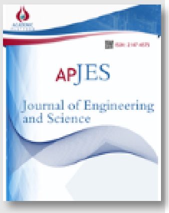
ACADEMIC PLATFORM

Academic Platform Journal of Engineering and Science

journal homepage: http://apjes.com/



Extending Wireless Sensor Networks' Lifetimes Using Deep Reinforcement Learning in a Software-Defined Network Architecture

*[1]Zainab Ali Abbood, [2]Mahmoud Shukr Mahmoud, [3]Çağatay Aydın, [4]Doğu Çağdaş Atilla,[5]Dr. Bassem G. Al-saadi
*[1]Altınbaş University, School of Engineering and Natural Sciences, Istanbul, Turkey,
zainab.al_mamoori@org.altinbas.edu.tr,
[2]Al-Mansour University College, Computer Technology Engineering Dept., Baghdad, Iraq,
mahmoud.shukur@muc.edu.iq,
[3]Altınbaş University, School of Engineering and Natural Sciences, Istanbul, Turkey,
cagatay.aydin@altinbas.edu.tr,
[4]Altınbaş University, School of Engineering and Natural Sciences, Istanbul, Turkey,
cagdas.atilla@altinbas.edu.tr,
[5]El-Esraa University College, Computer Technology Engineering Dept., Baghdad, Iraq,
dr.basimalsaady@gmail.com,

Research Paper Arrival Date: 11.02.2020 Accepted Date: 29.09.2020

Abstract

Routing packets in a Wireless Sensor Network (WSN) is a challenging task, according to the limited resources available on the nodes of these networks, especially their energy sources. The use of Machine Learning (ML) techniques in a Software-Defined Network (SDN) topology has shown a good potential toward solving such a complex task. However, existing techniques emphasize finding the shortest paths to deliver the packets, which can overload certain nodes in the network, depending on their positioning. In this study, a new method is proposed to extend the lifetime of the WSN by balancing the loading on the nodes, using a Deep Reinforcement Learning (DRL) approach. By emphasizing on the lifetime of the network, the proposed method has been able to discover and use alternative routes to deliver the packets, avoiding the use of nodes with low energy. Hence, the average number of hops the packets travel through has been increased but the time required for the first node to exhaust its energy has been significantly increased.

Keywords: Software Defined Network, Internet of Things, Wireless Sensor Network, Reinforcement Learning.

1. INTRODUCTION

With the rapidly growing interest in collecting different types of data from different environments, the need for flexible low-cost networks to collect and log these data has been increasing [1, 2]. As a solution, a set of small devices, i.e. sensors, that can measure the required data and communicate them through wireless connections has emerged as a solution for such tasks. These networks are known as Wireless Sensor Networks (WSNs), in which each sensor is considered as a node in the network that has the responsibility of delivering network packets from other nodes, in addition to the task it is designated for. Such a topology allows the WSN to work in any environment without the need for infrastructure and change the topology of the network as required by the environment it is being deployed in [3, 4].

With the absence of infrastructure and the changing topology of WSNs, routing the packets in the network to reach its destinations is a challenging task. This problem can become more complex when the nodes in the WSN are not stationary, i.e. mobile, as the route that is discovered between two nodes at a certain time instance becomes invalid as soon as one of the nodes in that route becomes out of the range of the remaining nodes in that route. Accordingly, several of the recent studies rely on reactive routing methods [5-7], in which a route is discovered on demand and set to be valid for a specific interval of time. When the route becomes expired, new route discovery is initiated [8, 9].

*[1] Corresponding Author: Altınbaş University, School of Engineering and Natural Sciences, Istanbul, Turkey

Doi: 10.21541/apjes.687496

According to the massive amount of traffic to flow through the network when such reactive methods are used, such as the Ad hoc On-Demand Vector (AODV) and Dynamic Source Routing (DSR), and the limited resources available on the nodes of a WSN, the use of these protocols can be exhaustive to the nodes [10, 11]. Additionally, these protocols consider only the length of the path from source to destination, i.e. finding the shortest route, without considering other factors, such as the energy remaining on these nodes. Reactive routing protocols rely on propagating a Route Request (RREQ) to all the nodes in the WSN, i.e. each node forwards this packet to all adjacent nodes once, until the destination node is reached. The destination then sends a Route Reply (RREP) packet to the sources indicating the shortest route between the nodes, which is used by the source node to send its payload packets [11, 12].

Optimizing the routes that the packets in the WSN travel to reach their destination can significantly improve the performance of the WSN, by optimizing the use of the limited resources available on the nodes of the network. Hence, Artificial Intelligence (AI) and Machine Learning (ML) techniques have been introduced to handle the dynamic nature of these networks' topologies. Such employment has produced the Intent-Driven Networks (IDNs) or Intent-Based Networks (IBNs), which has the ability to produce the networks' configurations from the business requirements. Additionally, such implementation requires the use of Software-Defined Network (SDN) to allow a central controller making the decisions required to optimize the operation of the network, based on the task requirements [13].

Mainly, the role of the SDN controller is to control the flow of packets in the network, i.e. selecting the next hop of a packet depending on the characteristics of that packet and the current state of the network. Hence, the overall performance of the network is defined based on the decisions made by the controller. Several measures are used to illustrate the performance of the network, such as the average number of hops packets travel through to reach their destinations and the lifetime of the network, depending on the energy that exists on each node on the network and the power consumption at that node [14]. In recent years, the use of devices with smaller resources, including energy sources, has been rapidly increasing, which has produced the era of the Internet of Things (IoT) [15].

According to the limited resources of IoT devices and the dynamic nature of IoT networks, the use of SDN architecture to control the flow of traffic has improved the overall performance of the network [16]. According to the limited energy available for these devices, overloading a certain device, that may be located in a vital position compared to the distribution of the nodes in the network, can drain the power of that device. Hence, the communications to, from and through that device are interrupted [17]. Hence, balancing the loading over the nodes can significantly extend the lifetime of the network [18].

Machine learning techniques gain knowledge about the environment from examples collected from that environment. Mainly, three types of ML techniques exist, which are supervised, unsupervised and Reinforcement Learning (RL). Unlike the other types, RL gains knowledge by directly interacting with the environment and collecting feedbacks to measure the quality of the executed actions. These feedbacks, known as rewards, are used to approximate the behavior of the environment, so that, the actions that can maximize the rewards can be selected by the RL technique. Thus, a generic RL model consists of the agent that executes the actions in the environment and the environment which returns the reward of the executed action [19, 20].

According to the good performance of Artificial Neural Networks (ANNs) in approximating the computations of any function, these networks are being widely used to approximate the function of the environment that the agent is interacting with [21, 22]. As the actual behavior of the environment is unknown to the agent, a neural network is used to approximate the environment, so that, the reward for each action selected by the agent can be estimated prior to the execution in order to select the action that is estimated to return the highest possible reward. Deep Reinforcement Learning (DRL) uses deep neural networks for this purpose, which has shown significantly better performance, compared to other RL methods [22, 23].

DRL has been employed by Zhang et al. [24] for an SDN controller that controls the flow of packets in Vehicular ad hoc networks (VANETs). According to the continuous movement of the nodes in VANETs, this framework emphasizes finding the routes that increase the probability of delivering the packet to its destination, without considering the power consumption of the network. Thus, the performance of the network is measured by the Packet Delivery Rate (PDR) and the average networks' throughput. However, the results show that the use of Convolutional Neural Networks (CNNs) has achieved the highest performance, compared to the user of networks that use only fully-connected layers.

Another method that uses DRL is proposed by Lin et al. [25], which relies on the Quality of Service (QoS) of the network as the reward for the neural network, hence, denoted as QoS-aware Adaptive Routing (QAR). However, this method measures the QoS using only the packet loss, delay and throughput and also neglects the lifetime of the network, i.e. does not consider the power consumption. The results of the experiments conducted in this study show that the use of a higher discount factor, i.e. Gamma, has reduced the average number of hops required to deliver the packet. This discount factor is used to reduce the effect of the reward collected by the end of a series of actions over each action, depending on its position in the series, so that, higher discount rate indicates more effect on earlier actions. Similarly, Stampa et al. [26] use a DRL model for the SDN controller, which focuses on reducing the time required to deliver the packets to their destination. Hence, the DRL agent is only trained to recognize the shortest possible path between the source of the packet and its destination.

Despite the good performance of these methods, regarding the delay or number of hops required to deliver a packet, the negligence of power consumption can exhaust one of the nodes in the network. For instance, when the existing methods are used to route packets in the sample network as shown in Figure 1, all packets between the nodes in the black and blue subnets are going to be routed through the red node, as routing them through the green nodes reduces the expected reward according to the longer path, i.e. an additional hop. Hence, the energy in the power source of the red node is going to be exhausted and the node is expected to have less uptime, which reduces the lifetime of the network.

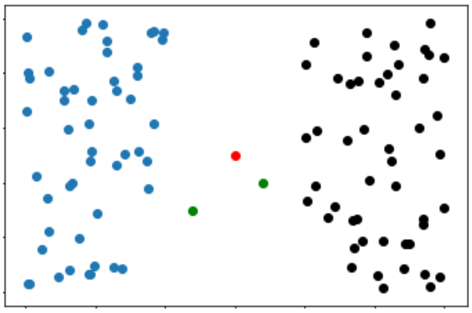


Figure 1. Sample wireless sensor network.

In this study, a new routing method is proposed for WSNs that use SDN topology. The proposed method is based on DRL, in which the lifetime of the WSN is included in the training of the DRL agent, so that, the efficiency of the WSN is improved, as the extension of the lifetime requires more efficient resources consumption from the nodes. According to their ability to process and output multi-dimensional arrays, two types of convolutional neural networks are used in the proposed method. One uses two-dimensional convolutional layers and the other uses three-dimensional. Despite the additional complexity that the use of the 3D convolutional layers imposes, the proposed method use these filters to summarize the characteristics of each node, represented by the corresponding feature values, into a single value, which can be used to provide more-accurate predictions. To improve the efficiency of the WSN and increase its lifetime, these models are required to avoid exhausting the nodes in the network by balancing the loading among the nodes and use alternative routes, which may not be the shortest, to avoid the use of nodes with very low energy remaining.

2. MATERIALS AND METHOD

The proposed method uses a deep neural network to estimate the reward expected for delivering the packet to the next hop, depending on the characteristics of the network and the packet's source and destination nodes. According to the good performance of the CNN, shown by Zhang et al. [24], the proposed method also uses this type of neural network for the required task. However, as there are two types of CNN layers, 2D and 3D, both types are implemented and evaluated in this study. However, to allow the use of the proposed method in all networks, regardless of their topologies, the output of the neural network is set to be a two-dimensional matrix, where each value in this matrix represents the estimated reward if the packet is forwarded to the node in the corresponding position.

The information of the network and the packet are distributed in a three-dimensional array, which has $100\times100\times5$ size. The positioning of the hosts is scaled to 100×100, regardless of the actual size of the environment, to maintain the simplicity of the neural network and allow the adoption of any possible topology of the networks. Each of the four layers in the input contains the values of one of the following information, each value is mapped based on the position of the host the value is corresponding to:

1. The remaining energy of each node.
2. A value of one corresponding to the position of the source host.
3. A value of one corresponding to the position of the destination host.
4. Value of ones positioned at the positions of the nodes that are within the range of the node that the packet is currently at.

5. Route description for the hosts that the packet has been through up to the current hop, where the source node is assigned with zero while the current host is assigned with one. Other hosts that the packet has passed through are assigned with value depending on the sequence of hosts in the route, lower values are assigned to the hosts the packet has passed through earlier. These values are calculated using the formula shown in Algorithm 1.

Algorithm 1: Packets hops representation algorithm.

```
Input: Hops' list of a packet; Position of nodes.
Output: Two-dimensional representation of the hops list.

Step1:  H ← Read hops list. L ← Length(H)
        R ← Full(100×100, -1). //A 100×100 array with
        the value -1 for the output.
Step2:  For i = 1 to L:
          p = H(i).position //Find the position of the node
                              of the current hop.
          R[p] ← i/L  //Place the ratio between the
                        position of the node in the path
                        to the length of the route
                        according to the mapped
                        position of the node.
Step3:  Return R
```

For instance, a packet initiated from the node *A* and has passed through the nodes *B, D, H* and *M* has a hops list *[A, B, D, H, M]*, as it has passed through these nodes in this order. According to Algorithm 1, a 100×100 matrix is initiated and filled with that value -1. Then, the values *[0, 0.25, 0.5, 0.75, 1]* are placed in the positions that the nodes *[A, B, D, H, M]* map to, according to their actual position in the environment. This representation allows the neural network to recognize the position of each node the packet has been through and the order it has been passing from one node to another.

2.1.The 2D CNN Model

The model implemented using the 2D CNNs, shown in Table 2, is used to predict the reward of forwarding the packet for each node in the network. The output of the neural network is identical to the dimensions of the network, except that it contains a single layer, i.e. two-dimensional. The value per each position represents the reward estimated by the model when the packet is forwarded to that node. However, according to the possibility that the maximum reward does not map exactly over an existing node, the closest node to the position of the maximum reward is selected.

Table 1. Structure of the implemented 2D CNN model.

Layer Type	Filter Size	Number of flters	Output Shape
Conv2D	(2×2)	32	(100×100×32)
Conv2D	(2×2)	16	(100×100×16)
Conv2D	(3×3)	8	(100×100×8)
Conv2D	(5×5)	4	(100×100×4)
Conv2D	(5×5)	1	(100×100×1)

2.2.The 3D CNN Model

Table 2. Structure of the implemented 3D CNN model.

Layer Type	Filter Size	Number of flters	Output Shape
Conv3D	(1×1×5)	32	(100×100×5×32)
Average Pooling	(1×1×5)	-	(100×100×1×32)
Conv3D	(2×2×4)	16	(100×100×1×16
Conv3D	(3×3×4)	8	(100×100×1×8)
Conv3D	(5×5×4)	4	(100×100×1×4)
Conv3D	(5×5×4)	1	(100×100×1×1)

Unlike the filters in 2D CNN layers, which can detect features in a single layer of the input array, the filters in the 3D CNN layers can detect features the combine values from multiple layers of the input, i.e. combine values from the third dimension of the input. Hence, a 3D average pooling layer is placed after the first 3D convolutional layer to summarize the values calculated by that layer into a single value per each node position. The existence of a 3D convolutional layer before the average pooling layer allows the neural network to adjust the effect of each piece of information over the value produced for that node. Hence, the filter is the following convolutional layer can detect features that represent the overall characteristics of the node, instead of a single characteristic in the 2D model.

2.3.Training the DRL Model

Initially, the neural network has no knowledge about the rewards it can get for each action. Hence, the packets are forwarded in a random manner, so that, the reward returned by the network based on the selected action, i.e. next hop, is used to train the neural network. After a few iterations, the neural network starts to gain knowledge about the environment and how to deliver the packets from one node to another. However, this knowledge can be limited to the approaches recognized during the use of random actions. For example, the neural network may start to learn to deliver the packet to the destination node using the shortest path but still unable to extend the lifetime of the network. Thus, a fraction of the decisions is still required to be executed randomly in order to balance exploration and exploitation. Thus, a variable with a value equal to one is set at the first iteration and compared to randomly generated numbers in the interval [0,1], so that, if the random number is greater than the value of the variable, the action is selected based on the output of the neural network. Otherwise, the action is selected randomly. This value of this variable is reduced by multiplying it to 0.99 after each iteration, so that, the number of actions selected based on the predictions of the neural network is increased as the knowledge of the neural network increases.

Per each iteration, the training of the neural network is continued until the energy of one of the network's nodes is drained. Then, the lifetime of the network is used to update the reward values of the neural network. However, as the delivery of each packet in the network is not related to other packets, the lifetime of the network is used to update the reward values of each packet solely, i.e. the packets deliveries are considered parallel operations rather than serial and the reward value is assigned for the last action or hop. This value is reduced using the discount factor (Gamma), which is set to 0.9, as higher values for the discount factor have shown better performance in [27].

The predictions of the neural network are updated using the formula shown in Equation 1, where Q is the predicted reward value for executing action a in state s. R is the actual reward value retrieved from the environment after executing the action, maxQ' is the maximum reward expected from the agent after being in the new state s', i.e. after forwarding the packet to the next hop.

$$New\ Q(s,a) = Q(s,a) + \propto (R(s,a) + \gamma max Q'(s',a') - Q(s,a)) \quad (1)$$

As the value computed using this formula represents only the reward value of the node the packet is forwarded to, the reward values for the other nodes are maintained as predicted by the neural network. Using such an approach, any prior knowledge is maintained and the knowledge extraction can continue even when random actions are selected. However, to present the knowledge required by the neural network to avoid forwarding packets to positions that do not nodes in them, reward values of -1, i.e. punishments, are placed in the positions that have no nodes in them. This training procedure is conducted after the first node in the network is exhausted. However, instant training occurs when one of the following conditions occurs:

- The packet is forwarded to a node that is out of the transmission range of the current node.
- The packet is forwarded to a node that does not have sufficient power to receive or forward the packet unless it is the destination node.
- The packet is forwarded to a node that is in the list of hops that the packet has been through, to avoid infinite loops.
- The number of hops the packet passes through exceed 10 times the number of nodes in the network.

3. PERFORMANCE EVALUATION

In order to train and evaluate the proposed method, a simulation of WSNs is implemented using Python programming language [28] with a Windows computer running using an Intel® Core™ i7-7700 CPU of 2.8GHz frequency and 16GB of memory. The neural network is implemented using Keras library [29] on top of the Tensorflow [30] machine learning library. The implemented WSNs consist of random numbers of nodes, varying from 8 to 32, distributed randomly in a $1000\times1000m^2$ area. The packet size is set to 1024 bytes with 2Mbps data rate. Each node is initiated with 1 joule of energy and consumed 50×10^{-9} joule per each packet forwarded or received. The range of the nodes is set to 300 meters, where nodes with distances larger than this distance are considered unreachable. Each node consumes 10-10Joule/sec while in idle mode, i.e. not sending or receiving packets.

The neural network is trained using 100 randomly generated WSNs, where packets are randomly generated per each network until one of the hosts is exhausted. To ensure consistency among the generated WSNs and packets, to avoid biased evaluation, a random seed is used to ensure generating the same random sequences of values among the different types of neural networks. The performance of the SDN controlling method is evaluated based on the average number of hops required to deliver the packets, the average lifetime of the WSNs and the time required per each decision-making process, i.e. next hop selection. The performance is evaluated using a set of 10 randomly generated WSNs, different from those used in the training. The performance of the proposed method, using both types of CNN, is shown in Table 3 and compared to the state-of-the-art methods from the literature.

Table 3. Performance measures of the evaluated methods.

Method	**Average hops**	**Average lifetime (s)**	**Prediction time (*us*)**
2D-CNN	12.37	638169.21	316.03
3D-CNN	9.81	678251.62	351.71
Stampa et al. [26]	8.76	520364.41	283.10
Lin et al. [27]	9.32	578122.16	328.11
Zhing et al. [24]	10.53	541839.76	341.87

As the results in Table 4 show, the proposed method using the 3D-CNN model has achieved a significantly longer lifetime, especially when compared to the method in earlier studies. However, the methods proposed by Lin et al. [27] and Zhing et al. [24] have lower average number of hops packets pass through to reach their destinations. Such increment in the number of hops is an expected behavior as the proposed method searches for paths that maximizes the lifetime of the network rather than the shortest path. The method proposed by Stampa et al. [26] has the lowest average number of hops as this method emphasizes mainly on finding the shortest path in the network, which illustrates the capabilities of DRL in achieving the required tasks. Moreover, these results prove the hypothesis of this study that relying on the lifetime of the network in routing packets can balance the length of the paths the packets travel through and the energy consumption in the network. Thus, the proposed method can significantly improve the lifetime of IoT devices in WSNs.

Additionally, the average minimum power of the nodes, in the 10 WSNs that are used for the evaluation, is monitored during the operation of these networks. Per each second, the least remaining energy in any of the WSN's nodes is logged and averaged for each method, as shown in Figure 2. This comparison shows that the proposed method has been avoiding loading nodes with lower energies in order to extend the lifetime of the network. Moreover, Figure 2 also shows that the 3D-CNN has been able to consider such loading before the nodes start to ge exhausted, which in return has been able to significantly improve the overall lifetime of the WSN, as shown in Figure 3.

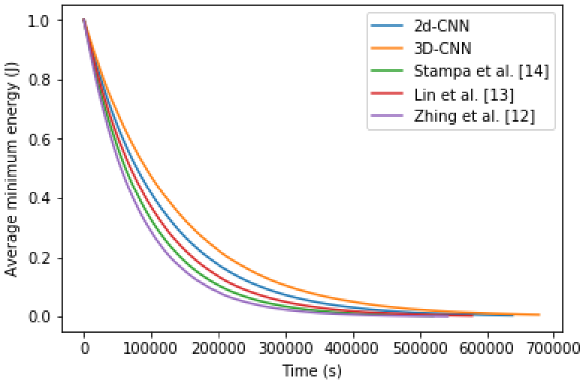


Figure 2. Average minimum node energy.

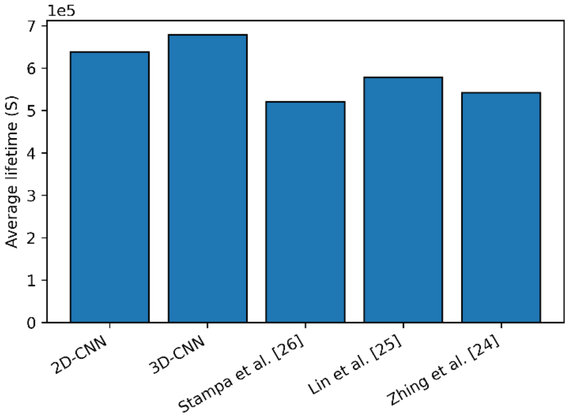


Figure 3. Average lifetime of the WSNs using existing and proposed methods.

Despite the ability of the 2D-CNN to improve the lifetime of the WSN, compared to the existing methods, the use of the 3D-CNN has shown significantly better improvement, as shown in Figure 3. Combined with the behavior of the WSN when using the 3D-CNN methods, shown in Figure 2, these results show that the earlier consideration that the 3D-CNN has been able to achieve has also been able to extend the lifetime of the network furthermore, compared to the use of the 2D-CNN. Additionally, the ability of the 3D-CNN to summarize the different input factors, after adjusting their values, to produce a single value per each node, according to the topology shown in Table 2, this neural network has been able to provide better predictions, in terms of finding the routes that can extend the lifetime of the WSN. However, the extension in the lifetime by using the proposed method, i.e. by avoiding the exhaustion of certain nodes, indicates that the packet are being forced to travel longer paths to reach their destination. Thus, the average number of hops that the use of each method has produced is measured and illustrated in Figure 4.

According to the results shown in Figure 4, the average number of hops required by the 2D-CNN is significantly higher than both the existing methods and the use of 3D-CNN. Such a behavior imposes a limitation toward the use of this neural network, especially in applications that require faster delivery of packets, as the additional hops in the route indicate a definitely longer route. Nevertheless, this behavior is expected from the proposed method to avoid the use of exhausted nodes and cannot be considered as a limitation if it was not for the performance of the proposed method using the 3D-CNN. This neural network has been able to produce more-efficient routing, in terms of resources consumption, and maintain similar average of number of hops, compared to the existing methods. However, as shown in Figure 5, despite the longer paths that are used by the 2D-CNN, this method has been able to maintain high Packet Delivery Rate (PDR), compared to existing state-of-the-art methods. This indicates that these paths are still valid, despite being longer than those predicted using existing methods. Additionally, the use of 3D-CNN has been able to achieve higher PDR than any other methods, which indicates that this neural network has better consideration of the overall performance of the network, as the longer paths in the 2D-CNN have reduced the overall lifetime of the WSN, as shown in Figure 3.

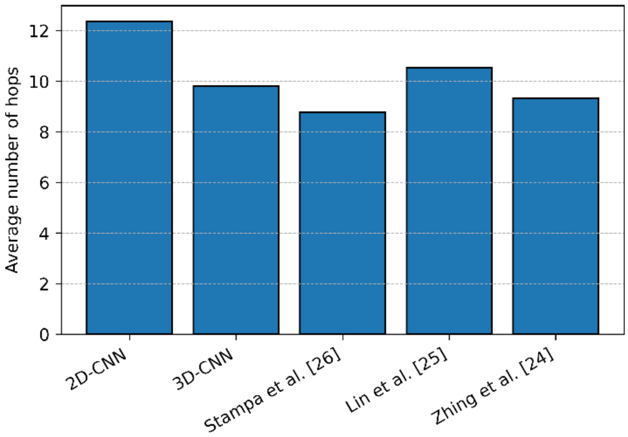


Figure 4. Average number of hops required to deliver the packets.

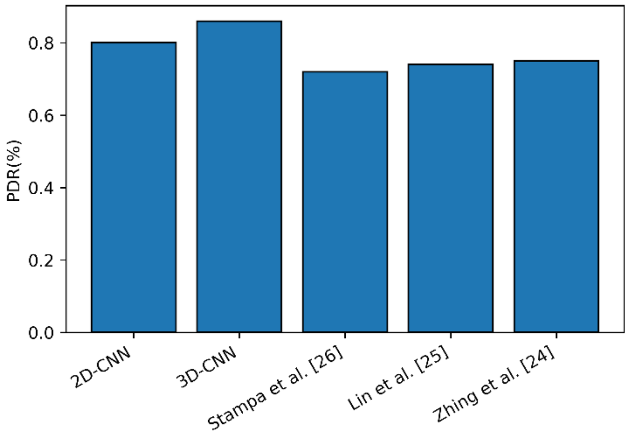


Figure 5. Average number of hops required to deliver the packets.

4. CONCLUSION

In this study, DRL is used to control the flow of packets in a WSN using SDN architecture. The proposed method aims to extend the lifetime of the network by balancing the loading among the network's nodes, to avoid exhausting certain nodes. According to the limited resources available on each node in the WSN, this balancing can significantly improve the lifetime of the node, which allows it to achieve more of its required task, i.e. collecting more data. This increment in lifetime is achieved by reducing the amount of traffic the nodes in the WSN are required to handle in order to route the payload packets that transfer the actual data that represent the measured values. Two CNN models are evaluated in this study, by using the 2D and 3D convolutional layers. The results show that the use of 3D layers has achieved better performance, according to the ability of these layers to detect features that combine different types of characteristics. Moreover, the proposed method has shown significant extensions in the lifetime of WSNs, compared to the existing state-of-the-art methods. However, this lifetime extension increases the average number of hops the packets travel through in order to reach their destinations. This increment in the number of hops is a result using alternative routes, rather than the shortest ones, to avoid exhausting the energy sources of certain nodes.

In future work, the use of a separate model for each node in the network is going to be evaluated and compared to the use of a central SDN. Using such an approach, each node can automatically select the next hop without contacting the SDN controller, which can reduce the amount of information being communicated with the controller. However, according to the high resources required to conduct the mathematical operations in neural networks, it is possible that the resources required to communicate the information remain less than that required to execute the computations locally.

REFERENCES

[1].R. Vijayashree and C. Suresh Ghana Dhas, "Energy efficient data collection with multiple mobile sink using artificial bee colony algorithm in large-scale WSN," *Automatika,* vol. 60, pp. 555-563, 2019.

[2].M. Krishnan, S. Yun, and Y. M. Jung, "Dynamic clustering approach with ACO-based mobile sink for data collection in WSNs," *Wireless Networks,* vol. 25, pp. 4859-4871, 2019.

[3].T. Wang, J. Zeng, Y. Lai, Y. Cai, H. Tian, Y. Chen*, et al.*, "Data collection from WSNs to the cloud based on mobile Fog elements," *Future Generation Computer Systems,* vol. 105, pp. 864-872, 2020.

[4].S. K. Singh and P. Kumar, "A comprehensive survey on trajectory schemes for data collection using mobile elements in WSNs," *Journal of Ambient Intelligence and Humanized Computing,* vol. 11, pp. 291-312, 2020.

[5].M. Anand and T. Sasikala, "Efficient energy optimization in mobile ad hoc network (MANET) using better-quality AODV protocol," *Cluster Computing,* vol. 22, pp. 12681-12687, 2019.

[6].P. Gupta, P. Goel, P. Varshney, and N. Tyagi, "Reliability factor based AODV protocol: prevention of black hole attack in MANET," in *Smart Innovations in Communication and Computational Sciences*, ed: Springer, 2019, pp. 271-279.

[7].V. Sharma, B. Alam, and M. Doja, "An improvement in dsr routing protocol of manets using anfis," in *Applications of Artificial Intelligence Techniques in Engineering*, ed: Springer, 2019, pp. 569-576.

[8].Z. Al Aghbari, A. M. Khedr, W. Osamy, I. Arif, and D. P. Agrawal, "Routing in Wireless Sensor Networks Using Optimization Techniques: A Survey," *Wireless Personal Communications,* pp. 1-28, 2019.

[9].V. K. Quy, N. T. Ban, V. H. Nam, D. M. Tuan, and N. D. Han, "Survey of recent routing metrics and protocols for mobile Ad-hoc networks," *Journal of Communications,* vol. 14, pp. 110-120, 2019.

[10].K. L. Arega, G. Raga, and R. Bareto, "Survey on Performance Analysis of AODV, DSR and DSDV in MANET," 2020.

[11].N. E. Majd, N. Ho, T. Nguyen, and J. Stolmeier, "Evaluation of parameters affecting the performance of routing protocols in mobile ad hoc networks (MANETs) with a focus on energy efficiency," in *Future of information and communication conference*, 2019, pp. 1210-1219.

[12].S. K. Singh and J. Prakash, "Energy Efficiency and Load Balancing in MANET: A Survey," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 832-837.

[13].Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A Survey of Networking Applications Applying the Software Defined Networking Concept Based on Machine Learning," *IEEE Access,* vol. 7, pp. 95385-95405, 2019.

[14] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan*, et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine,* vol. 51, pp. 36-43, 2013.

[15].F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato, "An inelligent traffic load prediction-based adaptive channel assignment algorithm in SDN-IoT: A deep learning approach," *IEEE Internet of Things Journal,* vol. 5, pp. 5141-5154, 2018.

[16].M. Ojo, D. Adami, and S. Giordano, "A SDN-IoT architecture with NFV implementation," in *2016 IEEE Globecom Workshops (GC Wkshps)*, 2016, pp. 1-6.

[17].M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, "Evolving SDN for low-power IoT networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 71-79.

[18].J. Wu, S. Luo, S. Wang, and H. Wang, "NLES: A novel lifetime extension scheme for safety-critical cyber-physical systems using SDN and NFV," *IEEE Internet of Things Journal,* vol. 6, pp. 2463-2475, 2018.

[19].L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators* vol. 39: CRC press, 2010.

[20].M. L. Littman, "Markov games as a framework for multi-aent reinforcement learning," in *Machine learning proceedings 1994*, ed: Elsevier, 1994, pp. 157-163.

[21].J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks,* vol. 61, pp. 85-117, 2015.

[22].D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche*, et al.*, "Mastering the game of Go with deep neural networks and tree search," *nature,* vol. 529, pp. 484-489, 2016.

[23].V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare*, et al.*, "Human-level control through deep reinforcement learning," *nature,* vol. 518, pp. 529-533, 2015.

[24].D. Zhang, F. R. Yu, and R. Yang, "A machine learning approach for software-defined vehicular ad hoc networks with trust management," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1-6.

[25].S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "QoS-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *2016 IEEE International Conference on Services Computing (SCC)*, 2016, pp. 25-33.

[26].G. Stampa, M. Arias, D. Sánchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," *arXiv preprint arXiv:1709.07080,* 2017.

[27].M. A. Alsheikh, D. Niyato, S. Lin, H.-P. Tan, and Z. Han, "Mobile big data analytics using deep learning and apache spark," *IEEE network,* vol. 30, pp. 22-29, 2016.

[28].B. Rhodes, J. Goerzen, A. Beaulne, and P. Membrey, *Foundations of Python network programming*: Springer, 2014.

[29].F. Chollet, "Keras: The python deep learning library," *ascl,* p. ascl: 1806.022, 2018.

[30].M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean*, et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265-283.