Research Article

Intelligent Resource Allocation for Ultradense Networks Based on Improved Reinforcement Learning

Zhou Ye

Zhejiang Institute of Mechanical and Electrical Engineering, Hangzhou, China

Correspondence should be addressed to Zhou Ye; yezhou@zime.edu.cn

Received 9 January 2022; Revised 19 January 2022; Accepted 25 January 2022; Published 2 March 2022

Academic Editor: Tongguang Ni

Copyright © 2022 Zhou Ye. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The urgent need of high-quality mobile services and improved user experience has driven to develop high-capacity mobile network system. To achieve this goal, ultradense network (UDN) has been widely considered as one of the promising solutions that significantly expand the network capacity. However, as the communication resources and the number of links per unit area are extremely densified within UDN, efficient and high-quality scheduling and resource allocation for UDN become challenging. Even worse, the complex environment of UDN limits the direct adoption of traditional allocation schemes to UDN. In this manuscript, the author is going to propose an efficient resource allocation algorithm for UDN based on deep reinforcement learning. First, the author presents the resource allocation strategies in cellular network server based on double Q-learning. Then, the author optimizes the algorithm by pruning redundant model weights and making a tradeoff between computational complexity and performance to meet the requirements of low latency and limited computing costs. The experiment and simulation results show that the pruning algorithm effectively reduces 50% model parameters. The UDN allocation performance is still acceptable as the proposed algorithms save up to 50% complexity.

1. Introduction

The rapid growing need of high-quality mobile services and improved user experience has become a driving force for a high-performance mobile network. On the other hand, the improved efficiency and higher network performance enable emerging user experiences as well as novel applications, like the next-generation smart Industry 4.0 [1]. The need for connection speed and data traffic is both growing at an exponential rate. According to some reports [1, 2], the data rates are projected to increase by around ten times every five years. Moreover, trillions of mobile devices and sensors will be connected together across the world via wireless network. The upcoming Internet of Things (IoT) and future fifth-generation (5G) mobile networks will encounter a halt unless we can rapidly increase the capacity of current communication system. Therefore, a paradigm shift and enhancement in every aspect of current mobile network system are required to provide the immense amount of data traffic arising from high-resolution video applications [2]. In

the next-generation communication system, many ideas and schemes are proposed to resolve the challenges imposed by the requirements of a lower network latency, higher peak data rates, more reliable connection stability, increased availability, and better user experiences.

The opportunities of 5G system have attracted great attention from academics, industries, and governments. Among the above-mentioned points, how to develop a high-performance network that offers massive capacity within a dense area becomes one of the major challenges. Ultradense network (UDN) is one of the promising solutions where the communication nodes and the number of links per unit area are extremely densified. Rather than focusing on the evolutionary improvements of previous generations in terms of spectral efficiency, the densification of network is able to bring about more significant overall system performance gain by dramatically increasing the network infrastructures [3]. The definition of UDN varies among different literature. Basically, UDN can be abstracted as a special type of network with very dense radio resources than current networks.

Specifically, UDN is equipped with tremendous amounts of base stations (BSs) with higher density, where the BS density reaches and even exceeds the user density. Besides, an UDN can also be modeled a network where the intersite wireless access points (APs) are separated by only a few meters.

However, UDN also faces several significant hurdles towards the way to constructing high-capability and high-efficiency mobile network. The factors causing the challenges include the dense radio resources, mobility, and interference between devices. Although the dense cellular networks can efficiently offload data traffic from traditional macro cellular networks, how to optimally allocate and schedule the available network resources affects the overall network performance and user experience. This makes the resource allocation algorithms of UDN a crucial part when design UDNs. Tremendous amounts of previous research works have studied the resource allocation policies in various perspectives, including UDN as well as heterogeneous network scenarios (HetNet) scenarios. The authors in [4] propose a novel UDN resource allocation algorithm that has low computational overhead. However, the resulting quality of resource allocation under fast changing and complex network environment is not satisfactory. To improve this shortcoming, deep reinforcement learning-based scheduling algorithms are presented in [5, 6]. But the limitations of these two algorithms lie in that they are still computationally intensive workloads, which are not suitable to be employed in UDNs with different sizes.

In this paper, we propose deep reinforcement learning-based resource allocation scheme to address the above-mentioned challenges. Our algorithms can be efficiently applied to allocate UDN resources in the 5G scenario. The following parts summarize our contributions:

- (i) We first model the mathematical model of resource allocation for UDN. Then the NP-hard nonconvex resource allocation problem in UDN is transformed into a form that can be solved by deep reinforcement algorithm.
- (ii) A novel and efficient resource allocation algorithm based on deep reinforcement learning is proposed to achieve near-optimal resource allocation under different communication environments.
- (iii) We also consider the efficient implementation problem of deep reinforcement learning-based resource allocation algorithm. The weight and model pruning schemes are utilized to train low-complexity models.
- (iv) Compared to previous works, our work further considers the factors of computing power-constrained scenarios. The optimized (pruned) algorithms are able to perform energy-efficient resource allocation.

The rest of this paper is organized as follows. Section 2 introduces the system model and problem formulation for UDN resource allocation. Section 3 presents the proposed

algorithms and optimization methods. Section 4 presents the simulation and experiment results. Section 5 concludes the whole paper.

2. System Model and Problem Formulation

To accurately model the resource allocation problem, we choose to use the ultradense networks of 5G downlink, as shown in Figure 1. In the following parts, we summarize the main points of this model according to [7]. In this case, a total of N small base stations (SBSs) will be randomly distributed within the area of one macro cell. In the center of the macro cell, one macro base station (MBS) is placed to process the incoming data and perform scheduling policies. Inside the MBS, a mobile edge computing (MEC) server is installed that is responsible for computing and allocating resources for the whole UDN.

Without loss of generality, we define the set of N SBSs as the form: $\mathcal{N} = [1, 2, \dots, N]$. Many small user equipments (SUEs) will randomly present and move within each SBS. The arrival of users is considered as a random process. Moreover, the user arrival and departure are regarded independent. Hence, the arrival of each SUE to each micro cell is parameterized with a Poisson process and parameter λ_t . For each time slot, the probability that a total of n new SUEs will arrive in the micro cell, and this can be expressed as the following equation:

$$P(n) = \frac{(\lambda_t \tau)^n}{n!} e^{-\lambda_t \tau}, \quad (1)$$

where τ denotes the time period. The probability of SUEs that will leave the small cell can also be modeled as a Poisson process similar to equation (1) with a different parameter μ_t .

We denote the SUE set as $\mathcal{U}_n = [1, 2, \dots, U_n] (n \in \mathcal{N})$. Outside the micro cell, there would be some macro user equipments (MUEs) that cannot be fully covered by SBS while can be reached by the center MBS. We define these MUEs as $\mathcal{U}_M = [1, 2, \dots, U_M]$. Over time domain, a micro cell n has various numbers of SUEs at a time slot t . So we define the set of SUEs at time slot t of micro cell n as $U_n(t)$.

In reality, the identical network resources can be reused by multiple SBSs at a specific time slot, therefore the interference between SUEs and SBSs in other micro cells. In this work, we quantize this kind of interference as the signal to interference-plus-noise ratio (SINR) as follows:

$$SINR_{U_n(t)} = \frac{p_n g_n^m}{\sum_{i=1, i \neq n}^N x_n^m(t) p_n g_i^m(t) + \sigma^2}, \quad (2)$$

where p_n represents the assigned power to each associated SUE and g_n^m denotes the gain of channel from SBS within micro cell n to the SUE when the resource block (RB) m is reused by multiple UEs. We assume that the channel is an additive-white Gaussian noise (AWGN) channel so σ denotes the variance of AWGN noise. Finally, $x_n^m(t)$ is a Boolean variable indicating whether the RB is assigned to a specific UE. If $x_n^m(t) = 1$, it means that RB m is allocated. $x_n^m(t) = 0$ means RB m is not assigned to any SUE.

FIGURE 1: An illustration of ultradense network redrawn from [7].

Once we have the metric SINR for each SUE set $U_n(t)$, the summation of overall downlink throughput at a specific time slot t within the micro cell n could be computed based on the following equation:

$$TP_n(t) = \sum_{U_n \in \mathcal{U}_n} TP_{U_n}(t), \quad (3)$$

where $TP_{U_n}(t)$ can be shown as

$$TP_{U_n}(t) = B_m \times \log_2(1 + \text{SINR}_{U_n}(t)), \quad (4)$$

where B_m is the bandwidth of the RB m , which is usually fixed for a given RB.

It is more meaningful to consider the total throughput for the whole UDN. Hence, we can calculate the total throughput based on the aforementioned $TP_n(t)$ as follows:

$$TP(t) = \sum_{n=1}^N TP_{U_n}(t). \quad (5)$$

The UDN system not only requires high-speed network throughput but also needs to achieve some levels of efficiency. Apart from the total throughput for the UDN system, the other factors need to be considered are the spectral efficiency (SE) and energy efficiency (EE) for the system. Similar to [5], the SE is denoted as the ratio between the total throughput and the bandwidth of the given UDN at time slot t . In a similar pattern, the EE for this UDN is defined as the ratio between the total throughput and power dissipation at time t . The metrics SE and EE are summarized as follows:

$$\begin{cases} SE(t) = \frac{R(t)}{\sum_{i=1}^M B_i}, \\ EE(t) = \frac{R(t)}{P_M + \sum_{n=1}^N \sum_{U_n \in \mathcal{U}_n} U_n(t) p_n}, \end{cases} \quad (6)$$

where p_M denotes the power consumption by macro cell network.

The goal of achieving optimal resource allocation for a given UDN is equivalent to maximizing the spectral

efficiency SE as well as the energy efficiency EE at the same time. The conventional mobile system adopts various approaches to generate a good resource allocation policy. These algorithms include discrete optimization approaches, convex optimization methods, and other heuristic-based algorithms. However, the feasibility and performance of these previous algorithms heavily depend on the application scenarios. When the communication environment varies, a significant degradation in the quality of resource allocation is observed. Hence, we proposed a novel and efficient deep reinforcement learning algorithm in the following sections.

3. Proposed Efficient Resource Allocation Algorithms for UDN

3.1. Deep Reinforcement Learning. The resource allocation tasks for the UDN system can be regarded as a problem of searching the optimal control sequence that distribute the resource blocks within the UDN. However, the complex communication environment makes it difficult to accurately obtain the optimal allocation strategies that dissipate minimum energy while maintaining high spectral efficiency. To address this problem, similar to [5], we take the advantages of deep Q-learning algorithms [8]. Moreover, we also present optimization schemes to develop computationally efficient models with negligible performance loss.

As shown in Figure 2, for the case of UDN resource allocation task, the MEC server is regarded as an agent that interacts with the macro cell and micro cell environment. The goal of MEC server agent is to maximize the total reward between the first state and the final state. MEC server will make certain decisions that allocate the network resources based on the received data and information from different levels of cellular networks. After the decision is made based on the policy within the agent, the state of the UDN system will be changed and reach to various scenarios defined as state. As the decision (or actions) are taken, the agent receives positive or negative rewards.

Figure 3 depicts the overall diagram of our proposed deep reinforcement learning-based efficient resource allocation scheme for UDN. There are two parts in the depicted figure: training phase and inference phase. For the training phase, we use the system model in Section 2 to emulate the environment of UDN. Once the training phase has been completed, the trained model is optimized via network pruning algorithm to reduce the required model weights and computation complexity. In the following parts, we introduce the development steps of our algorithms in detail, including reward function design, action space design, and network model pruning.

3.1.1. Reward Function Design. The reward function design is a key factor that determines the final quality of resource allocation. This is because the reward function will guide the agent to take actions and make decisions. The actions will impact the state of UDN system. The reward design plays an important role in the task since it determines the quality of resource allocation and intelligence level of trained

FIGURE 2: Overall diagram of deep reinforcement learning.

FIGURE 3: Proposed deep reinforcement learning-based efficient resource allocation scheme.

allocation algorithms. The key principle of designing reward function is that the reward should accurately reflect the goal of target application. Here, we adopt the two efficiency metrics, namely spectral efficient SE and energy efficiency EE in equation (6), as the reward function. We make minor modifications by summing these two metrics into one single reward with a weight ϵ . At a time slot t , the reward function is

$$r(t) = (1 - \epsilon)SE(t) + \epsilon EE(t), \quad (7)$$

where the reward function considers both energy consumption and the spectral efficiency. It should be noted that the reward at time t is determined by the current state of network state. If we train the model properly, the agent is able to achieve the optimal reward by balancing the two aspects.

Achieving the optimal reward at time t as equation (7) is not enough since it is not guaranteed that the agent can achieve the global optimal goal. We hope the agent is able to

maximize the total reward over a given period of time. In this case, the reward for single time step becomes a cumulative reward function through summing up the received reward at different time steps. The total reward that maximizes the reward between the initial and state becomes:

$$R_{total} = \sum_{t=1}^T \beta^t r(t), \quad (8)$$

where R_{total} is the sum of each reward the initial state to the end. In deep reinforcement learning, a discount factor $\beta \in [0, 1]$ is added to weight the different contributions of rewards in short term and long term. Small discount factor β reduces the importance of reward obtained from previous time steps. It is easy to understand that $\beta = 1$ means all the rewards are treated equally.

3.1.2. Action Space. The UDN state of next time step after the control signal is sent by MEC server can be considered to be only determined by the current network states, communication environment, and the resource allocation signals of the UDN. It is independent from the previous states. Hence, the resource allocation policy of UDN can be regarded as the Markov decision process (MDP). The large search space for the action sequence is significantly reduced under the assumption of MDP.

Considering the real realizability of simulation that accommodates real UDN system, we discretize the resource allocation signal of UDN system by a constant time interval. This is because the computing resources in MEC server are limited, thus unable to process data and make resource allocation with very high frequency. Therefore, we assume that the discrete control signal for RBs is in the following form:

$$A = [x_1(t), x_2(t), \dots, x_T(t)], \quad (9)$$

where the time range of the control signals varies from time step 1 to time step T . It can be seen that the dimension of action space grows linearly as the time steps increases.

3.1.3. Model Updating and Training. Several tricks are used to make the iterative updating and training process more stable. According to [9], since the environment is usually complex, it is not realistic to store all of them into the Q table of traditional reinforcement learning. To alleviate this problem, deep Q-learning uses a deep neural network to estimate the Q-value function. Based on the Bellman equation, the Q-value updating equation is given as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right], \quad (10)$$

where $Q(s_t, a_t)$ denotes the Q-value function associated to time step t while α represents the learning rate. Likewise, $Q(s_{t+1}, a)$ is the Q value associated to next time step after the MEC server is taking action a .

The weight of the neural network is initialized using random function. And we define the weight of neural network as w . The weights are updated in an iterative pattern through utilizing the mini-batch stochastic gradient descent (SGD) training algorithms. Specifically, the training of neural networks can be complete through updating the weights with the gradient with respect to the loss function. The updating procedure can be summarized as the following formula:

$$w_{k+1} = w_k - \alpha \cdot \mathbb{E} \left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a; w_k^-) - Q(s_t, a_t; w_k) \right] \times \nabla_{w_k} Q(s_t, a_t; w_k), \quad (11)$$

where the weights of target network at the k th iteration are called w_k^- . The weights w_k^- are set to be fixed during the training process and can only be updated using the Q-network weights w_k every C training iterations. The loss function we used in this paper is the mean squared error (MSE) between the target Q value and the output results of neural network.

The updating process in equation (11) for the target network and Q-network may experience some overestimating values of actions by the analysis in [9]. The reason behind this circumstance is that the mentioned scheme uses the same network to calculate both the predicted value and the target value. Hence, it leads to some divergence between the computed results. The overestimation problem hinders the resource allocation algorithms from choosing the good action in some UDN states. As a result, the overall performance of the resource allocation algorithms is negatively impacted. The negative effects arose from overestimating values can be reduced by using two different deep Q-network and target network [9]. In this case, the Q-value function is rewritten as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [R_{t+1} + \gamma Q'(s_{t+1}, a) - Q(s_t, a_t)], \quad (12)$$

where functions $Q'(\cdot)$ and $a = \max_a Q(s_{t+1}, a)$ are used to estimate the expected Q-value by the selected action a that maximizes the value of the predicted Q-value network.

As introduced in [8], the training process for the agent will experience unstable circumstances if we just run the deep Q-learning algorithms on the state-action pairs as they occur during simulation. To equip the agent with some capabilities of memory, we introduce the experience replay buffer to store the transition of discovered data and occurred experience. The experience replay is used to reduce the correlation of observed samples, increasing the robustness of training process. As shown in Figure 3, the experience replay buffer stores the agent's experiences, including taken actions, states, and rewards over a certain time steps.

Exploration and exploitation are the two critical concepts for reinforcement learning. Exploration makes an agent improve the current knowledge about each action, while exploitation is able to select the most rewarded action by exploiting current action-value estimations. To balance the exploration and exploitation, ϵ -greedy policy is investigated to choose between exploration and exploitation randomly. It can be expressed by the following equation:

$$a_t = \begin{cases} \text{a random action, with probability } \epsilon, \\ \max_a Q(s, a), \text{ with probability } 1 - \epsilon, \end{cases} \quad (13)$$

where the agent tends to select the action that maximizes the current Q-value with probability $1 - \epsilon$. In contrast, the agent selects a random action with probability ϵ . In other words, the ϵ -greedy policy will let the agent explore most of the time. The larger ϵ value indicates that the control algorithm is more likely to choose a random movement. The value of ϵ cannot be too large since the large ϵ may deteriorate the overall performance. In the contrast, small ϵ value let the agent has a small chance to explore new experience, thus limiting the peak performance. In this paper, ϵ value is selected empirically based on the previous study.

3.2. Model Pruning. The training and inference of fully connected networks are computation-intensive and memory-intensive. Large amounts of computing and memory resources are required to perform the training and inference of deep neural network models. When the MEC server is equipped with limited computing chips, the computationally intensive workload of DNN will become the bottleneck, thus increasing the response time of resource allocation. These problems not only increase the implementation costs but also hinder the deployment of proposed resource allocation algorithms on some resource-constrained MEC servers. To solve this challenge, we use the model pruning methods in [10, 11], which are used to reduce the size of network models by removing redundant connections between fully connected layers.

There exists various types of network pruning algorithms to optimize the neural network structures. The deep compression in [10, 12, 13] is one of the promising methods to optimize sparse neural network architectures. The basic idea of deep compression is progressively searching and removing the redundant weights that have relatively small magnitudes. Figure 4 depicts the neuron connections of original and pruned fully connected layers. The original layer connection is dense and requires large excessive number of weights. Network pruning searches for the most representative and important connections for each layer. Then remove the trivial connections through setting the weights to zero, thereby effectively cutting the weights and reducing the model size as well as computational complexity. After applying the optimizations, the inference of neural network can be accelerated by several times without loss of performance. Meanwhile, the model size will shrink dramatically as large amounts of weights are pruned.

4. Simulation Results

4.1. Experiment Platform and Parameters. We perform simulations and experiments to verify the performance of proposed algorithms. The detailed configurations and process of simulation are given as follows. The proposed deep reinforcement learning algorithms are implemented using the state-of-the-art deep learning library, Pytorch. GPU is

FIGURE 4: Weight pruning process for original deep neural network.

used to accelerate the training and inference process. The target network and deep Q-network both adopt a five-layer basic structure, where there are 64, 128, 256, 192, 96 neurons for each hidden layer.

During the training process, the mini-batch size is selected as 64 to stabilize the weight updating and achieve a smooth convergence process. The Adam optimizer [14] is used for achieving faster convergence. The input of trained resource allocation algorithm is the vector representing the network state of UDN. Suggested by [8], the probability of ϵ -greedy policy is chosen as 0.05. We create an experience replay memory with a buffer size of 3200. The weights of target network are copied every 5 training iterations. The discount factor of cumulative reward calculation is set to 0.96. The parameters of UDN are similar to those in [5]. The simulation parameters are summarized in Table 1.

4.2. Simulation Results. We use the UDN system model to generate multiple training and testing data. We train various models under different weight sparsity constraints as in [10]. For each model, we train the associated model for at least 800 training epochs.

The average loss value of various DQN models for different training epochs is illustrated in Figure 5. As we can see, the loss values of all models decrease as the training epochs increase. The dense model achieves the fastest convergence. The sparse model with sparsity = 0.4 achieves comparable loss value as the dense model. This means that this sparse model is also able to yield promising performance while reducing the weight number and computational complexity by around 60%. The convergence process of model with 60% sparsity is much slower than other models. It can be inferred that the resulting performance is not acceptable.

We also test the yield system efficiency of propose resource allocation schemes under different number of SBSs. The number of SBS ranges from 6 to 18. The results are depicted in Figure 6. It can be seen that the spectral and energy efficiency gradually drops as the number of SBS is

TABLE 1: Summary of simulation parameters.

Parameter	Value
Total bandwidth	20 MHz
Number of RBs	80
Number of time slots	1000
Value of p_M	46 dBm
Batch size, B	64
Discount factor, γ	0.96
Experience replay buffer size, N	3200
Update frequency, C	5
ϵ -Greedy probability	0.05

FIGURE 5: Loss value of DQN under different training epochs.

FIGURE 6: Spectral and energy efficiency under different numbers of SBS.

increasing for all models. We can see that the unpruned model obtains the highest system efficiency. The model with sparsity 50% is able to achieve about 90% efficiency compared to the unpruned baseline. Meanwhile, the total complexity to perform the resource allocation tasks is reduced by 50%. Hence, the weight and model pruning strategies provide a tradeoff between quality of resource allocation and algorithm complexity. For some MEC server that has limited computing resources or heavy workload, we can choose the resource allocation models with more sparsity to make the network resource scheduling more quickly and efficiently.

5. Conclusion

In this paper, we develop and propose the resource allocation algorithms for UDN based on deep reinforcement learning. First, we introduce the resource allocation strategies in MEC server based on double deep Q-learning. Then, to meet the requirements of low latency and limited cost, we optimize the algorithm by pruning redundant model weights and making a tradeoff between computational complexity and performance. The experiment and simulation results show the gain and effectiveness of proposed algorithms. Our proposed approach can be applied to UDN resource allocation in the 5G system, especially when the computing power is limited. The future work will focus on considering more impacting factors of UDN into this framework.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was sponsored in part by Zhejiang Machinery and Electrical Group Co., Ltd. under the 2017 Science and Technology Projects: Exploration of the Construction of Smart Campuses in Higher Vocational Colleges under the Background of Big Data (2017JD007). This work was also funded by the Special Project “Epidemic and Education” of Educational Science Planning Project of Zhejiang Province: Exploration and Practice of the Construction of “Smart Cloud Classroom” in Higher Vocational Colleges under the Background of the Epidemic Taking Zhejiang Vocational and Technical College of Mechatronics as an Example (2020YQJY316).

References

- [1] S. K. Rao and R. Prasad, “Impact of 5g technologies on industry 4.0,” *Wireless Personal Communications*, vol. 100, no. 1, pp. 145–159, 2018.
- [2] T. Q. Duong, X. Chu, and H. A. Suraweera, *Ultra-dense networks for 5G and beyond: modelling, analysis, and applications*, John Wiley & Sons, Hoboken, New Jersey, United States, 2019.
- [3] P. Hao, X. Yan, Y.-N. Ruyue, and Y. Yuan, “Ultra dense network: Challenges enabling technologies and new trends,” *China Communications*, vol. 13, no. 2, pp. 30–40, 2016.
- [4] C. Niu, Y. Li, R. Q. Hu, and F. Ye, “Fast and efficient radio resource allocation in dynamic ultra-dense heterogeneous networks,” *IEEE Access*, vol. 5, pp. 1911–1924, 2017.
- [5] Z. Liu, X. Chen, Y. Chen, and Z. Li, “Deep reinforcement learning based dynamic resource allocation in 5g ultra-dense networks,” in *Proceedings of the IEEE International Conference on Smart Internet of Things (SmartIoT)*, pp. 168–174, IEEE, Tianjin, China, 9–11 Aug. 2019.
- [6] H. Li, H. Gao, T. Lv, and Y. Lu, “Deep q-learning based dynamic resource allocation for self-powered ultra-dense networks,” in *Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, IEEE, Kansas City, MO, USA, 20–24 May 2018.
- [7] X. Chen, Z. Liu, Y. Chen, and Z. Li, “Mobile edge computing based task offloading and resource allocation in 5g ultra-dense networks,” *IEEE Access*, vol. 7, pp. 184172–184182, 2019.
- [8] V. Mnih, K. Kavukcuoglu, D. Silver et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [9] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, Phoenix, AZ, USA, 12 February 2016.
- [10] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” 2015, <https://arxiv.org/abs/1510.00149> arXiv preprint arXiv:1510.00149.
- [11] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Gutttag, “What is the state of neural network pruning?,” 2020, <https://arxiv.org/abs/2003.03033> arXiv preprint arXiv:2003.03033.
- [12] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *Proceedings of the Advances in neural information processing systems*, pp. 1135–1143, Montreal, Canada, 2015.
- [13] J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, and W. Lin, “Thinet: pruning cnn filters for a thinner net,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2525–2538, 2018.
- [14] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, <https://arxiv.org/abs/1412.6980> arXiv preprint arXiv:1412.6980.