Reinforcement Learning Based Computation Offloading in Vehicular Edge Computing

Qiaoxin Chen

***Abstract*—This paper designs a transmission model in VEC to improve the communication range of each vehicle, which considers the trust relationship and communication interference between edge servers. An offloading strategy based on a Deep Q-network algorithm is proposed, which can promote the convergence of deep reinforcement learning algorithm. Simulation results show that the proposed scheme has the best convergence compared to other benchmark algorithms.**

***Index Terms*—edge computing, Internet of Vehicles, Deep Reinforcement Learning**

I. Introduction

The 5-th generation (5G) mobile communication technology encourages the gradual migration of remote cloud functions to the network's edge [5], [6], driving a shift from centralized cloud computing to edge computing (EC) [4], [7]. It would be a promising technology for VTs in response to the above problems. EC places edge servers (small cellular base stations or WiFi access points) in RSUs for functions such as mobile computing, network control, and storage. The applications [8] can be offloaded to RSUs adjacent to the VT for processing, dramatically lowering VT workload and application execution latency, which is known as Vehicular Edge Computing (VEC) [9].

Despite the clear advantages, most of the current studies on multi-hop offloading [12]–[17] tend to offload application as a whole rather than partial offloading. Partial offloading takes into account the data dependency of tasks and optimizes the transmission latency by offloading only computationally complex tasks, thus reducing the application completion time. It is worth noting that different types of applications may require different computing resources. The offload decision should be adapted when the edge environment changes. Moreover, trust risks and transmission interference between edge servers create tremendous difficulties for fully reaping benefits of multi-hop offloading. In practice, VTs tend to offload applications to the powerful edge server for execution, but as resource providers, due to the cooperative competition among various SPs, edge servers may refuse to support offloading services or deliberately throttle resources only to provide low-quality services. [17]. Therefore, the feasible transmission path should consider the trust relationship between the physically adjacent edge servers. At the same time, due to the fluctuation of the channel and noise interference, choosing an unreasonable multi-hop offloading path may provoke the risk of offload failure, thereby suppressing efficiency of offloading services.

Qiaoxin Chen is with the Department of Informatics and Communication Engineering, Xiamen University, Xiamen University, Xiamen 361005, China. E-mail:fjqiaoxin_chen@163.com.

Thus, we develop a novel multi-hop computation offloading method to reduce the workflow application completion time caused by task computation and data transmission, while improving the offloading success rate in vehicle edge environments.

II. System Model

We assume that VT has computing and communication capabilities [1], [2], [28], and RSUs equipped with edge servers can act as routers to provide communication services covering a dedicated local area. Therefore, a group of mutually trusted RSUs can form an edge self-organizing network to share computing resources. Compared to single-hop offloading, applications can be computed not only in VT or the nearst RSUs, but also offloaded to further RSUs for execution via multi-hop transfers. The multi-hop offloading relies on the social trust relationship among RSUs, thus the trust-based multi-hop edge environment denoted by $\boldsymbol{D} = <\boldsymbol{M}, \boldsymbol{X}>$. $\boldsymbol{M}$ is the set of computation units $\{m_0, m_1, \cdots, m_x\}$, containing VT m_0 and x RSUs, and the computation units can be expressed as:

$$m_i = p_i \cdot f_i, \tag{1}$$

where p_i denotes the computing unit's average transmission capacity. f_i denotes the computing unit's calculation capacity. The calculation capacity of each computing unit is supposed to be known, and the RSU calculation capacity is more than the terminal one.

$$\boldsymbol{X} = [m_i, m_j, \cdots], \forall m_i, m_j \in \mathbf{M}. \tag{2}$$

As this study focuses on multi-hop offloading, we indicates $\boldsymbol{X}$ that a set containing all communicable routes. As shown in Eq. (2), $m_i, m_j \in \boldsymbol{X}$ indicate that the computing unit m_i and m_j are physical neighbors who trust one another. When m_i receives tasks, it can not only calculate independently, but also continue to offload the task to m_i. It is worth noting that if still has trustworthy neighbors, this process can be repeated for multi-hop collaborative computation. The channel between m_i and m_j is denoted as follows:

$$\beta_{i,j} = (w_{i,j}, h_{i,j}, {\sigma_j}^2, b_{i,j}^{dl}, b_{i,j}^{ul}, \alpha_{i,j}), \tag{3}$$

$$b_{i,j}^{ul} = \xi \cdot b_{i,j}^{dl} = w_{i,j} \cdot \log_2(1 + \frac{p_i h_{ij}}{{\sigma_j}^2}), \tag{4}$$

where $w_{i,j}$ denotes the bandwidth between m_i and m_j. $h_{i,j}$ denotes the channel transmission gain. ${\sigma_j}^2$ denotes the channel noise power of the receiving unit. $b_{i,j}^{dl}$ denotes the downlink transmission rate of the channel, which affects the transmission latency. $b_{i,j}^{ul}$ denotes the uplink transmission rate of the channel, which is asymmetrically distributed with $b^{dl}_{i,j}$. $\alpha_{i,j}$ denotes the stability of transmission in the channel and $\forall \alpha_{i,j} \in [0,1]$. When it tends to 1, it indicates that the channel is smooth. When it tends to 0, it indicates that the channel has difficulty maintaining a stable communication link.

VT can make offloade decisions through its offloading scheduler, with some tasks executed locally and the rest transferred to the RSUs for auxiliary execution. A feasible offloading strategy $\boldsymbol{H}_{n_0:n_i} = \{h_1, h_2, \cdots, h_n\}$ is defined for $\boldsymbol{G}$, where $h_i = (0, 1, \cdots, x)$ denotes the computational unit selection for n_i. When $h_i = 0$, n_i is executed on VT, and when $h_i = 1$, n_i is offloaded to RSU m_x for execution.

What's more, for latency-sensitive applications, transmission interference in wireless channels is still an important issue that cannot be ignored [29]. In other words, transmission interferences may cause sudden channel changes during tasks offloading, resulting in task transmission failure and data retransmission. In this paper, we denote it as δ_h, which indicates the probability that the application does not need to retransmit. As shown in Eq. (5), δ_h decreases with the increase of κ_{n_i}, where κ_{n_i} refers to the communication hop number of n_i, which is equal to the size of $\Re_{n_i}$.

$$\delta_h = \begin{cases} \dfrac{\sum\limits_{i=0}^{N} [(d_i^t + d_i^r) \cdot \alpha_{i,j}^{n_i}]}{\sum\limits_{i=0}^{N} (d_i^t + d_i^r)}, \Re_{\mathrm{n}_i} = \emptyset; \\ \dfrac{\sum\limits_{i=0}^{N} [(d_i^t + d_i^r) \cdot \prod\limits_{m_x \in \Re_{\mathrm{n}_i}, j=0}^{\kappa_{\mathrm{n}_i}} \alpha_{j,j+1}^{n_i}]}{\sum\limits_{i=0}^{N} (d_i^t + d_i^r)}, else. \end{cases} \tag{5}$$

III. Problem Transformation

The computation offloading problem P1 is an integer nonlinear optimization problem with unknown applications and edge environments, which is difficult to solve. Considering the RSUs topologies and task dependency, we adopt the stationary decision to address this problem, which is time-invariant and only depends on the current system status. The state space, action space, and value are as follows:

a) State space $\boldsymbol{S}$: For applications, a feasible offloading strategy should be comprehensively determined by the configuration file, application topology and edge environment. The scale of $\boldsymbol{S}$ increases with the number of tasks and the scale of available edge units. Therefore, the state can be defined as:

$$\boldsymbol{S} = \{s_i | s_i = (\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{E}>, \boldsymbol{H}_{\boldsymbol{n_0:n_i}})\}. \tag{6}$$

b) Action space $\boldsymbol{A}$: For current states, offloading of tasks is a multivariate choice, and the scale of $\boldsymbol{A}$ is determined by the number of available edge units. The action space is defined as follows:

$$\boldsymbol{A} = \{a_0, a_1, a_2 \cdots a_i\}, i = |\boldsymbol{N}|, \tag{7}$$

where a_0 refers to the change of edge units of computation task n_i from m_i to m_{i+1}.

c) Reward r_{a_i}: r_{a_i} is defined as the target when the MDP system stays in state s_i with adopting action a_i. In problem P1, since the objectives is to find a strategy h that choose appropriate actions at different states to minimize the long-term target, which consist of the immediate target (generated in the current episodes) and the future target (generated in the following episodes) for each state-action pair.

In the training phase, similar to traditional RL-based algorithms, the controller initializes network parameters and ovserves the current MDP system state including the edge environment $\boldsymbol{D}$, the preprocessing application ***G'***, the number of episode ***I*** and the max number of iteration $\boldsymbol{F}$. Then the state vector is input to RL to train the learning model. The random action is chosen with probability ε based on the unknown knowledge to keep trying new actions for exploring higher target, while the action with maximum reward is selected probability $1 - \varepsilon$ according to the current information. After executing the selected action, the agent receives a reward from the environment and observes the state from s_t to the next state s_{t+1}. Then, updating the action-value function $Q_\theta(s_t, a_t; \theta)$ and target action-value function $\hat{Q}_\theta\left(s_t, a_t; \hat{\theta}\right)$, before collecting and soring the transition tuple $(\phi_j, a_j, r_{a_j}, \phi_{j+1})$ into the experience replay memory buffer P_i,which include the current system state vector,selected action, reward and the next state vector. The transitions in P_i is selected with mini-batches and they are used to train RL y_i. In detail, the priority of each transition y_i is calculated by using Eq. (8), where the priorities ensure that high-TD-value transitions are replayed more frenquently. Then, the loss function is calculated according to Eq. (9) and the parameters θ_i are updated according to Eq. (10). Once RL converges, the out-loop training is acheieved.

$$y_j = \begin{cases} r_j, s_j = s_{best}; \\ r_j + \gamma \cdot \max\limits_{a_{j+1}} \hat{Q}\left(\phi_{j+1}, a_{j+1}; \hat{\theta}\right), otherwise, \end{cases} \tag{8}$$

where γ refers to the discount factor of each reward received from the current state with respect to the target value y_j.

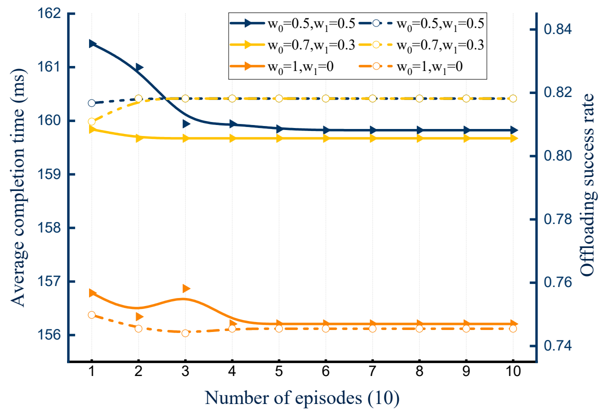
$$\begin{aligned} L_i(\theta_i) &= \mathop{\mathbb{E}}_{(\phi_t, a_t, r_{a_t}, \phi_{t+1})} (y_j - (Q(\phi_j, a_j; \theta_i))^2 \\ &= \mathop{\mathbb{E}}_{(\phi_t, a_t, r_{a_t}, \phi_{t+1})} [r_j + \gamma \cdot \max_{a_{j+1}} \hat{Q}\left(\phi_{j+1}, a_{j+1}; \hat{\theta}\right) \\ &\quad - Q(\phi_j, a_j; \theta)]^2. \end{aligned} \tag{9}$$

$$\begin{aligned} \nabla_{\theta_i} L_i(\theta_i) = \mathop{\mathbb{E}}_{\phi_t, a_t, r_{a_t}, \phi_{t+1}} &[(r_j + \gamma \cdot \max_{a_{j+1}} \hat{Q}\left(\phi_{j+1}, a_{j+1}; \hat{\theta}_i\right) \\ &- Q(\phi_j, a_j; \theta)) \nabla_{\theta_i} Q(\phi_j, a_j; \theta]. \end{aligned} \tag{10}$$

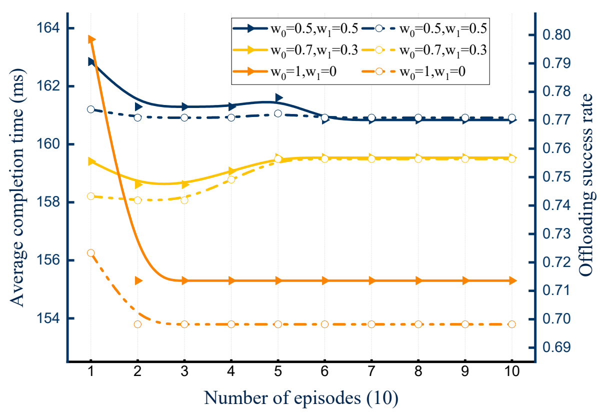
A. Experimental Settings

The simulation is conducted on a laptop equipped with 16 GB RAM and an AMD Ryzen 7-5800H CPU. The experiments are implemented in Python using the PyTorch framework for neural network construction and training. A vehicular terminal (VT) and several RSUs are considered, with communication ranges of 250 m and 500 m, respectively [27]. The computing capacities of the VT and RSUs are set to 1–2 GOPS and 10–15 GOPS, respectively, and the transmission power of both is 20 dBm.

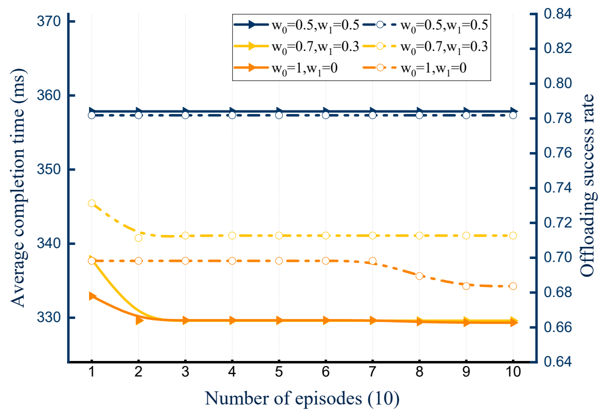
Fig. 1 shows the RL performance under different optimization weight settings over 100 episodes. The optimization objectives are the average completion time and the average offloading success rate. As shown in Figs. 1(b) and 1(c), both metrics generally improve as the number of episodes



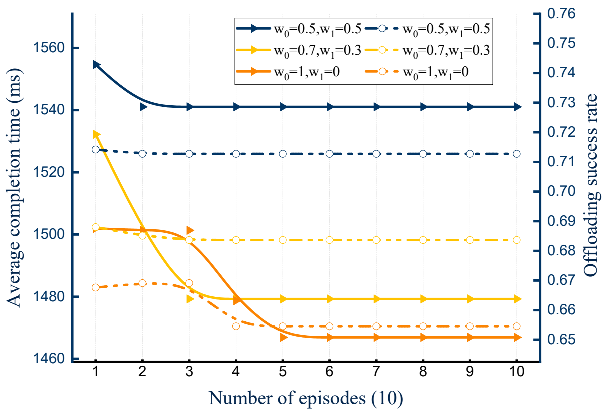
(a) GoogleNet



(b) ResNet101



(c) VGG19



(d) AlexNet

Fig. 1. Performance comparison of RL under different objective weights.

increases. This occurs because the algorithm gradually learns to assign computationally intensive tasks to more powerful RSUs, while parallelizable tasks are executed concurrently, improving overall efficiency.

It is worth noting that the average completion time does not always decrease monotonically. For instance, when the weight coefficients are set to $w_0 = 0.7$ and $w_1 = 0.3$, both metrics first decrease and then increase. This indicates that, after around the 30th episode, the algorithm finds it more beneficial to increase the offloading success rate at the expense of a slightly longer completion time, thereby achieving a higher overall reward.

Moreover, the experimental trends are positively correlated with the initial objective weights. Specifically, larger w_0 or w_1 values guide the algorithm to prioritize completion time or success rate optimization, respectively, demonstrating the flexibility and effectiveness of the proposed approach. It is also observed that the improvement magnitude of the two metrics is not linearly proportional to the weight values, likely due to differences in edge network topology and channel conditions. Therefore, for different applications, there exists an appropriate weight pair (w_0, w_1) that can balance the trade-off between completion time and offloading success rate.

IV. Conclusion

In this paper, we have proposed a efficient offloading strategy based on RL in the vehicle edge environment, which aims to reduce application completion time while improving its offloading success rate. The extensive simulation experiments show that the application completion time decrease and strategy offloading success rate increases as the number of iteration increases.

References

[1] M. Boban, A. Kousaridas, K. Manolakis, J. Eichinger, and W. Xu, "Connected roads of the future: Use cases, requirements, and design considerations for vehicle-to-everything communications," *IEEE Vehicular Technology Magazine*, vol. 13, no. 3, pp. 110–123, 2018.

[2] A. M. Vegni and V. Loscri, "A survey on vehicular social networks," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2397–2419, 2015.

[3] B. Ji, X. Zhang, S. Mumtaz, C. Han, C. Li, H. Wen, and D. Wang, "Survey on the internet of vehicles: Network architectures and applications," *IEEE Communications Standards Magazine*, vol. 4, no. 1, pp. 34–41, 2020.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5g: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[6] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.

[7] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[8] Y. Sun, S. Zhou, and J. Xu, "Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[9] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[10] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2021.

[11] H. Chen, S. Deng, H. Zhu, H. Zhao, R. Jiang, S. Dustdar, and A. Y. Zomaya, "Mobility-aware offloading and resource allocation for distributed services collaboration," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2428–2443, 2022.

[12] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial iot–edge–cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.

[13] W. Fan, D. Teng, F. Wu, and Y. Liu, "Pagerank-based multi-hop computation offloading in d2d networks," *IEEE Networking Letters*, vol. 2, no. 4, pp. 195–198, 2020.

[14] T. T. Vu, D. T. Ngo, M. N. Dao, S. Durrani, D. H. N. Nguyen, and R. H. Middleton, "Energy efficiency maximization for downlink cloud radio access networks with data sharing and data compression," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 4955–4970, 2018.

[15] C. Tang, X. Wei, C. Zhu, Y. Wang, and W. Jia, "Mobile vehicles as fog nodes for latency optimization in smart cities," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9364–9375, 2020.

[16] J. Baek and G. Kaddoum, "Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1041–1056, 2021.

[17] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2833–2849, 2020.

[18] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1133–1145, 2021.

[19] G. Zhang, S. Ni, and P. Zhao, "Learning-based joint optimization of energy-delay and privacy in multiple-user edge-cloud collaboration mec systems," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[20] X. Chen, M. Li, H. Zhong, Y. Ma, and C.-H. Hsu, "Dnnoff: Offloading dnn-based intelligent iot applications in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2820–2829, 2022.

[21] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for dnn-based smart iot systems in cloud-edge environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 683–697, 2022.

[22] G. Secinti, A. Trotta, S. Mohanti, M. Di Felice, and K. R. Chowdhury, "Focus: Fog computing in uas software-defined mesh networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2019.

[23] S. Misra and N. Saha, "Detour: Dynamic task offloading in software-defined fog for iot applications," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, 2019.

[24] X. Chen, J. Hu, Z. Chen, B. Lin, N. Xiong, and G. Min, "A reinforcement learning-empowered feedback control system for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2724–2733, 2022.

[25] Q. Liu, T. Xia, L. Cheng, M. van Eijk, T. Ozcelebi, and Y. Mao, "Deep reinforcement learning for load-balancing aware network control in iot edge systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1491–1502, 2022.

[26] M. Min, X. Wan, L. Xiao, Y. Chen, M. Xia, D. Wu, and H. Dai, "Learning-based privacy-aware offloading for healthcare iot with energy harvesting," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4307–4316, 2019.

[27] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9637–9650, 2020.

[28] F. Tang, B. Mao, N. Kato, and G. Gui, "Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 2027–2057, 2021.

[29] J. Johnston, Y. Li, M. Lops, and X. Wang, "Admm-net for communication interference removal in stepped-frequency radar," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2818–2832, 2021.

[30] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.

[31] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in iot edge computing," *IEEE Journal on Selected Areas in Communications*, vol. PP, no. 99, pp. 1–1, 2020.

[32] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 64–69, 2019.

[33] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–33, 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/25719670

[34] B. Lin, K. Lin, C. Lin, Y. Lu, Z. Huang, and X. Chen, "Computation offloading strategy based on deep reinforcement learning for connected and autonomous vehicle in vehicular edge computing," *Journal of Cloud Computing*, vol. 10, no. 1, 2021.