


Article

Wireless Lan Performance Enhancement Using Double Deep Q-Networks

Khizra Asaf ^{1,†}, Bilal Khan ^{1,†}  and Ga-Young Kim ^{2,*}

¹ Department of Computer Science, National University of Computer and Emerging Sciences, Chiniot-Faisalabad Campus, Chiniot 35400, Pakistan; khizraasaf94@gmail.com (K.A.); khan.bilal@nu.edu.pk (B.K.)

² Faculty of General Education, Kangnam University, Yongin-si 16979, Korea

* Correspondence: dolga2000@kangnam.ac.kr

† These authors contributed equally to this work.

Abstract: Due to the exponential growth in the use of Wi-Fi networks, it is necessary to study its usage pattern in dense environments for which the legacy IEEE 802.11 MAC (Medium Access Control) protocol was not specially designed. Although 802.11ax aims to improve Wi-Fi performance in dense scenarios due to modifications in the physical layer (PHY), however, MAC layer operations remain unchanged, and are not capable enough to provide stable performance in dense scenarios. Potential applications of Deep Learning (DL) to Media Access Control (MAC) layer of WLAN has now been recognized due to their unique features. Deep Reinforcement Learning (DRL) is a technique focused on behavioral sensitivity and control philosophy. In this paper, we have proposed an algorithm for setting optimal contention window (CW) under different network conditions called DRL-based Contention Window Optimization (DCWO). The proposed algorithm operates in three steps. In the initial step, Wi-Fi is being controlled by the 802.11 standards. In the second step, the agent makes the decisions concerning the value of CW after the TRAIN procedure for the proposed algorithm. The final phase begins after the training, defined by a time duration specified by the user. Now, the agent is fully trained, and no updates will be no longer received. Now the CW is updated via the OPTIMIZE process of DCWO. We have selected total network throughput, instantaneous network throughput, fairness index, and cumulative reward, and compared our proposed scheme DCWO with the Centralized Contention window Optimization with DRL (CCOD). Simulation results show that DCWO with Double Deep Q-Networks (DDQN) performs better than CCOD with (i) Deep Deterministic Policy Gradient (DDPG) and (ii) Deep Q-Network (DQN). More specifically, DCWO with DDQN gives on average 28% and 23% higher network throughput than CCOD in static and dynamic scenarios. Whereas in terms of instantaneous network throughput DCWO gives around 10% better results than the CCOD. DCWO achieves almost near to optimal fairness in static scenarios and better than DQN and DDPG with CCOD in dynamic scenarios. Similarly, while the cumulative reward achieved by DCWO is almost the same with CCOD with DDPG, the uptrend of DCWO is still encouraging.

Keywords: WLAN; 802.11; DRL; DDQN; DCWO



Citation: Asaf, K.; Khan, B.; Kim, G.-Y. Wireless Lan Performance Enhancement Using Double Deep Q-Networks. *Appl. Sci.* **2022**, *12*, 4145. <https://doi.org/10.3390/app12094145>

Academic Editor: Amadeo Benavent-Climent

Received: 11 March 2022

Accepted: 15 April 2022

Published: 20 April 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Wireless networks have seen increasing and continuous popularity resulting in increased data traffic over all the networks [1]. Wi-Fi networks have experienced incredible growth concerning traffic consumption. Due to the increased use of mobile devices, it is expected that 63% of the mobile data traffic will be shifted to Wi-Fi networks by the year 2021 [2].

IEEE 802.11 is a set of progressively improved standards to continue developing further modifications to overcome the limitations once discovered. Due to flexibility,

manageability, and popularity, IEEE 802.11 technologies can be used in different scenarios and become the right candidate for many applications with several requirements. With the ever-increasing demand for wireless traffic and quality services, WLANs have become one of the major wireless networks that affect human life entirely. WLAN development has led to an increasing number of wireless devices, such as access points and mobile nodes. Not only is the number of these devices increasing rapidly, but the communication speed required for each device is also increasing. IEEE 802.11 based WLAN has evolved significantly to provide a better performance and throughput [3] in recent years. The main factors behind 802.11 WLAN deployments' success were high performance and low technological cost. However, when these devices rarely operate in densely populated areas and share limited channels, and sometimes the same channel, performance will be affected due to the overlap of the common channel of its neighbors. Therefore, the increasingly intensive deployment of access points and nodes may not help to improve network performance.

Significance of IEEE 802.11ax

WLANs are facing significant expansion in internet-centric applications. High-tech markets are using WLANs, and their deployment is growing rapidly in private and public areas, such as cafes, shopping malls, hotels, bus/train stations, restaurants, airports, etc. Due to the exponential expansion in the usage of Wi-Fi networks, it is necessary to study its use pattern in dense environments for which the IEEE 802.11 standard was not specially designed. As the number of orthogonal channels available for IEEE 802.11 is limited, the status of OBSS in WLAN-based networks is redundant. Collision avoidance approaches tend to decrease network performance and increase transmission delays; however, all medium is never used despite the acceptable collision potential.

While the present IEEE 802.11 standards have been developed to enhance the overall peak performance of several nodes in the network, appropriate mitigation of the increase in interference encountered has not been tackled yet. In addition, the channel access method is highly protective and results in reduced spatial reuse. This standard aims to use technologies that will enhance the physical bit rate, reduce the data rate, and enhance spectrum reuse by enabling highly efficient multiple users.

Due to the exponential growth in the use of Wi-Fi networks, it is necessary to study its use pattern in dense environments for which the legacy IEEE 802.11 standard was not specially designed. Considering the performance issues affecting WLAN in dense environments, a study group was launched in May 2013 called High-Efficiency WLAN (HEW) within the working group of IEEE 802.11 [4]. HEW activities considered spectral efficiency to improve system performance in high-density environments, including a number of APs and non-AP nodes. Following the success of the Project Approval Request (PAR) and Criteria for Standard Development (CSD), the group was promoted to status of team to design a new standard; IEEE 802.11ax, in July 2014.

IEEE 802.11ax standard is the latest contribution to the innovation journey [5,6]. The Wi-Fi Alliance established a new naming standard with previous known as Wi-Fi 4 (802.11n) and Wi-Fi 5 (802.11ac). The development of the 802.11ax amendment began in 2013 when a team of technical experts met to discuss the challenges that Wi-Fi may face in the years to come. Wi-Fi was threatened by the possibility of being the victim of its success because of its global use. With more devices, Wi-Fi would experience increased interference and poor performance. The working group discussed data and solutions to the problem and finally described the requirements for Wi-Fi 6, also known as Highly Efficient WLANs (HEW). This new generation of Wi-Fi will be smart enough to enable dense and complete wireless environments in the future; 802.11ax takes advantage of the strengths of 802.11ac, with the added scalability and flexibility that allow existing and new networks to operate next-generation applications.

So, 802.11ax takes advantage of the strengths of 802.11ac, with the added scalability and flexibility that allows existing and new networks to operate next-generation applica-

tions. This new generation of Wi-Fi will be smart enough to enable dense and complete wireless environments in the future. However, the channel access method remains the same to ensure backward compatibility. This approach is robust; however, it can lead to inefficient operation in dense scenarios because of sudden increases and decreases in the Contention Window. This problem is described in detail in Section IV of the paper.

Artificial Intelligence (AI) can solve complex problems without the need for explicit programming. Motivated by its applications to several beneficial tasks like image recognition, research community has encouraged applications of AI in wireless communication [7]. AI practices for network operations, automation, and management deal with designing and implementing AI practices to enhance how we deal with networking nowadays. This growth towards better design opportunities and the increased complexity in networks and network-based applications have expanded the demand for better automation of network in agile infrastructures. Researchers in communication networks are applying AI techniques to improve network design, management, and to enhance performance leading to even more automation in network functions. Machine learning (ML) applies AI by using different algorithms to analyze the data, learn from the data, and make accurate predictions and decisions about real-world applications. Deep learning (DL) is an ML execution technology that empowers ML to perform and extends the scope of AI. DL involves huge amounts of training data and considerable computational capability. In recent years, data volumes have grown while computing power costs have decreased significantly, allowing deep learning applications to explode. Given the uncertainties and dynamics of wireless networking environments, traditional approaches require complete and perfect systems knowledge and are inefficient. RL has proven to be a practical tool for solving real-time decision-making challenges. Reinforcement Learning (RL) is a field of ML that focuses on learning via interaction. Deep Reinforcement Learning (DRL) combines the architecture of DL(NN) and RL [8,9]. In DRL, NNs act as function approximators and can be used in RL to estimate value function or policy. DRL has been intended to conquer these shortcomings, as it has the potential for managing dynamic systems on a large scale. Unlike traditional ML, RL does not provide instant results; instead, only a short-term reward is observed [10]. DRL on the other hand, is a potential solution [11,12].

Predicting the optimal Contention Window (CW) value for better performance of Wi-Fi networks is important for avoiding collision and enhancing the performance of Wi-Fi. Current approaches to improving MAC layer channel access and collision control have been found to have some research gaps, mainly when used with real-time applications. This research focuses on the effective use of a Deep Reinforcement Learning (DRL) technique i.e., DDQN for enhancing the performance of WLAN. For this, we have proposed an algorithm named DRL-based Contention Window Optimization (DCWO) for setting an optimal contention window (CW) under different network conditions. The main research objectives are:

- Address the open problems in the future deployable 802.11 networks (IEEE 802.11ax) using DRL.
- Enhance WLAN performance by predicting the values of CW correctly using DRL.

2. Deep Reinforcement Learning (DRL)

DRL differs from the traditional techniques and is the latest, targeted research in DL. In situations such as those that require the application of DRL, the state space and size, in general, can be quite large, and the agent can take a long time to learn adequate information about the environment and determine which actions can yield the best immediate or total rewards. In these situations, opportunities for exploration may be swept away, and the agent may get caught in exploring and estimating action-state combinations with fairly higher values. This results in the overestimation of Q values for some of the combinations. The overestimation problem might become huge if the actions are taken based on a Q network whose values are not updated frequently.

2.1. DQN

The concept of Double Q-Learning deals with the overestimation through a process-target analysis of the selection and evaluation of actions. Though not completely separate, the target network in the architecture of DQN offers a natural candidate for the role of the second value without the need for any additional networks [13]. The DQN-based model uses online and target networks to stabilize the performance. The agent is not familiar with the environment at the start of the training. Hence, choosing the maximum value of Q as the best action results in overestimating action values. Due to these issues, noises from the expected Q value will lead to significant positive biases, and hence the learning process will become complex. Implementing two DQN is the solution to this problem.

2.2. DDQN

DDQN combines DQN and Double Q-learning to solve the problems of overestimation for Q values. DQN and Double Q learning maintain two weights sets; however, the use of both sets is different. For both the algorithms, the online network is updated at every step by the square error of Q value as well as the target value. In the DQN algorithm, the target network selects and evaluates the action, while in Double Q learning, both the online network and target network are used in the target value function. One selects the best action and the other to get the value of Q.

DDQN keeps the target and online networks in DQN but uses the Double Q learning target feature with both the networks. It is still compatible with all the methods of DQN, i.e., target network and experience replay. For reusing the online network, there is no need for additional weights. With Double Q like target function, the value of Q is unlikely to be overestimated. DDQN estimates the greed policy as per the online network and uses the target network to calculate its value. In DDQN, at each step, the value of all the action value combinations of all possible actions in the given state is read from the constantly updated online network. Then argmax is applied to all state action values for those possible actions [14]. The value of the state action combination increases the value to the maximum, and that specific action is selected. However, the resultant value of a set of these specific actions is taken from the target network to update the online network. DDQN can simultaneously overcome the overestimation problem while avoiding the unpredictability of the target values.

2.3. Working of DDQN

Double DQN (DDQN) provides two approximations of value Q, characterized by two Neural Networks (NN's), each NN being updated by the other NN for the next state. Originally, the DQN calculates both the expected Q value and the target Q value [15]. This can lead to differences between the two [16]. So, with DDQN, we have one of the Q networks which estimates the target Q values, which will later be called the target network. The other network, called local network or prediction network, will be used to estimate the expected Q values. The destination network will have the same local network architecture and the same initial parameters.

The architecture of DQN provides a real candidate for the second value function devoid of the introduction of any other networks. Therefore, DDQN uses the DQN network to choose the best action to make a decision on the next transition. It uses the target network to estimate the selected action target value at the next state. To update the destination network, we copy the parameters of the local network, and this is done at each iteration. The overall effect of DDQN reduces the overestimation effect, which leads to better performance over standard DQN and more stable training as it keeps the target network more or less stable.

In DQN, the value of Q is calculated with a reward added to the maximum value of Q for the next state. If the value of Q estimates a high number for a given state everytime, the value obtained from the neural network output for that particular state will be incremented. Each output value of the neuron will increase more and more up to the high difference

between every output value. If we say that action a for the state has a higher value than the action b , then every time, action a will be chosen. Consider a case where for a memory experience, action b is a better choice than action a . Therefore, as the NN is trained to give the action a much higher value when in the given state, training the network to know that action b is the best action in some circumstances will be problematic. To deal with the difference in output values, a secondary model is used. This model is the version of the primary model of the last episode and, as the difference in values of the second model is lesser as compared to the main model, this second model will be used to obtain the Q value. This is how the Q value is calculated in a DDQN. The index of the highest Q value for main model is determined, and this index is used to attain the action from second model.

3. Related Work

Wi-Fi technology has a vital influence on how the broadband access is shared in home and business networks. However, High Efficiency 802.11 Wireless Local Area Network Task Group (TG) visualizes congested scenarios as one of the main future challenges for WLAN protocols. To deal with the problem of WLAN performance degradation in dense scenarios, researchers from the field of networking have proposed different solutions. Motivated by the success of the application of machine learning to wireless networks, recent researches have used different machine learning techniques for enhancing WLAN performance.

The physical medium is the main reason behind performance drop and errors in most cases. Therefore, Kremer et al. have introduced a method for the predictive estimation of wireless link performance using two machine learning techniques [17]. For this purpose, a measurement bench has been designed to precisely control the level of noise in a unidirectional Wi-Fi link in the safe environment of the echoic room. Furthermore, the study has also presented an analysis of the relationships between PHY WI-FI connection parameters and performance parameters on the IP layer. SVR and k-NN (nearest neighbors) algorithms were selected to estimate the performance drops at a one-second scale. Finally, the authors have assessed SNR's importance in predicting the throughput of WiFi. This study, however, focuses on the physical layer of wireless LAN instead of the MAC layer.

Most mobile devices today are equipped with many wireless interfaces. This led to developing research interest in Device to Device (D2D) communication. Testa et al. have proposed a new method to accurately estimate the active user equipment using ML techniques and only client-side information without the need for change in the standard communication protocol [18]. Information about the number of receiving nodes is vital in some cases, such as the user wants an accurate prediction of transmission approximate time of arrival when downloading a file. To achieve this, the time required to transfer the first part of the file from an access point to the receiving node is calculated, along with other information. The retrieved information then analyzed using ML techniques. In the paper, classification results of Naive Bayes (NB), Linear Support Vector Machines (SVM-L), Radial Support Vector Machines (SVMR), and k-Nearest Neighbor (kNN) are presented. This research work, however, considers Device to Device communication rather than the infrastructure based Wireless LAN.

Edalat et al. presented a new mechanism for estimating the performance of future network, based on the previous conditions of network, which is presented in [19]. This approach for estimating network performance is called SENSE (Smart Experts for Network State Estimation). It makes use of a simple but effective algorithm that combines a machine learning technique fixed share with EWMA (Exponentially Weighted Moving Average). SENSE, however, is only applicable to accurately estimate the RTT (return trip time) of the TCP (transmission control protocol) of the transport layer. In [20], Deep Reinforcement Learning-based MAC protocol for wireless networking, also known as a Deep-reinforcement Learning Multiple Access (DLMA) is proposed. DLMA learns an optimal channel access strategy to maximize throughput and fairness. DLMA, however, is a customized MAC protocol which does not follow the rules of binary exponential backoff as stated in IEEE 802.11. Correct setting of the Contention Window (CW) value has a massive

impact on the efficacy of Wi-Fi networks. Witold Wydmański and Szymon Szott proposed an approach to control the value of CW that takes advantage of the principles of Deep Reinforcement Learning (DRL) for learning the accurate setting under various network conditions [21]. The proposed method, Centralized Contention window Optimization with DRL (CCOD), supports two trainable algorithms offering near-optimal efficiency. The performance of CCOD is exhibited under 802.11ax using Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG). DQN is a showcase algorithm, while DDPG is an advanced method expected to offer higher network performance, particularly in dense areas. In addition, the authors also explained how time series analysis could be applied to recurrent neural networks of both DRL methods. In [22], the authors have designed an intelligent node capable of dynamically adapting a minimum contention window parameter to increase utility at the network level. For this work, authors have adopted Reinforcement Learning framework based on DQN architecture. It learns the optimal minimum contention value from the local observations. This research, however, unrealistically assumes that some nodes in network may deviate from the standard behavior and may choose a reduced Minimum Contention Window unfairly to acquire more channel access.

From the literature review, we conclude that the problem of optimization of CW has provided an opportunity to exhibit DRL features and successful applications of DRL in WLAN research. Recent researchers have focused on the used Q-networks and Deep Q-networks (DQN); however, these techniques suffer overestimation problems. To deal with the overestimation problem of DQN and to enhance WLAN performance, we have extended the work presented in [21] and proposed a scheme that uses DDQN to improve the performance of WLAN in dense scenarios. DDQN tackles the problem of DQN by decoupling action selection from the action evaluation. Main Neural Network decides best next action among all the available next actions and target Neural Network evaluates this action to know its Q-value thus providing better final policies.

4. Problem Statement

Wi-Fi has appeared as a significant technology to mitigate the outbreak of data on cellular networks, offering many benefits such as ease of setup, high data rates, and free internet access. Research results show there will be around 5.3 billion Wi-Fi users globally by the year 2023 and the volume of mobile traffic will also increase exponentially [23]. Dense implementation scenarios will become more widespread than today, and users will likely anticipate a high level of Wi-Fi. IEEE 802.11ax aims to increase the efficiency of Wi-Fi [24]. However, an efficiency-related aspect, the channel access method remains the same to ensure backward compatibility. This method is Carrier Sensitive Multiple Access/Collision Avoidance (CSMA/CA). This method, however, is robust to the dynamic network environment leading to incompetent and ineffective operation, specifically in dense scenarios [25,26].

It is a simple approach; however, it suffers from some limitations because of exponential increase in CW and sudden decrease in CW to its lowest value [27] as shown in Figure 1 and will be further explained in the next subsection.

4.1. Exponential Increase in CW

A double increase in CW in the event of a collision decreases the access time to the channel. As the CW doubles due to collision, the backoff values of the node increase and as a result, more time is spent sensing the channel than accessing it. Failure to optimize the channel access time reduces throughput. Moreover, collisions are assumed based on the lack of CTS or ACK, thus contributing to packet loss, and CW doubling based on this assumption is not a good idea. Additionally, with the increase in active nodes, doubling the size of CW to decrease the potential for collision becomes less effective. Figure 1 shows the exponential increase and sudden decreases in CW.

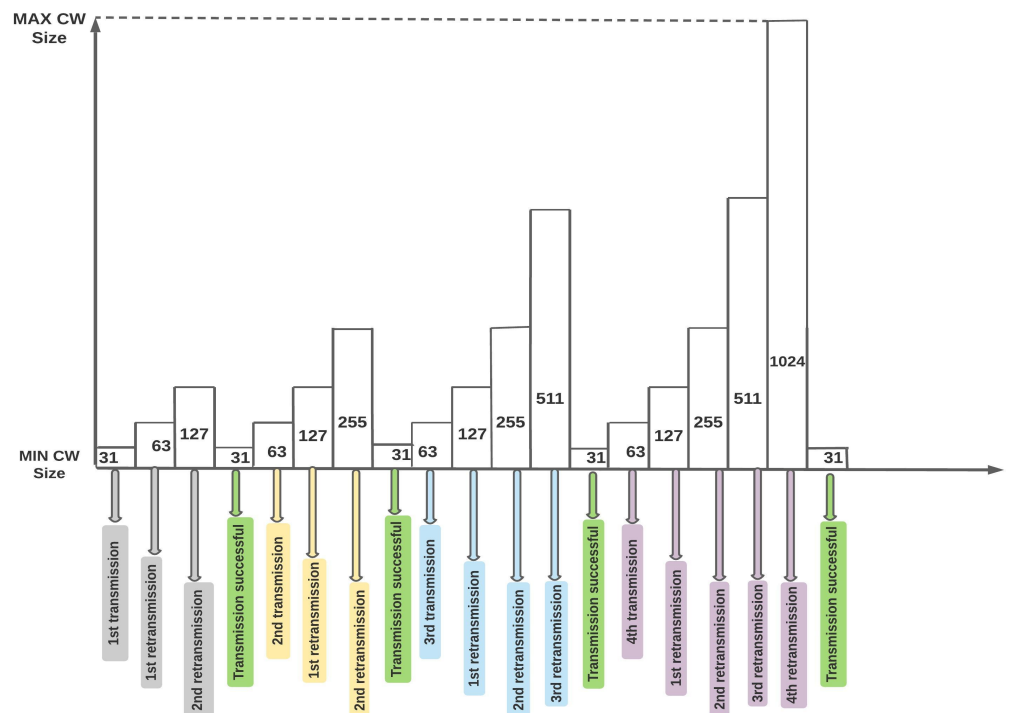


Figure 1. Variation in the size of CW due to collision and successful transmission.

4.2. Sudden Decrease in CW

Resetting the CW to its lowest value after a successful submission will destroy fairness. A node with successful transmission will have a better chance of channel access because of smaller CW as compared to a node that has had a collision. Concerning the analytical models of IEEE 802.11, most models assume a collision probability independent of the history of the node's transmission, which results in an inaccurate estimate of the possibility that a node transmits in a random period, thus resulting in an inaccurate analysis of throughput. This sudden increase and resetting of CW results in degraded performance [28]. In a dense scenario, resetting of CW to its minimum size can lead to high collisions probability resulting in poor network performance. Similarly, in the case of a small network, sudden increase in the size of CW can cause an unnecessary delay while accessing the channel. Moreover, doubling the CW size after every collision will take time and have to wait for a long time before counting down to zero, resulting in starvation. Furthermore, the Binary Exponential Backoff (BEB) algorithm always prefers the last node, which transmitted successfully. This can lead to an intensification of the fairness problem, resulting in the degradation of system performance. Frequent retransmissions also caused a lot of unnecessary energy waste.

This problem can be explained with a simple scenario. Figure 2 shows a simple scenario where four nodes N1, N2, N3, and N4 are sensing the channel for data transmission using BEB.

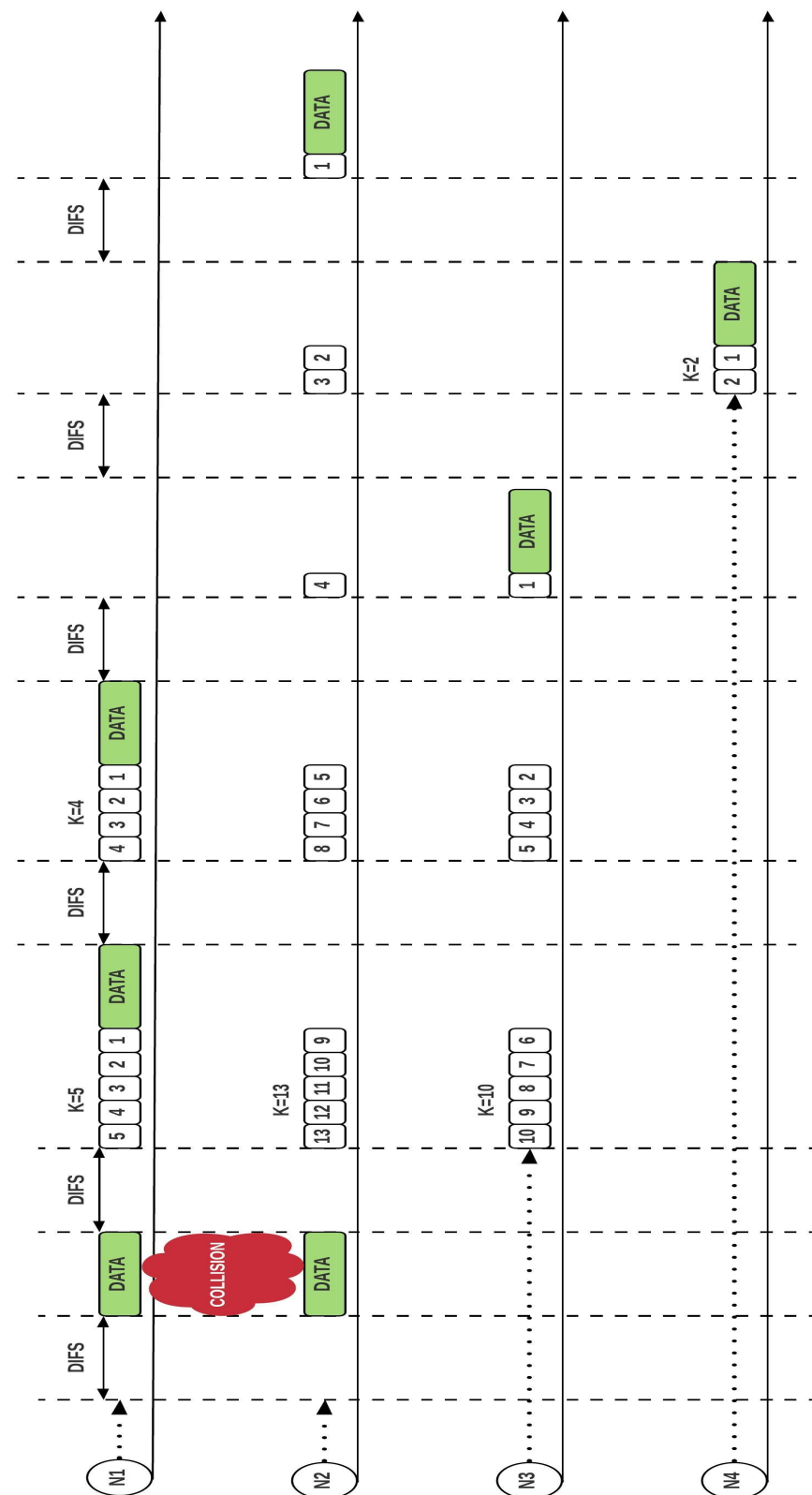


Figure 2. Four nodes competing for the channel using binary exponential backoff based distributed coordinated function.

Both nodes N1 and N2, send data at the same time which causes collision. When the collision occurs, CW size is doubled which forces the collided nodes to wait for additional

time before transmission attempt. N1 selects the value of the backoff counter (k) as 5, while N2 selects $k = 12$. N3 also senses the channel for data transmission and selects $k = 10$. N1 sends the data successfully and again decides $k = 4$ to send another packet of data. The backoff value chosen randomly by Node N1 in the second round is smaller than the remaining backoff counter of N2; therefore, N1 will again get the transmission opportunity. Moreover, as the remaining backoff of N3 is smaller than the remaining backoff of N2, N3 will get the opportunity of transmission. Now, N4 senses the channel for transmission and selects $k = 2$, which is smaller than the remaining backoff of N2 (i.e., $k = 2$), and N4 will get the transmission opportunity. The larger the size of the CW a node selects, the lower the opportunity to the channel it gets. This unfairness seriously compromises the performance of the network.

To tackle the problems mentioned above, we have proposed an efficient method of setting the optimal value of CW using the principles of DRL. Using DRL techniques, individual devices and entire networks can be characterized from a high-level perspective that relies on the data extracted directly from the measurements. Our proposed scheme will provide better results regarding network performance.

5. Proposed Scheme

As discussed in the previous Section, opting for a smaller value of CW results in more collisions. While opting for a large value of CW can result in excess downtime and additional delays. In both the cases, the channel will not be used effectively. Hence, it is important that the CW value should be adjusted, keeping in mind the actual level of contention in the channel.

With the presence of a massive number of advanced network devices, CW optimization can now be examined using DRL methods. DRL is well suitable for the problem of optimizing wireless networks as it deals with smart software agents that take actions in a specific environment in order to maximize the reward [29]. Ref. [21] showed that DRL could be applied successfully to a CW optimization problem as both algorithms, DQN and DDPG, provided close to optimal efficiency. However, DDPG performs slightly better than the DQN.

The main problem with Q-Learning is the slow pace of convergence owing to its iterative nature [30]. Moreover, it does not depend on any previous information while experiencing a new state. With DRL, a Deep Neural Network (DNN) estimates the state-action value function. The NN of Deep Q-Networks (DQN) is trained to reduce the prediction error caused by the loss function [31]. To further optimize DQN for the context of WLAN, we will use DDQN in this paper.

We decided to use DDQN for two main reasons.

- DQN estimates a set of strongly correlated values solved by DDQN
- DQNs tend to be overly optimistic. It will overreact to you being in this situation, even if it only happened due to a statistical error, and DDQN solves this problem.

The overestimation problem means that the estimated value function in DQN is greater than the actual value function. The root of this problem lies in the Q-Learning maximization process. By calculating the target Q , the maximum value of Q is obtained in the next state. For authentic approaches, the specific action that maximizes the Q value is not always selected because real strategies are stochastic strategies. Thus, choosing the maximum Q value of the action here often results in a target value more significant than the actual value. DDQN extends DQN by reducing the overestimation of the Q function. DDQN suggests using two value functions that lead to two sets of weights to avoid overly optimistic reward estimates. One weight set is used to identify the action, while the other weight set is used to assess its reward [13]. In this way, DDQN enhances the strength and performance of the learned model.

In order to deal with the limitations of CSMA/CA, we have proposed an algorithm named DRL Based Contention Window Optimization (DCWO) inspired by CCOD algorithm [21]. (see Algorithm 1).

Algorithm 1 DRL Based Contention Window Optimization (DCWO) [21]

```

1:  $D \leftarrow \text{dataset}$ 
2: Define  $N_t, N_r$ 
3:  $CP_{obs} \leftarrow$  observed collision probability
4:  $s = \text{state}, a = \text{Previous action}, A = \text{agent}$ 
5:  $CW \leftarrow 31$ 
6: function TRAIN PROCEDURE( $D, CP_{obs}, a$ )
7:    $obs \leftarrow \text{preprocess}(CP_{obs})$ 
8:    $n \leftarrow \text{normalize}(D)$ 
9:    $A.\text{step}(s, a, n)$ 
10:   $a \leftarrow A.\text{act}(s) + \text{noise}$ 
11:   $CW \leftarrow 2^i - 1$ 
12:  return  $CW$ 
13: end function
14: function OPTIMIZE PROCEDURE( $CP_{obs}$ )
15:   $obs \leftarrow \text{preprocess}(CP_{obs})$ 
16:   $a \leftarrow A.\text{act}(s)$ 
17:   $CW \leftarrow 2^i - 1$ 
18:  return  $CW$ 
19: end function

```

The proposed algorithm operates in three steps.

1. In the initial step, Wi-Fi is being controlled by the 802.11 standard. This phase evaluates the history of observed probability of collision in the network.
2. In the second step, selected DRL model, DDQN is trained by maximizing the reward. The agent makes the decisions with regards to the value of CW after the TRAIN Procedure for the proposed algorithm. Pre-processing consists of computing the mean as well as the standard deviation probabilities of observed collisions. For exploration, every action is revised with a noise factor that degrades during the training phase. For DDQN, noise is the possibility of exceeding the action of the agent with a random action. The final phase begins after the training, which is defined by a time duration defined by the user. Now, the agent is fully trained, and updates will no longer be received.
3. In the third phase, the DRL model is deployed in the network. Now the CW is updated via OPTIMIZE Procedure of our proposed algorithm. The flow chart of DCWO is shown in Figure 3.

Applying the DRL algorithm also requires the setting of some key parameters. Performance of DRL is based on the reward discount γ , which is linked to the significance of future rewards on immediate ones. Also, the integration of DL and RL algorithms creates a challenge in the form of a number of new hyperparameters; therefore, every neural network involves a file to configure the learning rate as a coefficient of update. The learning is performed by random gradient descent of the mini-batch, so choosing an appropriate batch size is also very important. The algorithm also splits local and target neural networks to smooth the reward noise. Finally, the algorithm uses a replay buffer that records all the interactions between the environment and agent, serving as a base for sampling mini-batch.

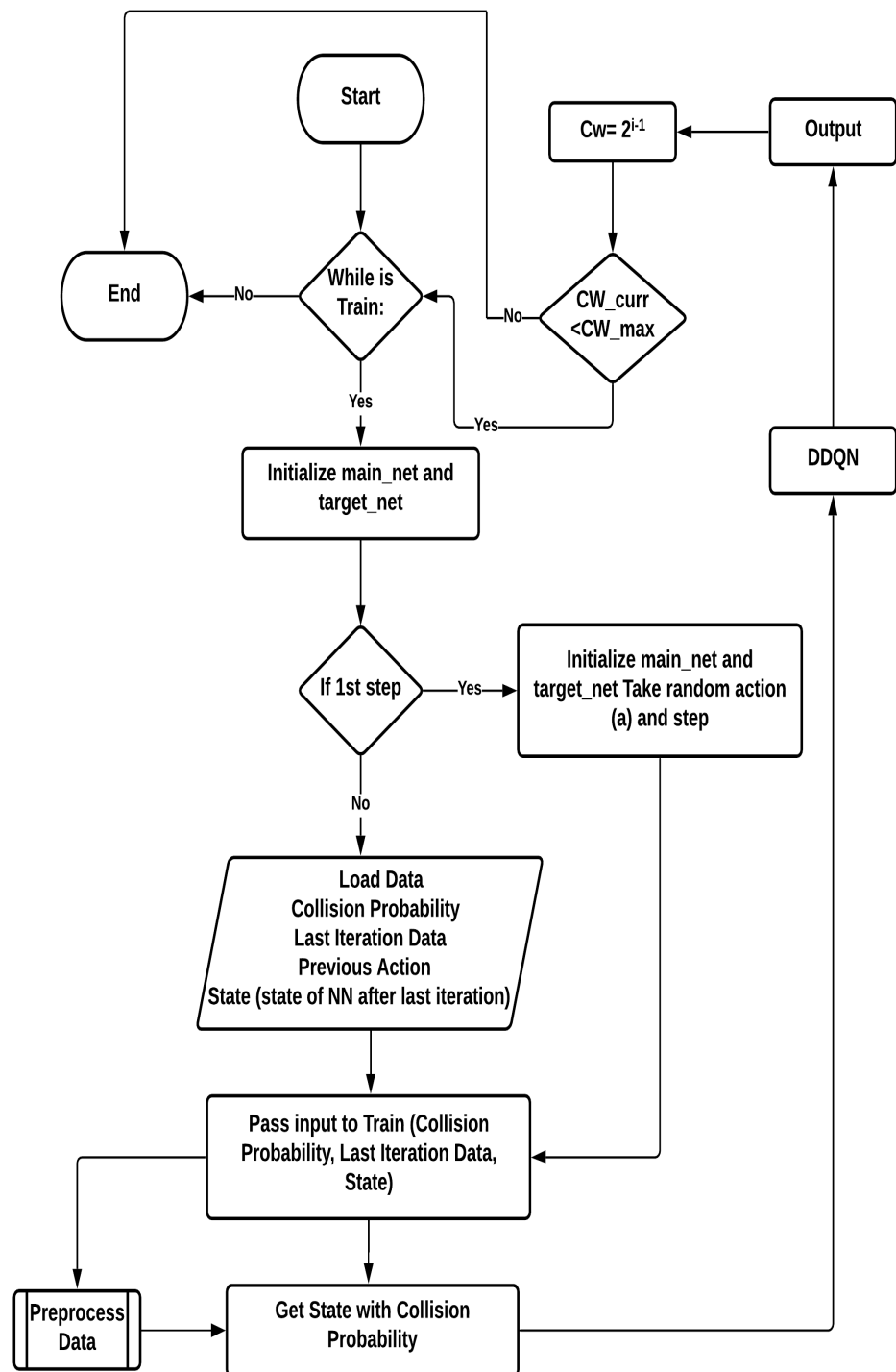


Figure 3. DCWO Flow Chart.

6. Scenario and Methodology

For the experimentation, we have created a scenario example in NS-3, taking the idea from the base paper. The NS-3 scenario for DCWO algorithm consists of maximum of 50 nodes connected to a single AP. Simulations in NS-3 used a simple topology and following settings.

- IEEE 802.11ax
- Single-user transmission

- Error-free radio channels
- 1024-QAM with 5/6 coding rate
- 20 MHz channel
- 1500 B packets
- CBR UDP uplink traffic to a single access point

In the considered topology, the nodes transmit data to AP. The AP calculates the observed probability of collision and selects an optimal CW value. The optimal CW value is calculated considering the network behavior and a new CW value is broadcasted. We have assumed immediate and perfect transfer of information from the state to the agent and also immediate CW configuration at each node.

All the hyperparameters used for DCWO are listed in Table 1.

Table 1. DCWO Hyperparameters.

Parameter	Value
Learning Rate (LR)	5.0×10^{-6}
curr_step	14
Batch Size	32
envStepTime	0.01
Interaction Period	10 ms
Reward Discount (γ)	0.9
History Length (h)	300
Episode Count	15
Scenario	Convergence
SimTime	60
Steps per episode	6300
TAU	0.001
UPDATE EVERY	1
Replay Buffer	18,000

The Neural Network of DDQN is implemented in tensorflow. TensorFlow is an open-source library designed for fast numerical computing. An illustration of our proposed approach is shown in Figure 4. Our proposed approach considers channel, number of nodes, data to be transferred, Contention Window(CW), and collisions, and decides an optimal CW value. Every observation in the scenario is defined as the current probability of collision in the network. This probability is calculated based on the number of successfully received frames and total transmitted frames.

6.1. Performance Metrics

We have evaluated our proposed scheme on the basis of following parameters.

6.1.1. Network Throughput

Network throughput is the amount of traffic flowing from the source towards the destination. It indicates the performance of a network and determines how many data packets have been successfully delivered from source to destination. Low throughput results in poor performance for the end-users. Packet loss, jitter, and latency are the three important factors that affect throughput of a network.

Network throughput is calculated as

$$NetworkThroughput = \frac{N_t}{\Delta t} \quad (1)$$

In Equation (1), N_t is the number of transmitted frames and Δt is the interaction period.

6.1.2. Fairness Index

Fairness metrics determine whether the nodes are getting a fair share of the available resources. Jain's fairness index is one of the most commonly used metrics for fairness evaluation. This is independent of its population size and the values lies in 0 to 1. Jain's fairness index is calculated as

$$f(x_1, x_2, x_3, \dots, x_n) = \frac{\left(\sum_{i=1}^n x_i\right)^2}{n \left(\sum_{i=1}^n x_i^2\right)} \quad (2)$$

In Equation (2), x is normalized throughput and n is the number of nodes. J equals to 1 means the fairest allocation of resources, and all the users can enjoy the resources.

6.1.3. Cumulative Reward

A DRL algorithm is judged by the quality of the policy it finds and the received reward. Rewards are the values in number an agent receives when it acts under certain environmental conditions. This value can be negative or positive, depending on the agent's actions. In RL, the goal is to maximize the cumulative reward rather than the reward received from the current state. The cumulative reward is the sum of all the rewards received so far. It shows the performance of a RL algorithm by plotting the cumulative reward as a function of a number of time frames.

$$CR = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (3)$$

In Equation (3), cumulative reward at every time step t is given by r_{t+1} and γ is the discount factor. r_{t+1} is the reward agent received at time step t when performing an action to go from one state to another state and the discount factor defines the importance of immediate and future rewards and must satisfy $0 \leq \gamma \leq 1$. The agent selects the actions in order to maximize the expected (discounted) return. Reward discount γ is set to refine the agent goal. A value close to 0 emphasizes the importance of immediate rewards, and a value close to 1 means that future rewards are more important.

6.2. Simulation Parameters

DDQN was executed using the hyperparameters listed in Table 2. These hyperparameters were determined through empirical observation by a simulation campaign for better performance. Random grid search is used to select the hyperparameters.

Table 2. DDQN Hyperparameters.

Parameter	Value
Learning Rate (LR)	5.0×10^{-6}
Batch Size	32
Interaction Period	10 ms
Reward Discount (γ)	0.9
History Length (h)	300
Replay Buffer	18,000

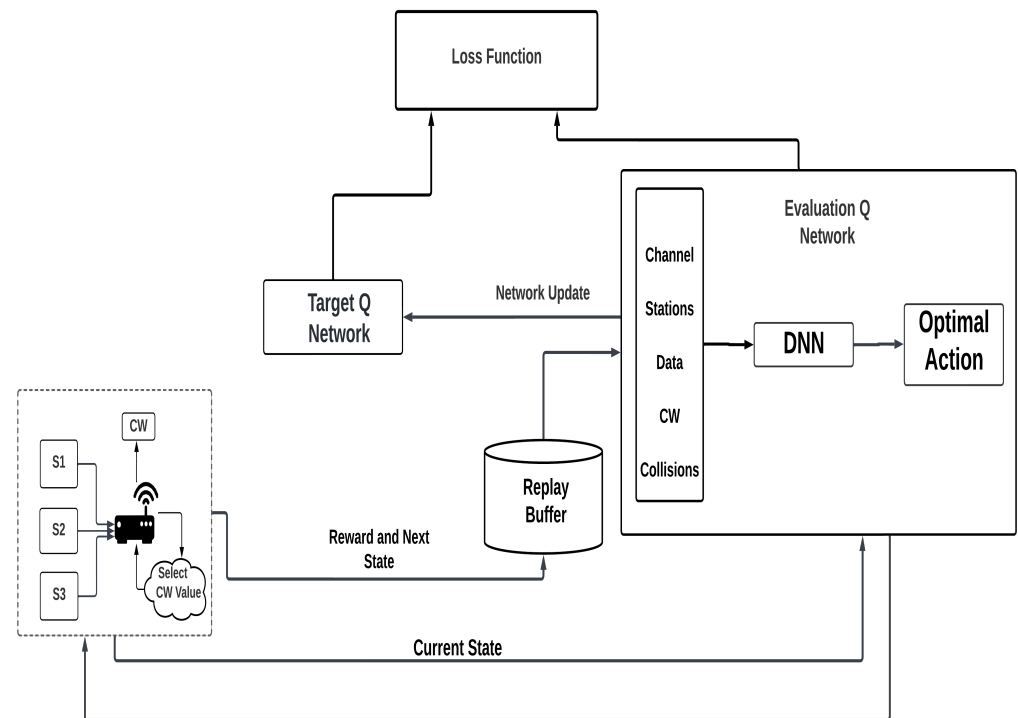


Figure 4. DDQN Approach for DCWO.

7. Results and Discussion

For the implementation of the proposed scheme, we have used NS3-gym. The architecture of the Ns3-gym framework has two modules, OpenAI Gym and the NS-3. OpenAI Gym framework is utilized to deploy agents, while NS-3 serves as the environment. The general interface between NS-3 and OpenAI Gym allows the perfect integration of these two frameworks. The interface manages the life cycle of the NS-3 simulation process, as well as providing state and action information between the Gym agent and simulation environment.

We have compared the results of our proposed scheme DCWO with that of CCOD [21]. DCWO control the value of CW taking advantage of the principles of Deep Reinforcement Learning (DRL) for learning the correct setting under various network conditions. Every experiment was carried out for 15 rounds of 60 s simulations. The first 14 rounds were training phase, and the 15th round was the operational phase. Every simulation consisted of interaction times of 10 ms, between which the proposed algorithm was executed.

The result of DCWO is evaluated in static and dynamic scenarios and are compared with CCOD. Graphs are plotted to observe the overall performance of the proposed algorithm in terms of instantaneous and average network throughput, fairness index and cumulative reward. Moreover, we have also represented the variation of CW as the number of nodes increases over the time.

For network throughput and fairness index, we evaluate the DCWO in static as well as dynamic scenarios and compare the results with those obtained from CCOD. A mean CW value was selected by DDQN in every round of simulation scenario for 25 nodes.

In the static scenario, number of nodes connected to AP were fixed throughout the simulations. In such a scenario, constant CW value is optimal. It can be seen from the Figure 5 that our proposed scheme, DCWO with DDQN performs better than CCOD for both the algorithms i.e., DQN and DDPG, and can optimize the value of CW in static network conditions. Figure 5 shows that DCWO achieves 25% higher throughput compared to CCOD when the number of nodes in the network exceed 30.

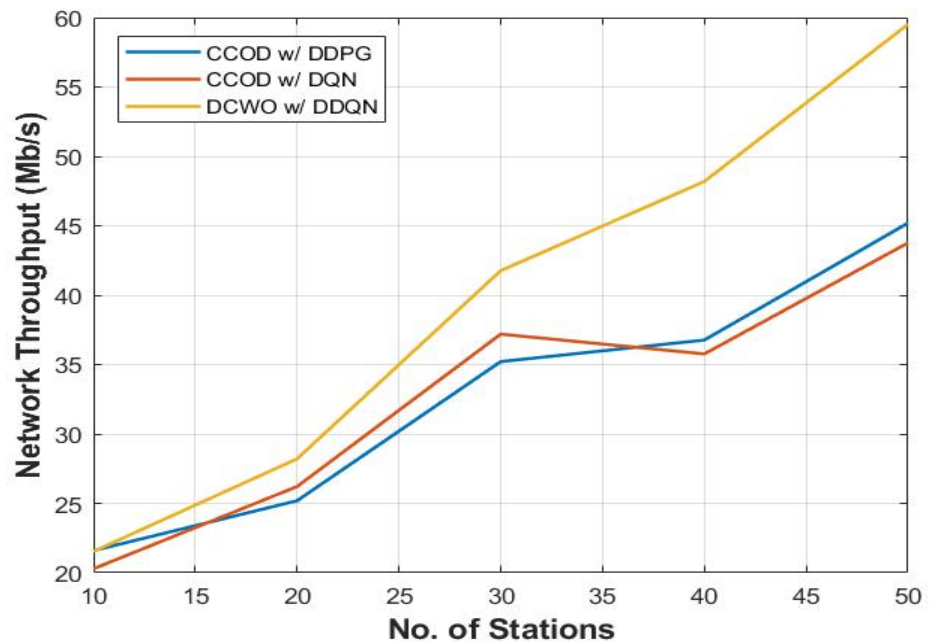


Figure 5. Network Throughput for Static Scenario.

As the standard 802.11 results in decrease of network throughput in such a scenario while our proposed scheme maintains the network efficiency, thus resulting in enhanced network performance. The results show that DDQN outperforms DQN and DDPG in terms of network throughput in the static scenario.

In the dynamic scenario, the number of nodes is not fixed and increases from 10 to 50. The purpose for designing this scenario was to evaluate the performance of proposed algorithm in a dynamic network. As the number of nodes increased from 10 to 50, it results in an increase in the collision rate. With the increase in the number of nodes, the values of CW are updated dynamically. Figure 6 shows the Instantaneous network throughput. It can be seen that DCWO performs better than CCOD with DQN and DDPG. CCOD improves instantaneous throughput by 5% and 10%, respectively, compared to the CCOD with DDPG and CCOD with DQN due to the former choosing optimal CW dynamically. The dynamic selection of the optimal size of the CW is shown in Figure 7.

The network throughput for the dynamic scenario is shown in Figure 8. From the Figure it is evident that DCWO is also outperforming CCOD with DQN and with DDPG in the dynamic scenario. DCWO achieves on average 23% higher network throughput than CCOD when the network size exceeds 40 nodes. DCWO also gives better performance even when the total number of nodes in the network are below 40.

Giving fair opportunity to access the wireless channel is also an important performance metric for an algorithm. We have also evaluated our simulation results considering the Jain's fairness index. Figures 9 and 10 show the Fainess index for DCWO in static and dynamic scenarios, respectively. The figures further compare the performance of DCWO and CCOD with DQN as well as CCOD with DDPG. In both scenarios it is shown that CCOD performance better than both of the algorithms of the CCOD.

Our scheme also performs better in terms of cumulative reward, as in the case of network throughput and the fairness index. The plot of cumulative reward against the number of steps shows performance of DDQN against DQN, and DDPG. The uptrend of DCWO(DDQN) curve, as shown in Figure 11, is learning to select the best control actions to get more rewards, thus achieving the optimal control policy more rapidly as compared to CCOD with DDPG and DQN.

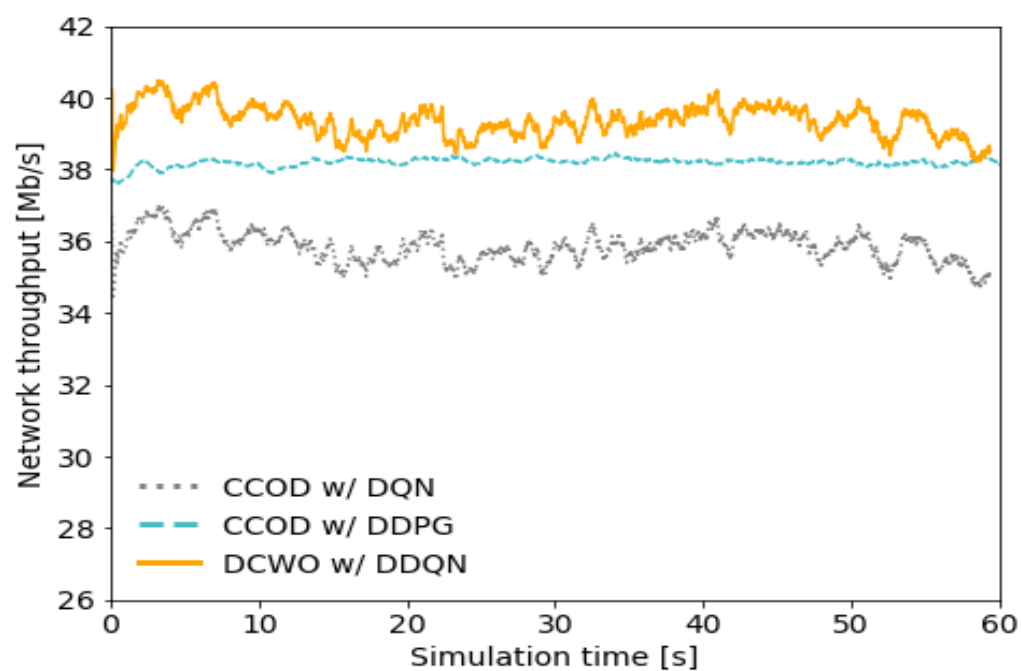


Figure 6. Instantaneous Network Throughput for Dynamic Scenario.

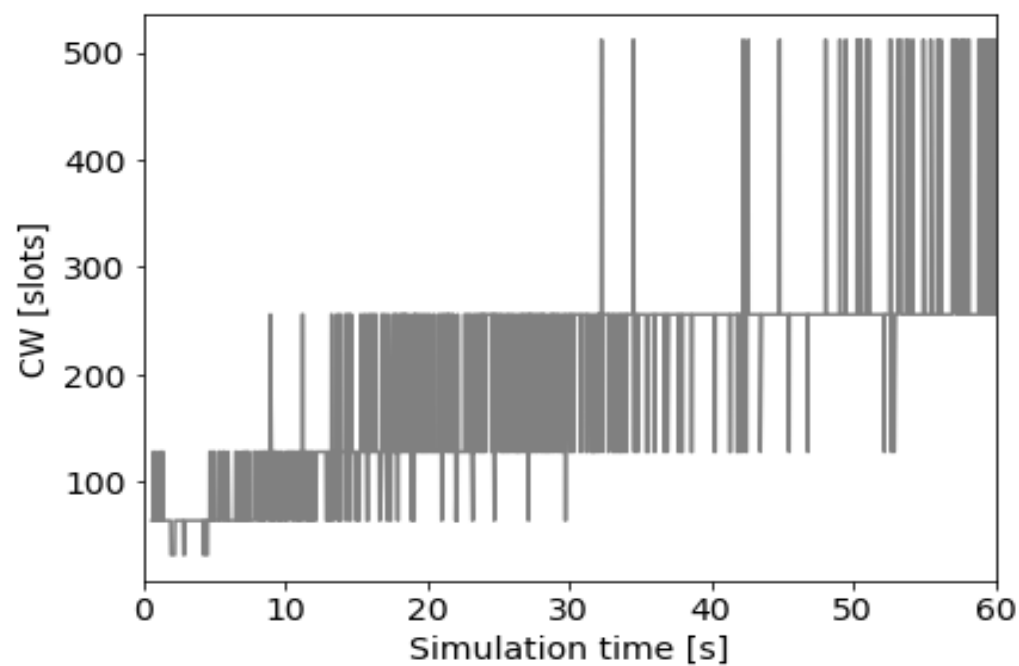


Figure 7. CW Variation by DCWO in Dynamic Scenario.

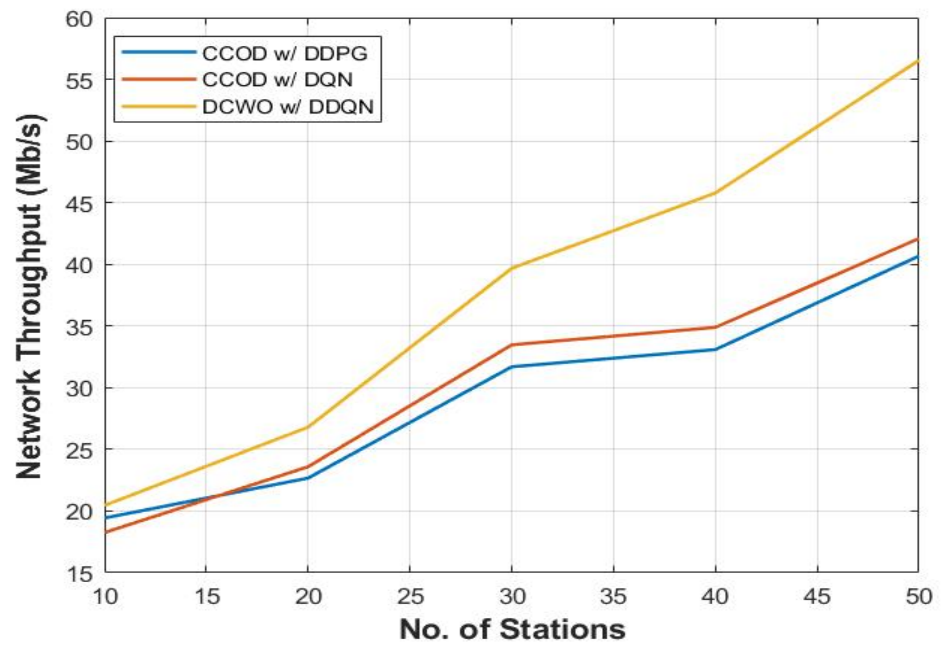


Figure 8. Network Throughput for Dynamic Scenario.

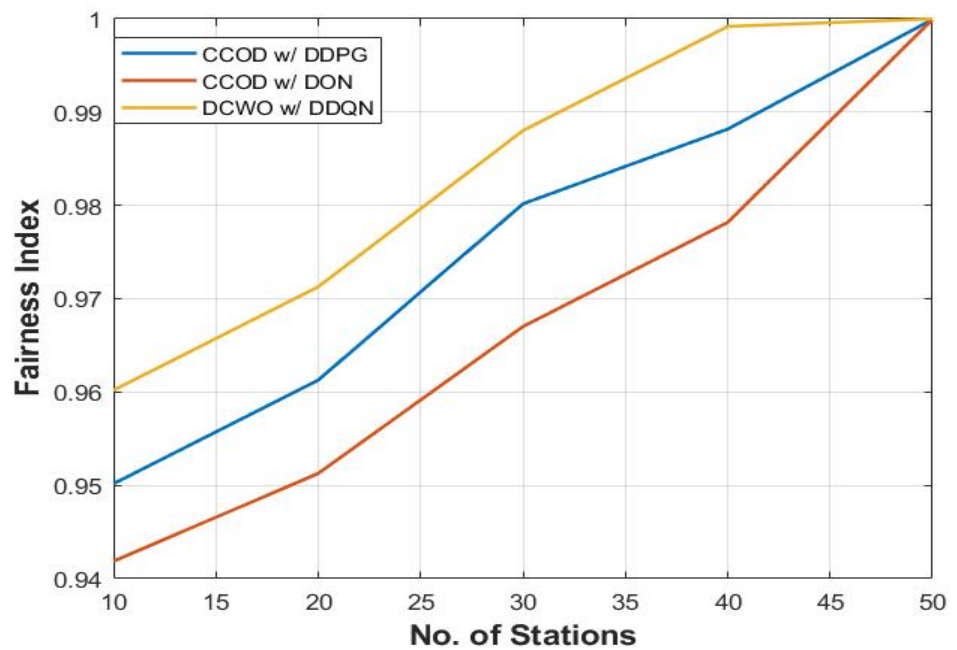


Figure 9. Network Fairness Index for Static Scenario.

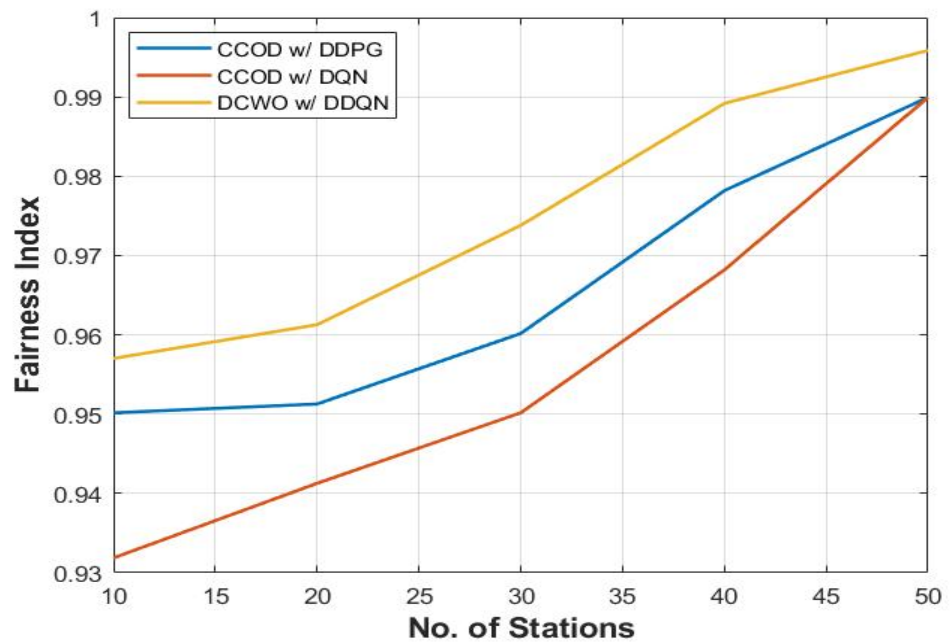


Figure 10. Network Fairness Index for Dynamic Scenario.

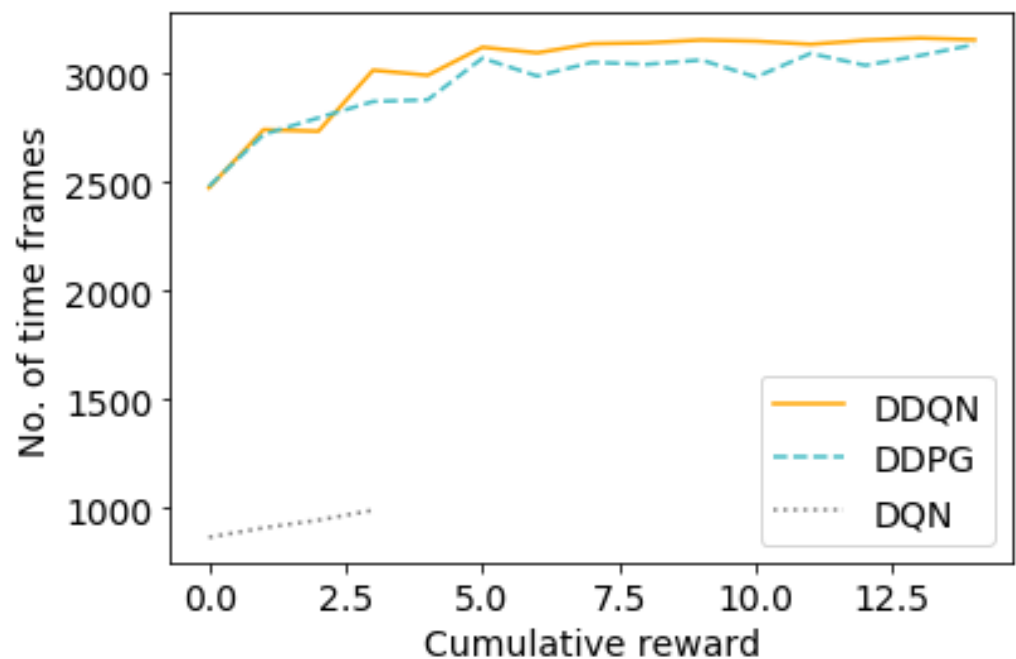


Figure 11. Cumulative Reward vs. No. of Time Frames.

8. Conclusions

With the presence of a massive number of advanced network devices, CW optimization can now be examined using DRL methods. DRL is suitable for optimizing wireless networks as it deals with smart software agents that take action in a specific environment to maximize the reward. We have proposed DCWO, a method that takes advantage of the principles of DRL for correct CW configuration for 802.11ax under different network conditions. It makes use of a trainable control algorithm, DDQN. Our experiments have shown that DRL can be used successfully for dealing with CW optimization problems. We have evaluated our proposed scheme in terms of total network throughput, instantaneous throughput and fairness in static as well as dynamic scenario and cumulative reward. The simulation

results show the effectiveness of our scheme as it yields overall better network performance than those obtained from CCOD. In terms of total network throughput DCWO gives 28% and 23% better performance than CCOD in static and dynamic scenarios, respectively. DCWO also gives around 10% better instantaneous throughput than the CCOD. In terms of fairness, DCWO with DDQN achieves almost near to optimal fairness in static scenario and better than CCOD with DQN and DDPG in dynamic scenarios. Similarly, cumulative reward achieved by DCWO with DDQN is much better than DQN with CCOD and almost the same with CCOD with DDPG; however, the uptrend of DCWO is better.

Author Contributions: Conceptualization, K.A. and B.K.; methodology, K.A. and B.K.; writing—original draft preparation, K.A.; writing—review and editing, G.-Y.K. and B.K.; supervision, B.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported in part by Basic Science Research Program through the National Research Foundation(NRF) Korea funded by the Ministry of Education (No. 2018R1D1A1B07049758).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Afaqui, M.S.; Garcia-Villegas, E.; Lopez-Aguilera, E. IEEE 802.11 ax: Challenges and requirements for future high efficiency wifi. *IEEE Wirel. Commun.* **2016**, *24*, 130–137. [\[CrossRef\]](#)
2. Index, C.V.N. *Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper*; Cisco: San Jose, CA, USA, 2017; Volume 7, p. 180.
3. López-Raventós, Á; Wilhelmi, F.; Barrachina-Muñoz, S.; Bellalta, B. Combining software defined networks and machine learning to enable self organizing wlangs. In Proceedings of the 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Barcelona, Spain, 21–23 October 2019; pp. 1–8.
4. Afaqui, M.S. *Contributions to the Evolution of Next Generation Wlangs*; Universitat Politècnica de Catalunya (UPC): Barcelona, Spain, 2016.
5. Qu, Q.; Li, B.; Yang, M.; Yan, Z.; Yang, A.; Deng, D.-J.; Chen, K.-C. Survey and performance evaluation of the upcoming next generation wlangs standard-ieee 802.11 ax. *Mob. Netw. Appl.* **2019**, *24*, 1461–1474. [\[CrossRef\]](#)
6. Liu, J.; Hatanaka, M.; Onoye, T. A Collision Mitigation Method on Spatial Reuse for Wlan in a Dense Residential Environment. In Proceedings of the 21st Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2018), Matsue, Japan, 26–27 March 2018; pp. 302–307.
7. Sun, Y.; Peng, M.; Zhou, Y.; Huang, Y.; Mao, S. Application of machine learning in wireless networks: Key techniques and open issues. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3072–3108. [\[CrossRef\]](#)
8. Nicholson, C. A Beginner’s Guide to Deep Reinforcement Learning. *A.I. Wiki*. 2020. Available online: <https://wiki.pathmind.com/deep-reinforcement-learning> (accessed on 10 March 2021).
9. Hjerde, S.T.N. Evaluating Deep Q-Learning Techniques for Controlling Type 1 Diabetes. Master’s Thesis, Faculty of Science and Technology, Department of Physics and Technology, UiT The Arctic University of Norway, Tromsø, Norway, 2020.
10. Ning, Z.; Dong, P.; Wang, X.; Guo, L.; Rodrigues, J.J.; Kong, X.; Huang, J.; Kwok, R.Y. Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 1060–1072. [\[CrossRef\]](#)
11. Ji, H.; Alfarraj, O.; Tolba, A. Artificial intelligence-empowered edge of vehicles: Architecture, enabling technologies, and applications. *IEEE Access* **2020**, *8*, 61020–61034. [\[CrossRef\]](#)
12. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.-C.; Kim, D.I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3133–3174. [\[CrossRef\]](#)
13. Hasselt, H.V.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
14. Sewak, M. *Deep Q Network (DQN), Double DQN, and Dueling DQN, Deep Reinforcement Learning*; Springer: Singapore, 2019; pp. 95–108.
15. Hasselt, H. Double q-learning. *Adv. Neural Inf. Process. Syst.* **2010**, *23*, 2613–2621.
16. Choudhary, A. A hands-On Introduction to Deep Q-Learning Using Openai GYM in Python. *Analytics Vidhya*. 2019. Available online: <https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learningpython> (accessed on 19 February 2020).
17. Kremer, G.; Owezarski, P.; Berthou, P.; Capdehourat, G. Predictive estimation of wireless link performance from medium physical parameters using support vector regression and k-nearest neighbors. In Proceedings of the International Workshop on Traffic Monitoring and Analysis, London, UK, 14 April 2014; pp. 78–90.
18. Testa, D.D.; Danieleto, M.; Nunzio, G.M.D.; Zorzi, M. Estimating the number of receiving nodes in 802.11 networks via machine learning techniques. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–7.

19. Edalat, Y.; Ahn, J.-S.; Obraczka, K. Smart experts for network state estimation. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 622–635. //Ref16 [CrossRef]
20. Yu, Y.; Wang, T.; Liew, S.C. Deep-Reinforcement Learning Multiple Access for Heterogeneous Wireless Networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1277–1290. [CrossRef]
21. Wydmański, W.; Szott, S. Contention window optimization in IEEE 802.11 ax networks with deep reinforcement learning. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021; pp. 1–6.
22. Kumar, A.; Verma, G.; Rao, C.; Swami, A.; Segarra, S. Adaptive contention window design using deep q-learning. In Proceedings of the ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–12 June 2021; pp. 4950–4954.
23. Cisco, U. Cisco Annual Internet Report (2018–2023) White Paper. 2020. Available online: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/whitepaper-c11-741490.html> (accessed on 26 March 2021).
24. Khorov, E.; Kiryanov, A.; Lyakhov, A.; Bianchi, G. A tutorial on IEEE 802.11 ax high efficiency w lans. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 197–216. [CrossRef]
25. Gallo, P.; Kosek-Szott, K.; Szott, S.; Tinnirello, I. Cadwan: A control architecture for dense wifi access networks. *IEEE Commun. Mag.* **2018**, *56*, 194–201. [CrossRef]
26. Deng, D.-J.; Chen, K.-C.; Cheng, R.-S. IEEE 802.11 ax: Next generation wireless local area networks. In Proceedings of the 10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, Rhodes, Greece, 18–20 August 2014; pp. 77–82.
27. Huang, Y.; Wang, Y.; Zhu, R.; Chen, X.; Meng, Q. Synchronized contention windows-based backoff algorithm in IEEE 802.11 wireless networks. In Proceedings of the 2016 International Conference on Computer, Information and Telecommunication Systems (CITS), Kunming, China, 6–8 July 2016; pp. 1–5.
28. Sung, C.-H.; Deng, D.-J. Contention window size adjustment in unsaturated IEEE 802.11 w lans. In Proceedings of the International Conference on Internet of Things as a Service, Linz, Austria, 22–25 October 2017; pp. 3–10.
29. Zhang, C.; Patras, P.; Haddadi, H. Deep learning in mobile and wireless networking: A survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2224–2287. [CrossRef]
30. Bast, S.D.; Torrea-Duran, R.; Chiumento, A.; Pollin, S.; Gacanin, H. Deep reinforcement learning for dynamic network slicing in IEEE 802.11 networks. In Proceedings of the IEEE INFOCOM 2019–IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; pp. 264–269.
31. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]