $A_t(\sigma^{\max}_{\text{in}}, \sigma^{\max}_{\text{rec}})$, while the random function class constructed by ESNs is $B_t(S_{\text{in}}, S_{\text{rec}})$, where the interval of S_{in} and S_{rec} are $(0, \sigma^{\max}_{\text{in}}]$ and $(0, \sigma^{\max}_{\text{rec}}]$, respectively. For any constants $a \in (0, \sigma^{\max}_{\text{in}}]$ and $b \in (0, \sigma^{\max}_{\text{rec}}]$, we have $B_t(a, b)$ as a realization of $B_t(S_{\text{in}}, S_{\text{rec}})$. According to Equation (A.1) and Equation (A.2), $B_t(a, b)$

is a subset of $A_t(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max})$. Let C be the smallest exterior ϵ-net of $A_t(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max})$, i.e., $|C| = N_\epsilon^{\text{ext}}(A_t(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max}))$. From Definition A.1, for any $f_t \in A_t(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max})$, there exists $\hat{f}_t \in C$ such that $sup_{o_{\le t}} d\left(f_t(o_{\le t}), \hat{f}_t(o_{\le t})\right) \le \epsilon$, where d is the ℓ_2 distance metric. Since each function of $B_t(a,b)$ belongs to $A_t(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max})$, C must be an exterior ϵ-net of $B_t(a,b)$. Note that the exterior ϵ-covering number is the cardinality of the smallest exterior ϵ-net, so we have

$$N_\epsilon^{\text{ext}}(B_t(a,b)) \le |C| = N_\epsilon^{\text{ext}}(A_t(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max})). \tag{A.10}$$

Since Equation (A.10) holds for every realization of $B_t(S_{\text{in}}, S_{\text{rec}})$, we can have

$$\Pr\left(\mathcal{N}_{\text{ESN},\epsilon,t}^{\text{ext}} \le N_{\text{RNN},\epsilon,t}^{\text{ext}}\right) = 1. \tag{A.11}$$

A.3 Proof of Theorem 2.4

A.3.1 Proof of the upper-bound of $N_{\textbf{RNN},\epsilon,t}^{\text{ext}}$

For the function class $A_t(a,b)$, we establish a recursion that relates h_t with h_{t-1} as follows:

$$\begin{aligned}
\|h_t\| &= \|\tanh(W_{\text{in}} o_t + W_{\text{rec}} h_{t-1})\| \\
&\le \|W_{\text{in}} o_t + W_{\text{rec}} h_{t-1}\| \\
&\le \|W_{\text{in}}\| \cdot \|o_t\| + \|W_{\text{rec}}\| \cdot \|h_{t-1}\| \\
&\le aB_o + b \cdot \|h_{t-1}\|,
\end{aligned} \tag{A.12}$$

where the last inequality follows Assumption 2.3 and Equation (A.1). Applying Equation (A.12) recursively with $h_0 = \mathbf{0}$, we get

$$\|h_t\| \leq B_o a \sum_{i=0}^{t-1} b^i = B_o a \frac{1-b^t}{1-b} \tag{A.13}$$

Then we can upper bound $\|y_t\|$ as follow:

$$\|y_t\| = \|W_{\text{out}} h_t\| \leq B_o \sigma_{\text{out}}^{\max} a \frac{1-b^t}{1-b} \tag{A.14}$$

Let $R = B_o \sigma_{\text{out}}^{\max} a \frac{1-b^t}{1-b}$ and $B^n(r)$ be the Euclidean ball in $\mathbb{R}^n$ centered at the origin with radius r. Since $\|y_t\| \leq R$, $B^{d_y}(R)$ contains all possible y_t from $A_t(a, b)$.

We define the set that contains all possible y_t from a given function class F_t as

$$\phi(F_t) = \{f_t(o_{\leq t}) : \forall f_t \in F_t\} \tag{A.15}$$

According to Proposition 4.2.12 in [69], we can bound the ϵ-covering number using volume as follows:

$$\frac{\text{vol}\left(\phi(F_t)\right)}{\text{vol}\left(B^n(\epsilon)\right)} \leq N_\epsilon^{\text{ext}}\left(F_t\right) \leq \frac{\text{vol}\left(\phi(F_t) + B^n(\epsilon/2)\right)}{\text{vol}\left(B^n(\epsilon/2)\right)}, \tag{A.16}$$

where vol$(\cdot)$ denotes the volume of a set and

$$\phi(F_t) + B^n(\epsilon/2) = \{a + b : a \in \phi(F_t), b \in B^n(\epsilon/2)\}.$$

Since the set of all possible y_t from $A_t(a, b)$ is contained in $B^{d_y}(R)$, we have the following relation:

$$\begin{aligned} N_\epsilon^{\text{ext}}\left(A_t(a,b)\right) &\leq \frac{\operatorname{vol}\left(B^{d_y}(R)+B^{d_y}(\epsilon/2)\right)}{\operatorname{vol}\left(B^{d_y}(\epsilon/2)\right)} \\ &= \left[\frac{R+\epsilon/2}{\epsilon/2}\right]^{d_y} \\ &= \left[1+\frac{2B_o\sigma_{\text{out}}^{\max}a\left(1-b^t\right)}{\epsilon(1-b)}\right]^{d_y} \end{aligned}$$

By letting $U\left(a,b\right)=\left[1+\frac{2B_o\sigma_{out}^{\max}a\left(1-b^t\right)}{\epsilon(1-b)}\right]^{d_y}$, we have

$$N_\epsilon^{\text{ext}}\left(A_t(a,b)\right)\leq U\left(a,b\right). \tag{A.17}$$

Since $N_{\text{RNN},\epsilon,t}^{\text{ext}}=N_\epsilon^{\text{ext}}\left(A_t\left(\sigma_{\text{in}}^{\max},\sigma_{\text{rec}}^{\max}\right)\right)$, we have

$$N_{\text{RNN},\epsilon,t}^{\text{ext}}\leq U\left(\sigma_{\text{in}}^{\max},\sigma_{\text{rec}}^{\max}\right). \tag{A.18}$$

A.3.2 Proof of the lower-bound of $N_{\mathbf{RNN},\epsilon,t}^{\mathbf{ext}}$

Let $G_t(a,b)$ be the function class that has the following properties: 1) W_{in} is a full rank matrix with all singular values equal to a. 2) W_{rec} is a full rank matrix with all singular values equal to b. 3) W_{out} is a full rank matrix with all singular values equal to $\sigma_{\text{out}}^{\max}$. We can find that $G_t(a,b)$ is a subset of $A_t(a,b)$ from Equation (A.1). Furthermore, $G_t(a,b)$ is also a subset of $B_t(a,b)$ from Equation (A.2). Similar to the analysis in Equation (A.10), we have

$$N_\epsilon^{\text{ext}}\left(G_t\left(a,b\right)\right)\leq N_\epsilon^{\text{ext}}\left(A_t\left(a,b\right)\right). \tag{A.19}$$

$$N_\epsilon^{\text{ext}}\left(G_t\left(a,b\right)\right)\leq N_\epsilon^{\text{ext}}\left(B_t\left(a,b\right)\right). \tag{A.20}$$

Now we identify the set of all possible y_t from $G_t(a,b)$. From Assumption 2.3, the set of o_t is an Euclidean ball with radius B_o. Since $h_0 = \mathbf{0}$ and W_{in} is a full rank matrix with all singular values equal to a, the set of all possible h_1 from $G_t(a,b)$ is an Euclidean ball with radius

$$H_1(a,b) = \tanh\left(aB_o\right) \tag{A.21}$$

Furthermore, we define the set of all possible $h_{t'}$ from $G_t(a,b)$ is an Euclidean ball with radius $H_{t'}(a,b)$, where $t' \leq t$. By using the fact that W_{rec} is a full rank matrix with all singular values equal to b, we can establish a recursion that relates $H_{t'}(a,b)$ with $H_{t'-1}(a,b)$ as follows:

$$H_{t'}(a,b) = \tanh\left(aB_o + bH_{t'-1}(a,b)\right). \tag{A.22}$$

Finally, we find that the set of all possible y_t from $G_t(a,b)$ is an Euclidean ball with radius $\sigma_{\text{out}}^{\max} H_t(a,b)$ because W_{out} is a full rank matrix with all singular values equal to $\sigma_{\text{out}}^{\max}$. According to Equation (A.16), we have

$$\begin{aligned} N_\epsilon^{\text{ext}}\left(G_t\left(a,b\right)\right) &\geq \frac{\text{vol}\left(B^{d_y}\left(\sigma_{\text{out}}^{\max} H_t(a,b)\right)\right)}{\text{vol}\left(B^{d_y}(\epsilon)\right)} \\ &= \left[\frac{\sigma_{\text{out}}^{\max} H_t(a,b)}{\epsilon}\right]^{d_y} \end{aligned} \tag{A.23}$$

By letting $L\left(a,b\right) = \left[\frac{\sigma_{out}^{\max} H_t(a,b)}{\epsilon}\right]^{d_y}$, we have

$$N_\epsilon^{\text{ext}}\left(G_t\left(a,b\right)\right) \geq L\left(a,b\right). \tag{A.24}$$

Since $N_{\text{RNN},\epsilon,t}^{\text{ext}} = N_\epsilon^{\text{ext}}\left(A_t\left(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max}\right)\right)$, we have

$$N_{\text{RNN},\epsilon,t}^{\text{ext}} \geq N_\epsilon^{\text{ext}}\left(G_t\left(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max}\right)\right) \geq L\left(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max}\right). \tag{A.25}$$

A.3.3 Proof of the upper-bound of $N^{\text{ext}}_{\text{ESN},\epsilon,t}$

From Equation (A.1) and Equation (A.2), we can find that $B_t(a,b)$ is a subset of $A_t(a,b)$. Similar to the analysis in Equation (A.10), we have

$$N^{\text{ext}}_{\epsilon}\left(B_t\left(a,b\right)\right) \leq N^{\text{ext}}_{\epsilon}\left(A_t\left(a,b\right)\right). \tag{A.26}$$

From the definition of $N^{\text{ext}}_{\text{ESN},\epsilon,t}$ in Equation (A.5), we can obtain

$$\begin{aligned} N^{\text{ext}}_{\text{ESN},\epsilon,t} &= \iint\limits_{\substack{0<a\leq\sigma^{\max}_{\text{in}}\\0<b\leq\sigma^{\max}_{\text{rec}}}} N^{\text{ext}}_{\epsilon}\left(B_t\left(a,b\right)\right) f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b)\mathrm{d}a\mathrm{d}b \\ &\leq \iint\limits_{\substack{0<a\leq\sigma^{\max}_{\text{in}}\\0<b\leq\sigma^{\max}_{\text{rec}}}} N^{\text{ext}}_{\epsilon}\left(A_t\left(a,b\right)\right) f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b)\mathrm{d}a\mathrm{d}b \\ &\leq \iint\limits_{\substack{0<a\leq\sigma^{\max}_{\text{in}}\\0<b\leq\sigma^{\max}_{\text{rec}}}} U\left(a,b\right) f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b)\mathrm{d}a\mathrm{d}b, \end{aligned}$$

where the last inequality follows Equation (A.17).

A.3.4 Proof of the lower-bound of $N^{\text{ext}}_{\text{ESN},\epsilon,t}$

From the definition of $N^{\text{ext}}_{\text{ESN},\epsilon,t}$ in Equation (A.5), we have

$$\begin{aligned}
N^{\text{ext}}_{\text{ESN},\epsilon,t} &= \iint\limits_{\substack{0<a\leq\sigma^{\max}_{\text{in}}\\0<b\leq\sigma^{\max}_{\text{rec}}}} N^{\text{ext}}_{\epsilon}\left(B_t\left(a,b\right)\right) f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b)\mathrm{d}a\mathrm{d}b \\
&\geq \iint\limits_{\substack{0<a\leq\sigma^{\max}_{\text{in}}\\0<b\leq\sigma^{\max}_{\text{rec}}}} N^{\text{ext}}_{\epsilon}\left(G_t\left(a,b\right)\right) f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b)\mathrm{d}a\mathrm{d}b \\
&\geq \iint\limits_{\substack{0<a\leq\sigma^{\max}_{\text{in}}\\0<b\leq\sigma^{\max}_{\text{rec}}}} L\left(a,b\right) f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b)\mathrm{d}a\mathrm{d}b,
\end{aligned}$$

where the first inequality follows Equation (A.20) and the second inequality follows Equation (A.24).

A.3.5 Proof of the lower-bound of $N^{\text{ext}}_{\text{RNN},\epsilon,t} - N^{\text{ext}}_{\text{ESN},\epsilon,t}$

From Equation (A.3) and Equation (A.5), we have

$$\begin{aligned}
N^{\text{ext}}_{\text{RNN},\epsilon,t} - N^{\text{ext}}_{\text{ESN},\epsilon,t} &= N^{\text{ext}}_{\epsilon}\left(A_t\left(\sigma^{\max}_{in},\sigma^{\max}_{rec}\right)\right) - \iint\limits_{\substack{0<a\leq\sigma^{\max}_{\text{in}}\\0<b\leq\sigma^{\max}_{\text{rec}}}} N^{\text{ext}}_{\epsilon}\left(B_t\left(a,b\right)\right) f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b)\mathrm{d}a\mathrm{d}b \\
&= \iint\limits_{\substack{0<a\leq\sigma^{\max}_{\text{in}}\\0<b\leq\sigma^{\max}_{\text{rec}}}} \left[N^{\text{ext}}_{\epsilon}\left(A_t\left(\sigma^{\max}_{in},\sigma^{\max}_{rec}\right)\right) - N^{\text{ext}}_{\epsilon}\left(B_t\left(a,b\right)\right)\right] f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b)\mathrm{d}a\mathrm{d}b.
\end{aligned}$$

From Equation (A.10), we have

$$N^{\text{ext}}_{\epsilon}\left(A_t\left(\sigma^{\max}_{\text{in}},\sigma^{\max}_{\text{rec}}\right)\right) - N^{\text{ext}}_{\epsilon}\left(B_t\left(a,b\right)\right) \geq 0 \tag{A.27}$$

for any constants $a \in (0, \sigma_{\text{in}}^{\max}]$ and $b \in (0, \sigma_{\text{rec}}^{\max}]$. Furthermore, we can obtain

$$N_{\epsilon}^{\text{ext}}\left(A_t\left(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max}\right)\right) \geq L\left(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max}\right) \tag{A.28}$$

from Equation (A.24) and

$$N_{\epsilon}^{\text{ext}}\left(B_t\left(a, b\right)\right) \leq N_{\epsilon}^{\text{ext}}\left(A_t\left(a, b\right)\right) \leq U(a, b) \tag{A.29}$$

from Equation (A.26) and Equation (A.17).

Combining Equation (A.27), Equation (A.28), and Equation (A.29), we can derive that

$$N_{\text{RNN},\epsilon,t}^{\text{ext}} - N_{\text{ESN},\epsilon,t}^{\text{ext}} \geq \iint\limits_{\substack{0<a\leq\sigma_{\text{in}}^{\max} \\ 0<b\leq\sigma_{\text{rec}}^{\max}}} \left[L\left(\sigma_{\text{in}}^{\max}, \sigma_{\text{rec}}^{\max}\right) - U(a, b)\right]^{+} f_{S_{\text{in}}}(a) f_{S_{\text{rec}}}(b) \mathrm{d}a\mathrm{d}b, \tag{A.30}$$

where $[x]^{+} = \max(0, x)$.

A.4 Proof of Theorem 2.5

To satisfy the spectrum norm constraints on weights in Assumption 2.4, W_{in} and W_{rec} of ESNs are initialized using Algorithm 3. Take W_{in} as an example, we first initialize W_{in}^{1} as a $d_o \times d_h$ Gaussian random matrix. Without loss of generality, we assume that the number of neurons, d_h, is a large integer. According to [52], $T_{\text{in}} = \frac{\|W_{\text{in}}^{1}\| - \mu_{\text{in}}}{\gamma_{\text{in}}}$ follows type-1 Tracy-Widom distribution, where

$$\mu_{\text{in}} = \left(\sqrt{d_o - 0.5} + \sqrt{d_h - 0.5}\right)^{2},$$
$$\gamma_{\text{in}} = \sqrt{\mu_{\text{in}}}\left(\frac{1}{\sqrt{d_o - 0.5}} + \frac{1}{\sqrt{d_h - 0.5}}\right)^{\frac{1}{3}}.$$

Second, we let

$$W_{\text{in}}^2 = W_{\text{in}}^1 \cdot \frac{\sigma_{\text{in}}^{\max}}{3\gamma_{\text{in}} + \mu_{\text{in}}}. \tag{A.31}$$

Then we have

$$\|W_{\text{in}}^2\| = \|W_{\text{in}}^1\| \cdot \frac{\sigma_{\text{in}}^{\max}}{3\gamma_{\text{in}} + \mu_{\text{in}}}. \tag{A.32}$$

Last, we obtain W_{in} as follows

$$W_{\text{in}} = \begin{cases} W_{\text{in}}^2 \cdot \frac{\sigma_{\text{in}}^{\max}}{\|W_{\text{in}}^2\|}, & \text{if } \|W_{\text{in}}^2\| \geq \sigma_{\text{in}}^{\max}, \\ W_{\text{in}}^2, & \text{if } \|W_{\text{in}}^2\| < \sigma_{\text{in}}^{\max}. \end{cases} \tag{A.33}$$

Therefore, we have

$$S_{\text{in}} = \|W_{\text{in}}\| = \begin{cases} \sigma_{\text{in}}^{\max}, & \text{if } \|W_{\text{in}}^2\| \geq \sigma_{\text{in}}^{\max}, \\ \|W_{\text{in}}^2\|, & \text{if } \|W_{\text{in}}^2\| < \sigma_{\text{in}}^{\max}. \end{cases} \tag{A.34}$$

If $S_{\text{in}} < \sigma_{\text{in}}^{\max}$, then we have

$$S_{\text{in}} = \|W_{\text{in}}^2\| = \|W_{\text{in}}^1\| \cdot \frac{\sigma_{\text{in}}^{\max}}{3\gamma_{\text{in}} + \mu_{\text{in}}} = (T_{\text{in}} \cdot \gamma_{\text{in}} + \mu_{\text{in}}) \cdot \frac{\sigma_{\text{in}}^{\max}}{3\gamma_{\text{in}} + \mu_{\text{in}}}$$

In this case,

$$T_{\text{in}} = K_{\text{in}} S_{\text{in}} - \frac{\mu_{\text{in}}}{\gamma_{\text{in}}},$$

where

$$K_{\text{in}} = \frac{3\gamma_{\text{in}} + \mu_{\text{in}}}{\gamma_{\text{in}} \sigma_{\text{in}}^{\max}}.$$

On the other hand, if $S_{\text{in}} = \sigma_{\text{in}}^{\max}$, then we have

$$\begin{aligned}\Pr\left(S_{\text{in}} = \sigma_{\text{in}}^{\max}\right) &= \Pr\left(\|W_{\text{in}}^2\| \geq \sigma_{\text{in}}^{\max}\right) \\ &= \Pr\left((T_{\text{in}} \cdot \gamma_{\text{in}} + \mu_{\text{in}}) \cdot \frac{\sigma_{\text{in}}^{\max}}{3\gamma_{\text{in}} + \mu_{\text{in}}} \geq \sigma_{\text{in}}^{\max}\right) \\ &= \Pr\left(T_{\text{in}} \geq 3\right) = 1 - F_{T_{\text{in}}}(3),\end{aligned}$$

where $F_{T_{\text{in}}}(\cdot)$ is cumulative distribution function (CDF) of T_{in}. Therefore, we can obtain

$$f_{S_{\text{in}}}(a) = \delta\left(a - \sigma_{\text{in}}^{\max}\right) \cdot \left[1 - F_{T_{\text{in}}}(3)\right] + \mathbb{1}\left(a < \sigma_{\text{in}}^{\max}\right) \cdot K_{\text{in}} f_{S_{\text{in}}}\left(K_{\text{in}} a - \frac{\mu_{\text{in}}}{\gamma_{\text{in}}}\right), \tag{A.35}$$

where $\mathbb{1}(\cdot)$ is the indicator function.

Similarly, we can derive $f_{S_{\text{rec}}}(\cdot)$ as

$$f_{S_{\text{rec}}}(b) = \delta\left(b - \sigma_{\text{rec}}^{\max}\right) \cdot \left[1 - F_{T_{\text{rec}}}(3)\right] + \mathbb{1}\left(b < \sigma_{\text{rec}}^{\max}\right) \cdot K_{\text{rec}} f_{S_{\text{rec}}}\left(K_{\text{rec}} b - \frac{\mu_{\text{rec}}}{\gamma_{\text{rec}}}\right), \tag{A.36}$$

where

$$\begin{aligned}\mu_{\text{rec}} &= 4(d_h - 0.5), \\ \gamma_{\text{rec}} &= \sqrt{\mu_{\text{rec}}}\left(\frac{2}{\sqrt{d_h - 0.5}}\right)^{\frac{1}{3}}.\end{aligned}$$

A.5 Proof of Theorem 2.7

We can rewrite Equation (A.9) as

$$\eta_{\max,t}^2 \leq 4 \sup_{\theta' \in \Theta} \inf_{\theta \in \Theta} \mathbb{E}_{\sigma}\left[\left(\mathcal{B}Q_{\theta'} - Q_{\theta}\right)^2\right] + C \cdot \log N_{\epsilon,t}^{\text{ext}} + C', \tag{A.37}$$

where

$$C = C_1 \cdot \frac{V_{\max}^2}{T} = \left(8\sqrt{2T} + \frac{256}{V_{\max}}\right) \cdot \frac{V_{\max}^2}{T},$$

$$C' = C_2 \cdot V_{\max} \cdot \epsilon = \left(16 + 4\sqrt{2T} + 36\right) \cdot V_{\max} \cdot \epsilon.$$

Let $R_{\max}$ be the maximum reward value, and the maximum Q-value is written as

$$V_{\max} = \sum_{t=1}^{\infty} \gamma^{t-1} R_{\max} = \frac{R_{\max}}{1-\gamma}.$$

Therefore, the optimal Q-function Q^* is contained in a ∞-norm ball in $\mathbb{R}^{d_y}$ centered at the origin with radius $V_{\max}$, which is denoted by $B_\infty^{d_y}(V_{\max})$. Then the upper-bound of the squared bias term in Equation (A.37) is written as

$$\sup_{\theta' \in \Theta} \inf_{\theta \in \Theta} \mathbb{E}_\sigma \left[(\mathcal{B}Q_{\theta'} - Q_\theta)^2 \right] \leq \sup_{Q^* \in B_\infty^{d_y}(V_{\max})} \inf_{\theta \in \Theta} \mathbb{E}_\sigma \left[(Q^* - Q_\theta)^2 \right]. \tag{A.38}$$

From Equation (A.16), we can obtain that the volume of the set of y_t from RNN/ESN is lower bounded by $\mathrm{vol}\left(B^{d_y}(\epsilon)\right) \cdot N_{\epsilon,t}^{\mathrm{ext}}$, so the largest Euclidean ball that can be contained in the set of y_t has radius $\epsilon \left(N_{\epsilon,t}^{\mathrm{ext}}\right)^{\frac{1}{d_y}}$, which is denoted by $B_2^{d_y}\left(\epsilon \left(N_{\epsilon,t}^{\mathrm{ext}}\right)^{\frac{1}{d_y}}\right)$. Therefore, we have

$$\sup_{Q^* \in B_\infty^{d_y}(V_{\max})} \inf_{\theta \in \Theta} \mathbb{E}_\sigma \left[(Q^* - Q_\theta)^2 \right] \leq \sup_{Q^* \in B_\infty^{d_y}(V_{\max})} \inf_{Q \in B_2^{d_y}\left(\epsilon \left(N_{\epsilon,t}^{\mathrm{ext}}\right)^{\frac{1}{d_y}}\right)} (Q^* - Q)^2 \tag{A.39}$$

Since the maximum ℓ_2 distance from the origin to $B_\infty^{d_y}(V_{\max})$ is $\sqrt{d_y} V_{\max}$, we have

$$\sup_{Q^* \in B_\infty^{d_y}(V_{\max})} \inf_{Q \in B_2^{d_y}\left(\epsilon \left(N_{\epsilon,t}^{\mathrm{ext}}\right)^{\frac{1}{d_y}}\right)} (Q^* - Q)^2 \leq \max\left(0, \sqrt{d_y} V_{\max} - \epsilon (N_{\epsilon,t}^{\mathrm{ext}})^{\frac{1}{d_y}}\right)^2. \tag{A.40}$$

Therefore, we can obtain

$$\eta^2_{\max,t} \leq 4 \max\left(0, \frac{\sqrt{d_y} R_{\max}}{1-\gamma} - \epsilon (N^{\text{ext}}_{\epsilon,t})^{\frac{1}{d_y}}\right)^2 + C \cdot \log N^{\text{ext}}_{\epsilon,t} + C'. \tag{A.41}$$