Internet of Things 20 (2022) 100597

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

Deep Q learning based secure routing approach for OppIoT networks

Nisha Kandhoul [a,*], Sanjay K. Dhurandher [b]

[a] Division of Information Technology, NSIT, University of Delhi, New Delhi, India
[b] Department of Information Technology, Netaji Subhas University of Technology, New Delhi, India

ARTICLE INFO

Keywords:
OppIoT
Reinforcement learning
Deep learning
Deep Q-learning
Markov decision process
Sinkhole attack
Hello flood attack
Distributed denial of service attack

ABSTRACT

Opportunistic IoT (*OppIoT*) networks are a branch of *IoT* where the human and machines collaborate to form a network for sharing data. The broad spectrum of devices and ad-hoc nature of connections, further alleviate the problem of network and data security. Traditional approaches like trust based approaches or cryptographic approaches fail to preemptively secure these networks. Machine learning (*ML*) approaches, mainly deep reinforcement learning (*DRL*) methods can prove to be very effective in ensuring the security of the network as they are profoundly capable of solving complex and dynamic problems. Deep Q-learning (*DQL*) incorporates deep neural network in the Q learning process for dealing with high-dimensional data. This paper proposes a routing approach for *OppIoT*, *DQNSec*, based on *DQL* for securing the network against attacks viz. sinkhole, hello flood and distributed denial of service attack. The actor–critic approach of *DQL* is utilized and *OppIoT* is modeled as a Markov decision process (*MDP*). Extensive simulations prove the efficiency of *DQNSec* in comparison to other *ML* based routing protocols, viz. *RFCSec*, *RLProph*, CAML and *MLProph*.

1. Introduction

Mobile Ad-hoc Networks (*MANET*) [1] are the traditional networks where messages are transmitted between devices if an end-to-end connection exists between them. There exists certain situations where ensuring this end-to-end connection is not possible, such as in rural areas, disaster management scenarios, where the application of *MANETs* is not an option. A subclass of Delay Tolerant Networks (*DTNs*) [2], Opportunistic Networks (*OppNets*) [3] are suitable for use in such conditions where communication opportunities are rare and an end-to-end path may never exist between the source and the destination. This characteristic of *OppNets* applies to Internet of Things (*IoT*) networks [4] as well. *IoT* networks comprise of a large spectrum of devices that are connected to the internet. Opportunistic Internet of Things (*OppIoT*) networks [5] bring the benefits of *OppNets* and *IoT* together and perform data transmission using the contact opportunities between human beings and their smart devices. Routing of the messages is performed making use of the store-carry-forward principle, where the carrier node waits for another node that brings the message closer to the destination node.

The vast spectrum of devices and the broadcast nature of transmission gives rise to enormous security issues. Traditional security measures viz. trust schemes, cryptography etc. [6] are not able to effectively deal with threats to the security of the network. Machine learning techniques can prove to be a really efficient [7] in such scenarios. Indeed, Reinforcement Learning (*RL*) [8], which is a sub-class of *ML*, resembles human learning as using experience gained via exploitation and exploration, it learns on its own in an

* Corresponding author.
E-mail addresses: nishakandhoul1990@gmail.com (N. Kandhoul), dhurandher@gmail.com (S.K. Dhurandher).

https://doi.org/10.1016/j.iot.2022.100597
Received 3 July 2022; Received in revised form 6 August 2022; Accepted 6 August 2022
Available online 12 August 2022
2542-6605/© 2022 Elsevier B.V. All rights reserved.

unknown environment. *RL* models an agent or player that takes sequential actions or decisions, with or without prior knowledge of the environment, for achieving optimal output, which is associated with rewards. *Q-learning* [9] is a form of model-free *RL* approach.

Recently, deep learning concepts have been included into *RL* methods for solving several complex problems. Deep Q-learning network (*DQN*) [10] is one such network that combines deep neural network (*DNN*) [11] and *Q-learning*. *DQN* can follow any of the below approaches for learning procedure:

- **Value-based method:** The aim of value-based methods is to find an optimal value function by evaluating the goodness of each action. These methods are ineffective as the number of actions or states rises.
- **Policy-based method:** A policy is randomly chosen initially and then in every evaluation step, the value function is computed for that policy. Then transitions are made with the intention of finding an optimal policy. This approach suffers from huge fluctuation.
- **Actor–critic method:** This approach brings the advantages of both the above defined approaches thereby removing their shortcomings. The actor continuously tries to improve a policy by taking feedback from the critic until he reaches to an optimal policy. Asynchronous advantage actor–critic (*A3C*) [12] is an example of such methods. *A3C* has multiple agents and a global network. Each agent interacts with its own copy of environment which it resets to match the global network in the beginning. The global network is updated asynchronously using the gradients obtained from these agent's learning processes. Multi- agent approach helps in diversifying and speeding up the learning process.

The increasing number of devices that are getting connected to the *OppIoT* networks has led to an increase in the number of complex security threats [13]. *DQN* are suitable for solving such complex problems. The learning process is modeled as a Markov decision process (*MDP*) [14] where the *OppIoT* is assumed to be composed of a set of all possible states that the nodes in the network can be in. These states are defined using node's contextual information or using certain features. The trace data of the system calls are fed into *MDP*, whose state values are used for detection of abnormal behaviors [15] of the nodes. The attack detection problem is thus converted to a state- value prediction job making use of the Markov chains. Detection techniques can be broadly categorized as network-based or host-based. Network-based techniques rely on a specialized node for collection and analysis of packets transmitted in the network. Host-based techniques analyze the log files of the nodes for differentiating normal behavior from anomalous ones. Both these above mentioned techniques use either anomaly-based or signature-based detection methods [16]. Anomaly detection techniques observe the behavior of the nodes and raises an alert if deviations are detected from expected behavior. Signature detection techniques on the other hand save the patterns of several known attacks and later on compares them with the characteristics of possible attackers. The presented approach is a host based anomaly detection technique.

This work presents an actor–critic Deep Q-learning based secure routing approach for *OppIoT*, *DQNSec*, that protects the network from hello flooding [17], sinkhole [18] and distributed denial of service attack [19]. While performing hello flood attack, the attacker floods the network with huge traffic making the services unavailable to the legitimate nodes of the network. During sinkhole attack, an attacker compromises a node and uses it for sending fake routing information to its neighboring nodes. Attacker attracts the network traffic towards itself by advertising as having the shortest distance path to the destination node. Later this data can be used for launching several other attacks. Denial of service (*DoS*) attacker aims at preventing the legitimate users of a service from using the desired resources. Distributed denial-of-service (*DDoS*) is a form of *DoS* attack that is distributed in nature, affects a larger volume of traffic and targets huge number of nodes.

The major contributions of the proposed *DQNSec* scheme are:

- **Actor–Critic based deep Q- learning:** Actor–critic approach has the advantages of both the value-based and policy-based methods. The actor generates an optimal policy by receiving feedback from the critic.
- **Markov Decision Process:** The learning process follows Markov decision process (MDP) where the *OppIoT* is assumed to be composed of a set several states that are defined using node's contextual information for detection of abnormal behaviors of the nodes.
- **Detection and isolation of hello flooding, sinkhole and distributed denial of service attack:** *DQNSec* successfully detects and isolates the sinkhole, DDoS and hello flooding attackers from participation in the packet forwarding procedure.
- **Simulation on real data trace:** *DQNSec* is simulated using the real data trace and satisfactory results were obtained, the attackers were detected successfully.

The rest of this paper is organized as follows. Literature review is presented in Section 2. The proposed *DQNSec* scheme is described in Section 3. Section 4 provides the details of the simulations conducted. Section 5 concludes the proposed work and throws light on the future aspirations.

2. Literature review

RL, typically Deep Q- Learning (*DQL*) has been used in a wide range of applications in the literature, varying from privacy of data and users to protection of the critical infrastructure.

Yavuz et al. [20] proposed a deep learning based approach for detection of routing attacks using big data. Proposed approach showcased high scalability and generalization. The routing attacks viz. version number, decreased rank and hello-flood attack were successfully detected by this approach. The major disadvantage of this approach were that the attack datasets were generated via simulation and real data trace were not used. *Jinarajadasa et al.* [21] proposed a *RL* based security approach for enhancing the trust level in *MANETs*. Trust was computed using Q-value that served for calculating the reputation of the nodes, identifying malicious

behavior and finding routes for creating a trusted network infrastructure. As the size of data increases, the number of Q- tables and hence the size of memory required for storing the results exponentially leading to increased cost.

Yang et al. [22] proposed a multidimensional approach for feature phishing detection based on deep learning. URL's character sequence features are used for performing classification using deep learning. Then, various statistical features of the URL are combined to obtain multidimensional features which are then classified to achieve very high accuracy. *Vashishth et al.* [23] proposed the use of a cascaded learning-based ML approach for routing in *OppIoT*. A logistic regression classifier was used as an input cascade to a neural network classifier. The major disadvantage of this approach was the use of multiple classifiers that led to an increase in the system complexity and computation time. The malicious behavior of nodes was not addressed.

Yan et al. [24] proposed a malware detection method using two deep neural networks *CNN* and *LSTM* for learning from grayscale images and opcode sequences extracted from raw binary executable files. Then the result obtained from these networks were combined using stacking ensemble along with extra metadata features. Finally the results were obtained for classification of malware. Stochastic Gradient Decent was used later for training the logistic regression model. Use of so many techniques in a single protocol makes the system very complex computationally as well as on cost level.

Adeel et al. presented a survey of several techniques for disaster management using IoT networks [25]. The work presents details of several wireless technologies that can prove to be really helpful in communication and enhancing the efficacy of prediction, management and monitoring of disasters. Future scope and opportunities for improvement are very well emphasized in this work. *Irfan et al.* [26] proposed a system for gathering the movement data of users using sensors built into a home. The authors proposed a data reduction technique and demonstrated the graphical view of several daily living activities. The system automatically predicts and sends a message to healthcare professional in case of an emergency.

Srinivas et al. [17] suggested a defense mechanism against *HELLO* flooding attack based on deep learning. Their approach included cluster head selection and optimal shortest path selection. Route discovery time and threshold function was used for detection of *HELLO* flood attackers. The shortest path was selected based on trust, transmission delay, distance between the nodes and packet loss ratio. The major limitation of their work was that the trust-aware schemes can make their security mechanism invalid. The algorithm is works really slowly and other attacks have not been addressed.

All the work in literature focused on either simple *RL* techniques or deep learning alone. The procedures followed were complex and hence non- realistic for application to a range of *OppIoT* devices. In addition to this, only a few types of attacks were addressed and attackers were not identified proactively. The efficiency of the security approaches was not taken into consideration. These limitations have motivated the authors to propose a Deep Q- Learning based security approach that brings the advantages of *RL* and Deep learning techniques together. The proposed protocol addresses several attacks and is quite efficient.

3. DQNSec: Deep Q-learning based secure routing protocol

Ensuring security of the *OppIoT* networks is a challenging task due to the lack of fixed path between nodes and a wide spectrum of devices. Traditional security approaches lack the ability to predict the future conduct of the nodes. *ML* techniques learn from the past behavior of the nodes and predict their future actions, thereby isolating the malicious nodes before they could cause any harm to the network. *DQNSec* is a deep neural network based security approach that uses Q leaning to learn the behavior of the nodes and later predict the future conduct of the nodes for protecting against hello flooding, sinkhole and distributed denial of service attacks.

3.1. Preliminaries

In *RL*, an agent interacts with its environment and creates several learning experiences. Agent transitions from one state to another. *RL* can be modeled as a *MDP* [14]. Q-learning is a type of *RL* where the main aim is to achieve maximum total reward which is computed using bellman equation [9] as shown in Eq. (1) where γ is the discount factor that lies in the range 0 to 1 and helps in managing the importance of future rewards. Q-learning makes use of Q-table for saving the rewards. The size of Q table increases with increasing number of state action pairs making it inefficient.

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a - Q_t(s_t, a_t)) \tag{1}$$

Here deep learning comes to the rescue [27]. Deep learning is a powerful tool has the ability for function approximation and can easily learn to reduce dimensions of input data. The combination of deep neural networks (*DNN*) and Q learning approach are deep Q networks (*DQN*) that were proposed by *Google DeepMind* [28].

DQN architecture is depicted in Fig. 1. Every action creates an experience and are stored in the experience replay memory, comprising of state *s*, reward *R*, action *a* and the next possible states. Experience replay helps in achieving a learning process that is uniform and stable. Target network is replica of the estimation network whose parameters are updated after certain period of time. *Q value* can be computed using the bellman equation as shown in Eq. (1) where α is the learning rate. The loss function described by Eq. (2) where β and β' are parameters of estimation and target *DNN*.

$$L(\beta) = E[(R + \gamma max_{a'} Q(s', a'|\beta') - Q(s, a|\beta)^2] \tag{2}$$

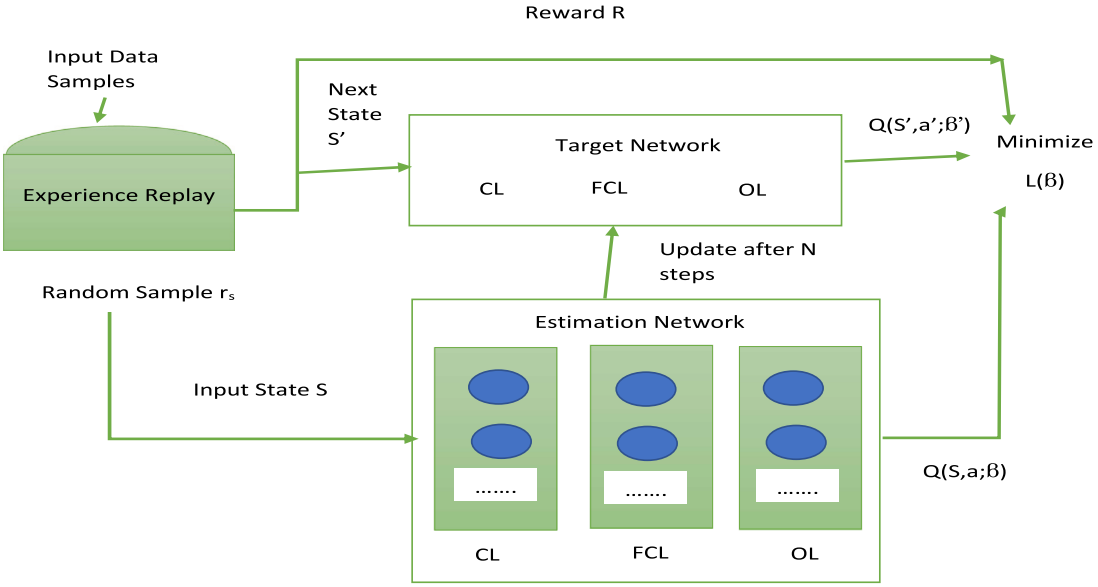

Fig. 1. DQN architecture.

3.2. Network and attack model

OppIoT is assumed to be comprising of *n* nodes that cooperate in the transmission of messages and possess enough battery and buffer capacity. Nodes exchange several parameter values like, (the number of messages created, message copies, messages received, destination ids, malicious nodes encountered, packets forwarded, and packets dropped), upon encounter. In the proposed approach, malicious node can attack the network in any of the following ways:

- **Hello flooding attack**: where the attacker generates a huge number of 'Hello' messages and floods the network for wasting the valuable resources of the network.
- **Sinkhole attack**: The attacker advertises itself as having the shortest path to the destination node for attracting maximum traffic from its neighbors. The compromised node, later consumes the data for launching other attacks or simply drops the packets.
- **Distributed denial-of-service (DDoS)**: DDoS is the most complicated attack and very difficult to identify as the attacker generates packets very similar to the original data and the attack is executed in a distributed manner. The main aim is to prevent the legitimate users of a service from using the desired resources by burdening the service provider.

DQNSec predicts the encountered node's behavior based on their past actions so that the delivery probability of the protocol stays high and the network is protected before any major loss can be done to the network.

3.3. Proposed DQNSec protocol

DQNSec comprise of two phases, training and prediction phase. During the training phase, first step is the *Pre- processing* of data where raw data is transformed into meaningful dataset and noise is removed. *Feature extraction* is then performed where several important features are extracted from the dataset and labeling of data is also performed. The data still have different mean and variance, so *Feature normalization* is performed so that all the data lie in the same range, 0 to 1. Next important step is, *Feature selection* where an optimal subset of features is obtained and irrelevant features are removed from the dataset. The dataset is then divided into two parts, training dataset (*DTrain*) and testing dataset (*DTest*), usually in the ratio 2/3 and 1/3 respectively. Then the training dataset is fed into the *DQN* which learns using multiple agent approach. The advantage of multi- agent over experience replay is that the overhead and cost of extra memory requirement is reduced and the learning is diversified. Ada delta optimizer [29] is used for performing learning. The training procedure followed is described in algorithm 1.

The use of multiple actors approach for learning where the learners are just the CPU threads that operate on a specific machine that helps in reducing the cost of communication between learners. This approach helps in diversifying the learning process as each learner explores different environment. In addition to this, separate policies can be adopted for each actor–learner pair. This justifies the fact that there is no requirement for experience replay memory, as the stabilization is performed by actors running in parallel, thus reducing the cost of implementation and the time required for training is also reduced.

DQNSec makes use of the *A3C* approach [12] which is a type of multi- actor approach comprising of a single *DNN* for approximating the value and policy functions. *DNN* has two convolutional layers [30] and a fully connected layer and produce two outputs, softmax layer which approximates the policy function $\pi(a_t|s_t;\theta)$ and the output of the linear layer is the value function $V(s_t;\theta)$. Asynchronous gradient descent technique [31] is utilized by multiple agents for optimization. Gradients are computed by

Algorithm 1 DQN Training

1: **Begin**
2: Input the dataset csv file.
3: Remove noise and convert csv into meaningful dataset.
4: Feed dataset into decision tree for feature selection and importance.
5: Split dataset as *DTrain* and *DTest*.
6: Model deep learning sequential model.
7: Set the layers of the neural network.
8: Set Ada Delta as the optimizer.
9: Set the Loss function.
10: Input DTrain.
11: Begin training.
12: Save the output model and weights obtained.
13: Input DTest.
14: Produce the final classification report.
15: **End**

the agents and updates are sent after a regular interval to the server. The central server then sends the updated weights to the agents for maintaining a common policy among the agents. The policy function is given in Eq. (3) where θ_t is the value of θ at a given time t. R_t is the reward function for interval t to $t+k$, as defined in Eq. (4). $H(\pi(s_t;\theta))$ is entropy for favoring exploration over exploitation during training and β helps in controlling the strength of entropy.

$$f_\pi(\theta) = \log \pi(a_t|s_t;\theta)(R_t - V(s_t;\theta_t)) + \beta H(\pi(s_t;\theta)) \tag{3}$$

$$R_t = \sum_{i=0}^{k-1} \gamma^i r_t + 1 + \gamma^k V(s_{t+k};\theta_t) \tag{4}$$

The value function is given in Eq. (5).

$$f_v(\theta) = (R_t - V(s_t;\theta))^2 \tag{5}$$

During training the gradient is calculated as shown in Eq. (6).

$$g = \alpha g + (1-\alpha)\Delta\theta^2 \tag{6}$$

$$\theta \leftarrow \theta - \eta\Delta\theta\sqrt{g+\varepsilon} \tag{7}$$

After the training is complete, the testing phase begins where the DTest, which is the remaining 1/3 part of the dataset, is used for testing purpose. The final classification report produced is then tested for accuracy.

Features considered for *DQNSec* are as follows.

- **Number of packets transmitted** (n_1) is the count of packets that are transmitted or forwarded by a node.
- **Number of packets dropped** (n_2) is the count of packets dropped at each node in the network.
- **Number of packets received** (n_3) is the measure of packets being received at a node.
- **End to end delay** (n_4) is total delay from packet creation to the delay in packet forwarding till it gets delivered to the destination node.
- **Buffer occupancy** (n_5) depicts the residual buffer at a node.
- **Energy level** (n_6) is the current residual energy or battery power of a node.
- **Reception time** (n_7) is total time spent by node in receiving data packets.
- **Transmission time** (n_8) is the total time spent by a node in message transmission.
- **Misrouted packets** (n_9) is the total number of packets routed on a wrong route by maliciously behaving nodes.
- **Packet delivery probability** (n_{10}) is the ratio of packets correctly delivered to the total number of packets transmitted.

The above mentioned features help in the identification of the correct class of the encountered node viz. benign or malicious. The attackers perform either of the attacks, sinkhole, hello flood or distributed denial of service attack.

Sinkhole attacker makes an attempt at attracting maximum traffic towards itself by advertising as having the shortest path to the destination. The attacker compromises a node for sending fake routing information to its neighboring nodes. The sinkhole attack can be identified using the average time spent by a node in packet reception (PRT). PRT is usually very high for sinkhole attacker. In addition to this, n_1 is low and n_5 is usually high.

$$Packet\_Reception\_Time = \frac{n_7}{n_3} \tag{8}$$

While performing the hello flood attack, the attacker sends random *HELLO* packets and floods the network making the services unavailable to the legitimate nodes of the network. Keeping a track of the transmission average time (TT) can prove to be very helpful in identifying hello flooding attack. n_1 feature is very high for flooding attackers.

$$Transmission\_Average\_Time = \frac{n_8}{n_1} \tag{9}$$

DoS attackers prevent legitimate users of a service from using the desired resources. Distributed denial-of-service (*DDoS*) is a form of *DoS* attack that is distributed in nature, affects a larger volume of traffic and targets huge number of nodes. *DDoS* can be identified by tracking the energy level of a node. Performing such attack exhausts the attackers resources. n_8 is usually high and huge delay is observed in packet delivery.

In order to promote desirable behavior of nodes, reward function is designed. Whenever an actor takes some action and transitions its state, a reward is assigned to him. In the proposed approach, an actor is rewarded positively whenever it conducts benign behavior and its action enhances throughput and reduces packets dropped and delay and saves energy. Reward function is defined in Eq. (10), where ω_i is the weight for controlling the weights given to each feature.

$$R = \omega_1 * n_{10} + \omega_2 * n_6 - \omega_3 * n_9 - \omega_4 * n_5 - \omega_5 * n_4 - \omega_6 * n_2 \tag{10}$$

The final output is generated making use of the Rectified Linear (*ReLu*) function [32]. *ReLu* is a non-linear function that makes the back propagation of errors and activates the neurons in multiple layers. The training is performed several times until the error is minimal. The computed reward value is then input into the *DNN* for predicting the Q- value and ultimately finding an optimal path to the destination. The nodes with higher Q- value are preferred for packet forwarding. The routing procedure is described in algorithm 2.

Algorithm 2 DQNSec

```
Begin
Initialize learning rate α and discount factor γ.
Initialize Malicious_Host list.
Obtain the training dataset.
Obtain the feature set, F = (n1, n2....., n10).
Call DQN Training Algorithm 1.
Sender generates message.
for (each neighboring node N do
  if (Malicious_Host contains (N)) then
    Malicious host encountered, wait for a better carrier.
  else
    Predict whether N is malicious or benign.
    if (N is malicious)) then
      Malicious node encountered.
      Add node as Malicious_Node to the Malicious_Host list.
      Wait for a better forwarder till packet.TTL expires.
    else
      Compute the Q- value (Q_val) for N.
      if (N. Q_val > Carrier.Q_val) then
        Forward the packet to the benign node encountered.
        Compute the reward R for the transition made.
      else
        Wait for a better forwarder till packet.TTL expires.
      end if
    end if
  end if
  if (packet.Destination == Receiving_Node) then
    Destination reached.
  else
    Continue routing.
  end if
end for
End
```

4. Simulation results

DQNSec protocol is simulated using the ONE simulator [33] making use of the real data traces, *Kaggle's Microsoft Malicious dataset* [34] which is a malware dataset released by Microsoft in 2015. *Tensorflow Playground* [35] is used to train and deploy the deep neural network for the implementation of the *DQNSec* scheme.

4.1. Simulation setup

OppIoT is assumed to be composed of n nodes each possessing a buffer size of 100 MB. The area for simulation is set as 1000 m $*$ 1000 m. Random Movement Model is used for depicting the node movement. The scanning granularity of bluetooth is set at 120 per second and the *TTL* of the packets is set as 100 min. Every simulation runs for 337 418 s and a new message is generated every 25–35 s of size 500 kB – 1 MB.

DQNSec protocol's performance is compared against several machine learning protocols viz., *RFCSec* [36], *MLProph* [37], *CAML* [23] and *RLProph* [38]. Performance metrics against which the comparison is made are as follows:

- **Packets dropped:** It is the average number of packets dropped by a node during routing is a representative of its behavior. This metric should be very low for receiving higher rewards.
- **Message delivery probability:** It indicates the number of packets successfully delivered to its destination. This metric should be high for an efficient routing protocol.
- **Average delay:** It is the latency observed in delivering the packets to its destination node. The value of this metric should be low for good performance.
- **Average overhead ratio:** It is the total overhead observed in packet delivery by the network. This metric should be low as a higher value indicates that the resources are not managed well.

The performance of secure protocol in presence of malicious nodes is further measured against metrics viz., *Accuracy* and *F-score*.

True positive T_P is the outcome when a protocol makes correct prediction when a node is malicious. True negative T_N on the other hand is the outcome obtained when the protocol makes correct prediction and the node is benign. False positive F_P is when the protocol predicts incorrectly and classifies a benign node as malicious. False negative F_N is when the protocol classifies a malicious node incorrectly as benign. *Accuracy* metric is computed as shown in Eq. (11).

$$Accuracy = \frac{T\_P + T\_N}{T\_P + T\_N + F\_P + F\_N} \tag{11}$$

F score is measured using recall and precision as shown in Eq. (14).

$$Precision = \frac{T\_P}{T\_P + F\_P} \tag{12}$$

$$Recall = \frac{T\_P}{T\_P + F\_N} \tag{13}$$

$$F\_Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{14}$$

4.2. Simulation results

The learning rate α is used for controlling the size of steps at each iteration while minimizing the loss gradient. If the value of α is low, the time spent for training is very high and a higher rate results in higher cost and the predictions are not accurate. During simulations, firstly efforts are made for deciding an optimal value for α. The time to live for a message is set as 100 min and α is varied for observing its impact on the delivery probability as shown in Fig. 2. Zero value for α implies that the agent takes only the previously obtained information into consideration and a value of 1 signifies that only the new learning is considered. The value of α is chosen to be 0.6 as it produces highest delivery probability and after that saturation and even a slight fall is observed after this value.

γ is the discount factor that helps in controlling the importance of future rewards. As the value of γ rises, the delivery probability rises. The impact of varying γ on delivery probability is depicted in Fig. 3.

The impact of changing the percentage of attackers present in the network is noted on various quality metrics as shown in Fig. 4. is varied from 5% to 25%. As the number of attackers rise in the network, the model improves its learning based on past encounters and the value of metrics tend to rise. The average value observed for *F score* is 91% and that for *Accuracy* is 0.896%.

Further, the impact of varying percentage of attackers is noted on delivery probability as shown Fig. 5. As the count of attackers rise, the delivery probability tends to fall. Amongst all studied protocols, *DQNSec* produces highest delivery probability as it works proactively for attacker isolation. The average value observed for delivery probability for *DQNSec* is 0.479 which is about 35.5% superior to *RFCSec*, 33.3% better than *RLProph*, 31.2% better than *CAML* and 29.4% superior to *MLProph*. The effect of this change is then noted on the number of dropped packets as shown in Fig. 6. Rising count of attackers lead to a rise in number of data packets getting dropped in the network. Average packets dropped for *DQNSec* is about 15.2% lower as compared to *RFCSec*, 41.9% lower than *RLProph*, 45.2% lower than *CAML* and 46.9% lower than the count of packets dropped by *MLProph*. Fig. 7 depicts the impact of varying attacker percentage on packet delivery. Average latency tends to rise with increasing count of malicious nodes in network. The average latency for *DQNSec* is 1874.5 s which is about 10.5% lower than *RFCSec*, 23.2% lower than *RLProph*, 36.5% better in comparison to *CAML* and 35% lower than *MLProph*. Fig. 8 displays the overhead ratio observed in the network over rising attacker percentage. The overhead rises with rising attacker percentage. The average overhead ratio observed for *DQNSec* is 73.4 which is the lowest. Table 1 provides a summary of the results shown in Figs. 5 to 8. DQNSec outperforms all the studied protocols.

Fig. 2. Delivery probability vs. learning rate.

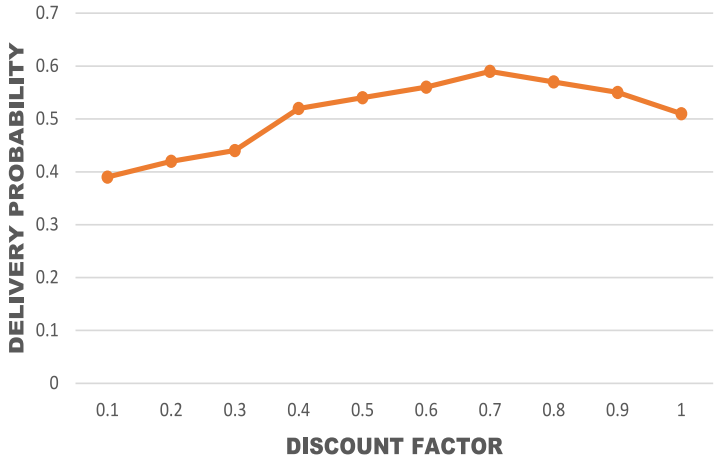

Fig. 3. Delivery probability vs. discount factor.

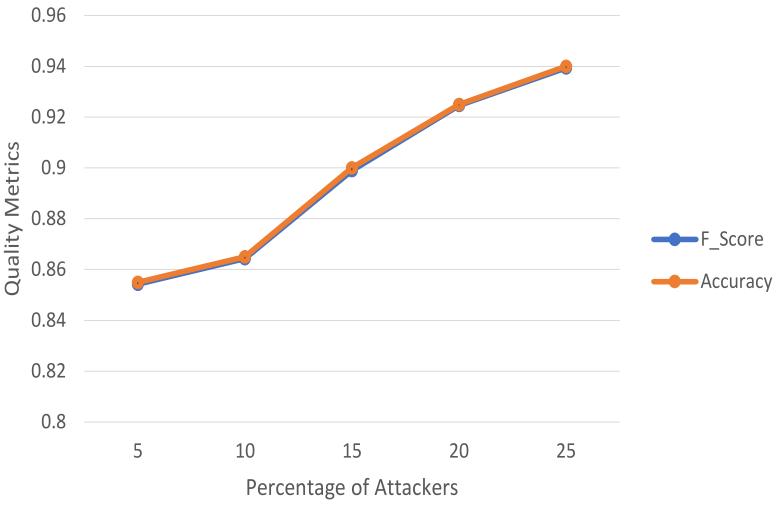

Fig. 4. Percentage of malicious nodes vs. metrics of quality.

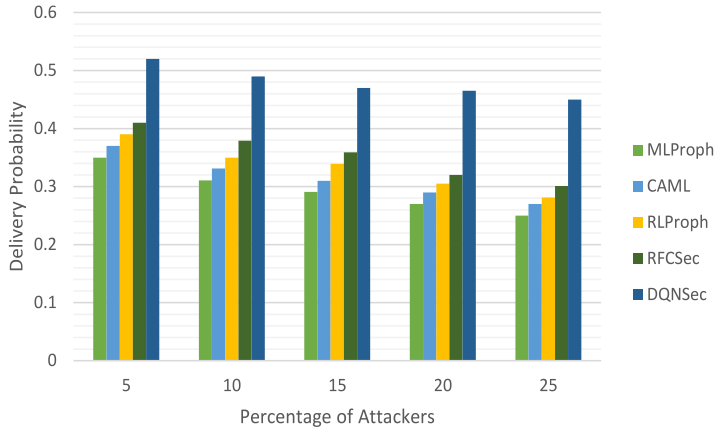

Fig. 5. Delivery probability vs percentage of attackers.

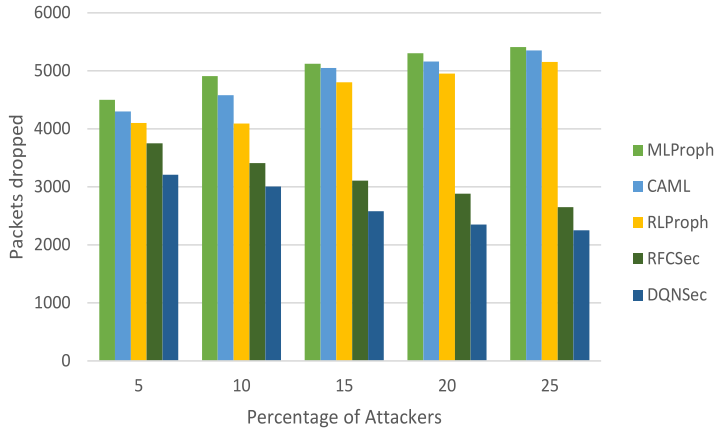

Fig. 6. Packets dropped vs percentage of attackers.

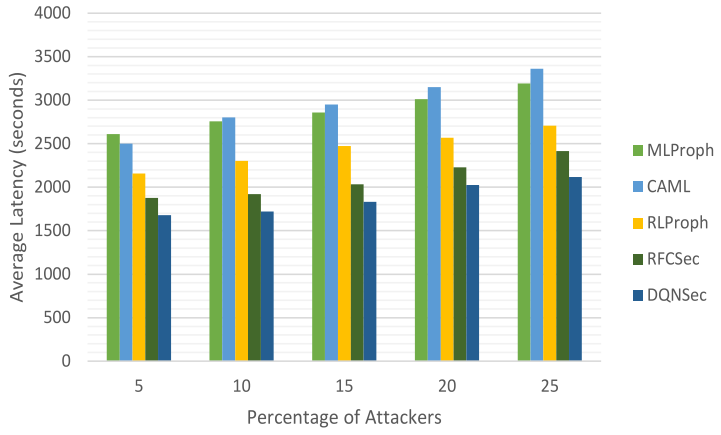

Fig. 7. Average latency vs percentage of attackers.

For the next set of observations, the percentage of attackers is kept fixed at 5% and the size of buffer is varied. Fig. 9 shows the impact of changing buffer on delivery probability. As the size of the buffer rises, the probability of packet delivery rises. This is attributed to the fact that with enhanced buffer size, larger number of packets are able to reside in the buffer leading to an increase in the packet delivery. The average packet delivery probability for *DQNSec* is 0.56 which is about 24% higher in comparison to *RFCSec*, 32% superior to *RLProph*, 38.3% higher than *CAML* and 46.7% higher than *MLProph*. Fig. 10 shows the impact on the

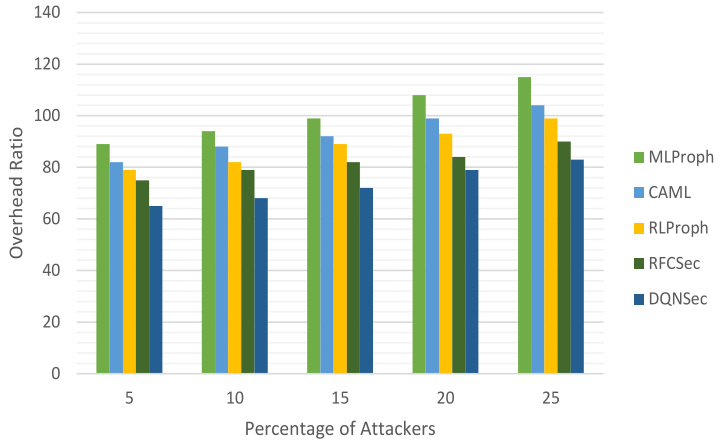

Fig. 8. Overhead ratio vs percentage of attackers.

Table 1
Metrics across varying percentage of attackers.

Protocol	Delivery probability	Packets dropped	Average latency	Overhead ratio
MLProph	0.294	5048	2885.21	101
CAML	0.32	4888	2952.41	93
RLProph	0.334	4618	2441.11	88.4
RFCSec	0.354	3160	2094.97	82
DQNSec	0.479	2679	1874.58	73.4

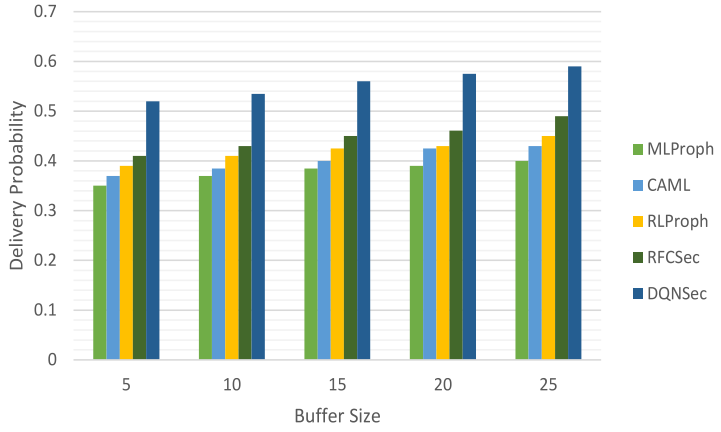

Fig. 9. Delivery probability vs buffer Size.

Table 2
Metrics across changing buffer size.

Protocol	Delivery probability	Packets dropped	Average latency	Overhead ratio
MLProph	0.379	4170	2867.258	75
CAML	0.402	4051	2725.60	71
RLProph	0.421	3821	2340.9	67.2
RFCSec	0.4482	3382	2060.17	63
DQNSec	0.556	2933	1863.24	55.4

packets getting dropped in the network. A fall is observed in the packets dropped with rising buffer size. For *DQNSec* the count of packets dropped is the lowest. The impact on delay in packet delivery is captured in Fig. 11. The average latency for *DQNSec* is about 9.5% lower to *RFCSec*, 20.4% lower to *RLProph*, 31.6% lower than *CAML* and 35.2% lower than *MLProph*. Fig. 12 depicts the impact on the network overhead ratio. Average overhead tend to fall for all studied protocols with rising buffer size. Average overhead for *DQNSec* 55.4 which is the lowest. Table 2 summarizes the results under varying size of buffer.

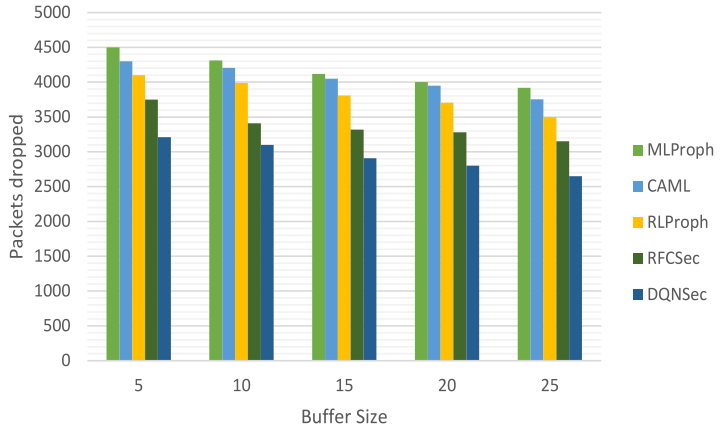

Fig. 10. Messages dropped vs buffer Size.

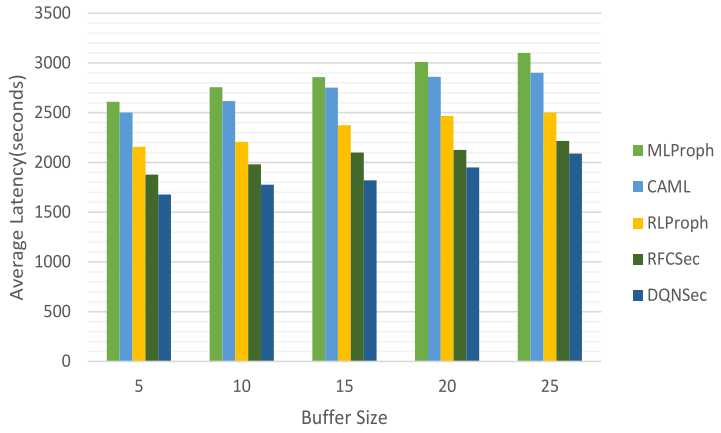

Fig. 11. Average latency vs buffer Size.

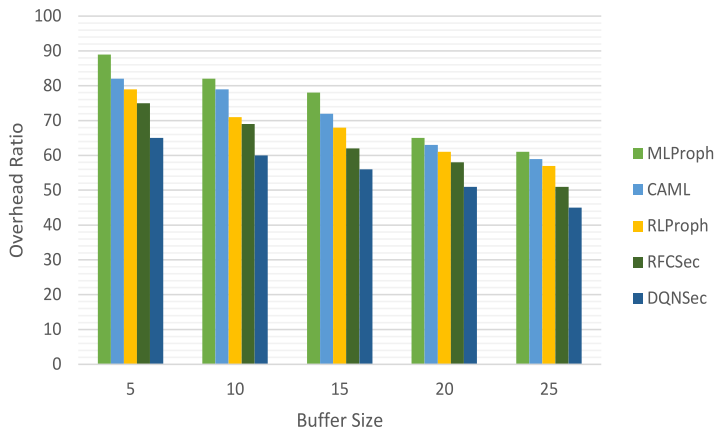

Fig. 12. Overhead ratio vs buffer Size.

Final set of observations were made across varying time to live for messages. Fig. 13 shows that an increasing *TTL* leads to a fall in packet delivery probability as the packets reside in the buffer for a longer time, enhancing its chances of getting dropped from the buffer. Average delivery probability for *DQNSec* is 0.486 which is about 28.5% higher in comparison to *RFCSec*, 35% greater than *RLProph*, 44.2% higher than *CAML* and 50.9% superior to *MLProph*. The overall count of packets getting dropped in the network rises with the rising *TTL* as shown in Fig. 14. Average number of packets dropped in the network for *DQNSec* is found to be the

Table 3
Metrics across changing *TTL*.

Protocol	Delivery probability	Packets dropped	Average latency	Overhead ratio
MLProph	0.322	4696	2814.41	106.6
CAML	0.337	4549	2672.41	99.4
RLProph	0.36	4311	2345.11	93.8
RFCSec	0.378	3927	2059.57	88.2
DQNSec	0.486	3441	1849.98	71.4

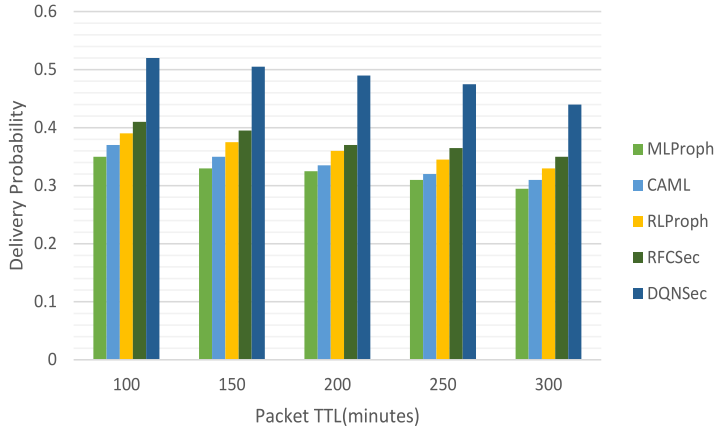

Fig. 13. Delivery probability vs *TTL*.

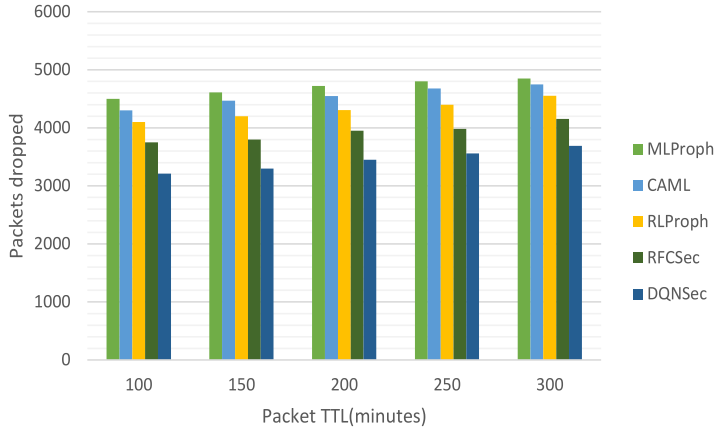

Fig. 14. Number of messages dropped vs *TTL*.

lowest. Fig. 15 shows the impact on delay in packet delivery with changing *TTL*. Average latency observed for *DQNSec* is 1849.5 s which is the lowest as compared to other protocols taken into consideration. Finally the impact is noted on the overhead incurred in the network with rising *TTL* as shown in Fig. 16. *DQNSec* performs best with lowest overhead ratio of 71.4 which is about 19.2% lower than *RFCSec*, 23.38% lower than *RLProph*, 28.16% better than *CAML* and 33.33% lower than *MLProph*. Table 3 provides a summary of the results obtained with changing *TTL* of the packets.

The results obtained via simulations prove that the proposed *DQNSec* secure routing protocol performs best under all circumstances, the accuracy is high and still provides security against several attacks.

5. Conclusion

This paper presented an actor–critic Deep Q-learning based secure routing approach for *OppIoT*, *DQNSec*, that protects the network from hello flooding, sinkhole and distributed denial of service attack. The use of actor–critic approach provides advantages of both the value-based and policy-based methods. The learning process is modeled as a Markov decision process. *DQNSec* successfully detects and isolates the sinkhole, DDoS and hello flooding attackers from participation in the packet forwarding

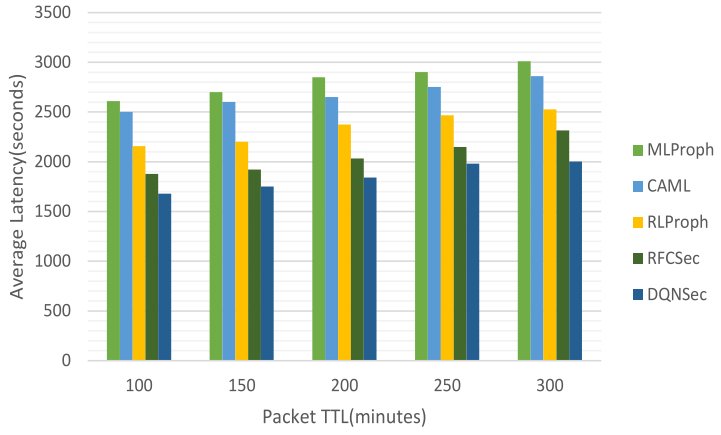

Fig. 15. Average latency vs *TTL*.

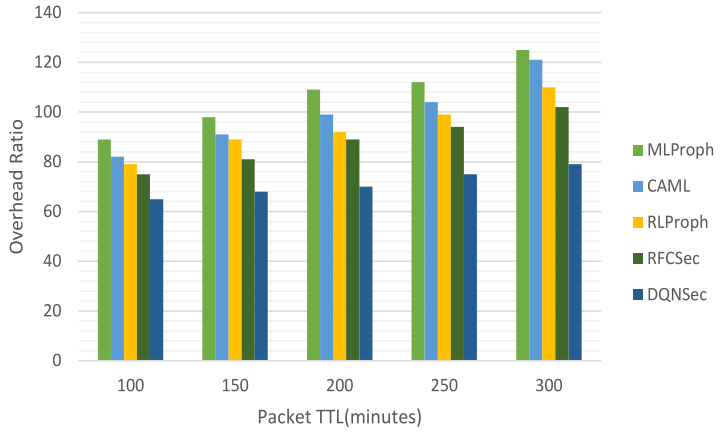

Fig. 16. Overhead ratio vs *TTL*.

procedure. Simulations performed using the real data trace show that the *accuracy* of prediction and *F- score* are very high. Other results obtained in terms of delivery probability, average latency, overhead ratio and number of packets dropped are superior in comparison to other studied protocol. In the future, we plan to incorporate other deep learning techniques for ensuring network security and compare their effectiveness with current work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] F. Bai, N. Sadagopan, A. Helmy, IMPORTANT: A framework to systematically analyze the impact of mobility on performance of RouTing protocols for adhoc networks, in: IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428), Vol. 2, IEEE, 2003, pp. 825–835.

[2] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2003, pp. 27–34.

[3] L. Pelusi, A. Passarella, M. Conti, Opportunistic networking: data forwarding in disconnected mobile ad hoc networks, IEEE Commun. Mag. 44 (11) (2006) 134–141.

[4] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput. Netw. 54 (15) (2010) 2787–2805.

[5] B. Guo, D. Zhang, Z. Wang, Z. Yu, X. Zhou, Opportunistic IoT: Exploring the harmonious interaction between human and the internet of things, J. Netw. Comput. Appl. 36 (6) (2013) 1531–1539.

[6] A.B. Paul, S. Biswas, S. Nandi, S. Chakraborty, MATEM: A unified framework based on trust and MCDM for assuring security, reliability and QoS in DTN routing, J. Netw. Comput. Appl. 104 (2018) 1–20.

[7] J. Canedo, A. Skjellum, Using machine learning to secure IoT systems, in: 14th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2016, pp. 219–222.

[8] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[9] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[10] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al., Deep q-learning from demonstrations, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.
[11] J. Chen, J. Zhou, Z. Cao, A.V. Vasilakos, X. Dong, K.-K.R. Choo, Lightweight privacy-preserving training and evaluation for discretized neural networks, IEEE Internet Things J. 7 (4) (2019) 2663–2678.
[12] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1928–1937.
[13] X. Liu, L. Xie, Y. Wang, J. Zou, J. Xiong, Z. Ying, A.V. Vasilakos, Privacy and security issues in deep learning: A survey, IEEE Access (2020).
[14] E. Levin, R. Pieraccini, W. Eckert, Using Markov decision process for learning dialogue strategies, in: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181), Vol. 1, IEEE, 1998, pp. 201–204.
[15] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, Application of deep reinforcement learning to intrusion detection for supervised problems, Expert Syst. Appl. 141 (2020) 112963.
[16] A. Patcha, J.-M. Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends, Comput. Netw. 51 (12) (2007) 3448–3470.
[17] T.A.S. Srinivas, S. Manivannan, Prevention of hello flood attack in IoT using combination of deep learning with improved rider optimization algorithm, Comput. Commun. (2020).
[18] J. Deogirikar, A. Vidhate, Security attacks in IoT: A survey, in: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), IEEE, 2017, pp. 32–37.
[19] K. Sonar, H. Upadhyay, A survey: DDOS attack on internet of things, Int. J. Eng. Res. Dev. 10 (11) (2014) 58–63.
[20] F.Y. Yavuz, U. Devrim, G. Ensar, Deep learning for detection of routing attacks in the internet of things, Int. J. Comput. Intell. Syst. 12 (1) (2018) 39–58.
[21] G. Jinarajadasa, L. Rupasinghe, I. Murray, A reinforcement learning approach to enhance the trust level of manets, in: 2018 National Information Technology Conference (NITC), IEEE, 2018, pp. 1–7.
[22] P. Yang, G. Zhao, P. Zeng, Phishing website detection based on multidimensional features driven by deep learning, IEEE Access 7 (2019) 15196–15209.
[23] V. Vashishth, A. Chhabra, D.K. Sharma, A machine learning approach using classifier cascades for optimal routing in opportunistic internet of things networks, in: 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), IEEE, 2019, pp. 1–9.
[24] J. Yan, Y. Qi, Q. Rao, Detecting malware with an ensemble method based on deep neural network, Secur. Commun. Netw. 2018 (2018).
[25] A. Adeel, M. Gogate, S. Farooq, C. Ieracitano, K. Dashtipour, H. Larijani, A. Hussain, A survey on the role of wireless sensor networks and IoT in disaster management, in: Geological Disaster Monitoring Based on Sensor Networks, Springer, 2019, pp. 57–66.
[26] M. Irfan, H. Jawad, B.B. Felix, S.F. Abbasi, A. Nawaz, S. Akbarzadeh, M. Awais, L. Chen, T. Westerlund, W. Chen, Non-wearable iot-based smart ambient behavior observation system, IEEE Sens. J. 21 (18) (2021) 20857–20869.
[27] Y. He, Z. Zhang, F.R. Yu, N. Zhao, H. Yin, V.C. Leung, Y. Zhang, Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks, IEEE Trans. Veh. Technol. 66 (11) (2017) 10433–10445.
[28] J. Powles, H. Hodson, Google DeepMind and healthcare in an age of algorithms, Health Technol. 7 (4) (2017) 351–367.
[29] Adadelta, 2020, [Online]. Available: https://keras.io/api/optimizers/adadelta/, accessed December 2020.
[30] S. Lawrence, C.L. Giles, A.C. Tsoi, A.D. Back, Face recognition: A convolutional neural-network approach, IEEE Trans. Neural Netw. 8 (1) (1997) 98–113.
[31] I. Grondman, L. Busoniu, G.A. Lopes, R. Babuska, A survey of actor-critic reinforcement learning: Standard and natural policy gradients, IEEE Trans. Syst. Cybern. C 42 (6) (2012) 1291–1307.
[32] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: ICML, 2010.
[33] A. Keränen, J. Ott, T. Kärkkäinen, The ONE simulator for DTN protocol evaluation, in: Proc. of the 2nd International Conference on Simulation Tools and Techniques, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2009, pp. 1–10, http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5674.
[34] Kaggle dataset, 2020, [Online]. Available: https://www.kaggle.com/c/microsoft-malware-prediction/data, accessed December 2020.
[35] Tensorflow playground, 2020, [Online]. Available: https://github.com/tensorflow/playground, accessed December 2020.
[36] N. Kandhoul, S.K. Dhurandher, I. Woungang, Random forest classifier-based safe and reliable routing for opportunistic IoT networks, Int. J. Commun. Syst. 34 (1) (2021) e4646.
[37] D.K. Sharma, S.K. Dhurandher, I. Woungang, R.K. Srivastava, A. Mohananey, J.J.P.C. Rodrigues, A machine learning-based protocol for efficient routing in opportunistic networks, IEEE Syst. J. 12 (3) (2016) 2207–2213.
[38] D.K. Sharma, J.J. Rodrigues, V. Vashishth, A. Khanna, A. Chhabra, Rlproph: a dynamic programming based reinforcement learning approach for optimal routing in opportunistic IoT networks, Wirel. Netw. (2020) 4319–4338.