International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-07, Issue-04, JULY 2021

A Novel Approach to Network Intrusion Detection using Deep Reinforcement Learning

[1]Gowthami R, Student, Bangalore Institute of Technology, Bengaluru, India, rgowthami97@gmail.com

[2]Dr. R Nagaraja, Professor PG and Research Coordinator, Bangalore Institute of Technology, Bengaluru, India, profrnagaraja@gmail.com

Abstract: Intrusion detection system has become the next step for many organizations as current tools often fail to adapt to ever-growing threats. The number of threats rising exponentially presents increasing challenges due to extensive growth and usage of networks. Therefore, there is a need for adaptive and robust technique to detect complicated and powerful attacks. To tackle these problems, a novel approach to network intrusion detection using Deep Reinforcement Learning (DRL) is proposed. Thus, DRL opens up a new application in computer network domain. To guarantee a better performance for multi-class classification, Deep Q-learning, a variant of Q-learning, has been applied extensively to solve a wide range of complex decision-making tasks. This work also presents a hybrid learning system of applying DRL to labelled data. Unlike real-life scenario, where the algorithm is said to interact with a live environment, this approach is trained and evaluated in a simulated environment. This IDS not only detects local intruders but can also detects outside intruders. Thus, the system with its ability of self-updating to unknown attacks is capable of identifying new patterns also. The proposed model has achieved an overall accuracy of 82.9% successfully.

*Keywords — **Network Intrusion Detection, Deep Reinforcement Learning, Deep Q-networks, Neural network***

I. INTRODUCTION

With the increase in the number of data and connected devices in a system, a variety of security challenges are encountered. Various security tools such as firewalls, anti-malwares help prevent unauthorized traffic entering the network which affects the integrity of the network. Although these solutions help detect and report any unauthorized intrusions successfully, they fail to identify the security breaches within the organization. As a result, a valuable security tool called Intrusion Detection System (IDS) was developed to identify any suspicious activities and detect attacks that originate from within the network.

IDS can be classified into two categories i.e., (1) Network-based and (2) Host-based IDS. A network-based IDS (NIDS) usually monitors all the traffic in the network. While the host-based IDS (HIDS) can only monitor hosts/devices in a system for network activities.

Network-based Intrusion Detection System (NIDS) is a security tool which examines the patterns for the known and unknown attacks in the entire network. Network IDS has an extensive range of applications when it comes to protecting the network and securing the data. NIDS takes upper hand over traditional methods in monitoring the network expansion and availability in case increase in demand. Two other approaches are used for attacks: Signature-based intrusion detection system for identifying patterns of with known attacks. Anomaly-based network intrusion detection system to alert suspicious behavior associated with unknown attacks.

Many Machine Learning (ML) and Deep Learning (DL) techniques are adopted for designing NIDS. But the models based on these types of learning are vulnerable to complicated and powerful attackers. Even though many research has been made in exploring different techniques for intrusion detection, there are certain issues related to the existing system. The issues are (1) large amount of data and incorrect information affects the performance of the system, and (2) the ability to store information for a long time. This paper develops a reinforcement learning (RL) system for intrusion detection as the solution. Reinforcement Learning (RL) is a very general framework for learning problem by interacting with the environment. It is said to make sequential decision according to the feedback from the environment. The agent in RL chooses an action based on its current state and receives reward/penalty and new state from the environment. Through trial-and-error, the agent learns the optimal policy (choosing the best action) to maximize the reward over time. Recently, Deep Reinforcement Learning (DRL) algorithm, a combination of

 DOI : 10.35291/2454-9150.2021.0395

Deep learning and Reinforcement Learning has shown excellent result in the field of robotics, computer vision, manufacturing, healthcare, but less explored in networking domain. The proposed system introduces Deep Q-learning based approach to develop an efficient Network Intrusion Detection System. The objectives of the proposed system are (1) Study the existing techniques in IDS and identify their strengths and weaknesses, (2) Explore the field of RL and extend its application to networking domain, and (3) Develop a hybrid learning system by infusing Reinforcement learning to supervised learning problems.

The system is designed to have the ability to automatically learn from changing attack patterns in the environment. The network traffic data, called NSL-KDD dataset, is extracted from the network sources. The proposed method will be trained on labelled data collected from a private and isolated environment. The deep Q-learning approach is implemented in python programming language. Once the learning is complete, the performance of the proposed system is evaluated by testing it on NSL-KDD test set which includes additional attack types that did not appear in the train set. This approach addresses the issues discussed earlier by (1) selecting a subset of data from bigger samples with the help of a technique called 'minibatch', and (2) the use of Replay memory capable of storing experiences, thus helping speed up the learning process.

Paper organization. The rest of this paper is organized as follows: The literature survey is described in Section 2. Reinforcement learning is introduced in Section 3. The proposed system with technical details is described in Section 4. Section 5 demonstrates the implementation of the proposed system and presents the experimental setup and simulation results. Finally, the paper is concluded in Section 6.

II. RELATED WORK

There are many related works using existing machine learning and deep learning models for intrusion detection with different datasets. In this section, we discuss the works applied to the NSL-KDD dataset, deep reinforcement learning in various domains.

Various machine learning classifiers [1], [2] were employed for classification of data as malicious or not. Four feature subsets were analyzed by removing irrelevant features. The model found to be effective with higher prediction rate and lower computational complexity. Abnormal activities were recognized by a WEKA tool [3] on two different datasets. Another scheme [4] using ANN was used to evaluate the performance of NSL-KDD dataset. Higher detection rate is 81.2% and 79.9% was achieved. [5] discusses challenges with datasets used for IDS.

Deep learning models are also explored in developing a flexible IDS. A scalable, hybrid framework called Scale-Hybrid-IDS-AlertNet (SHIA) has been proposed [6] to monitor the network traffic and host-level attacks. [7] presents a comparative study of deep learning approaches and datasets used for intrusion detection. Two types of categories (binary and multi) have been analyzed by seven deep learning models.

Q-learning has become the driving force in the innovations of intelligent systems [8]. Deep reinforcement learning [9] has been applied to various fields such as robotics [10], [11], manufacturing [12], [13], [14] and one of its types, Deep Q-learning has been massively applied on gaming applications [15], [16], [17], [18], [19] and also used to exploit the trained labelled data in real-world stock markets [20].

III. BACKGROUND

A. *Reinforcement Learning*

Reinforcement learning is a technique of Machine Learning where the agent is trained to make a sequence of decisions. The agent learns to take actions by interacting with the environment in order to maximize the total rewards as shown in Figure 1. The machine is trained through a reward-based feedback mechanism to continuously improve the action. RL is modelled with a set of constraints such as states, actions, policy and rewards. At each step, the agent identifies the state, chooses random action, and receives a reward from the environment. The agent's policy maps states to actions. As a result, Reinforcement learning provides solution by correlating immediate actions with the delayed outcomes. Reinforcement learning finds a diverse set of applications in the field of Robotics, Natural Language Processing, Computer vision, Manufacturing and Healthcare.

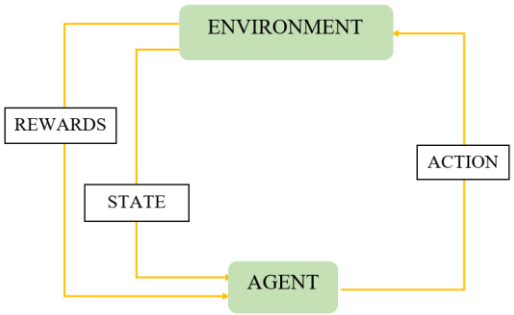

Figure 1. Block diagram of Reinforcement Learning

B. *Deep Reinforcement Learning*

DRL, a sub-field of RL, is a combination of Deep learning and Reinforcement Learning. Reinforcement learning when combined with deep learning yields phenomenal results. The "deep" in reinforcement learning refers to the layers of artificial neural networks.

The DQN (Deep Q-Network) algorithm, developed by Google DeepMind in 2015 [18], is the most successful DRL algorithm. Most of the Atari games have been implemented by Deep Reinforcement Learning algorithms. The algorithm

 DOI : 10.35291/2454-9150.2021.0395

comprises Q-Learning in combination with deep neural networks and a buffer called Experience replay as shown in Figure 2.

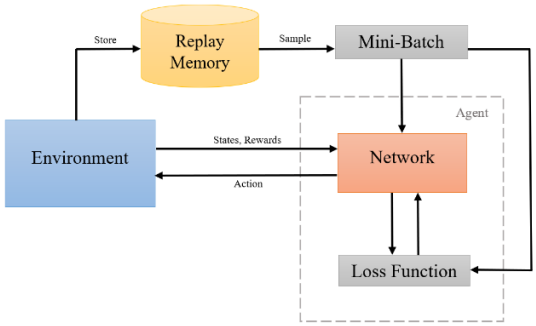

Figure 2. DQN Architecture

Deep Q-learning algorithm is the modified version of Q-Learning algorithm. It replaces the regular Q-table with the neural networks [21]. A DQN uses neural network as a functional approximation with Q-Learning. A technique called Experience Replay is used as a memory for storing the episode steps in case of off-policy learning from which samples are drawn at random.

The Deep Q-Network is used to implement 1) deep neural network classifier which approximates the value function, 2) replay memory for the training phase, and 3) a q-network to estimate the predicted values and a target network to estimate the target values.

C. Neural Network

Neural network consists of several nodes that are interconnected in multiple layers as shown in Figure 3. These layers comprise the following:

1. Input Layer: The input layer refers to the number of states (input data) in an environment [22].
2. Hidden Layers: The neural network architecture is comprised of one or more hidden layers. It is the layer of mathematical functions.
3. Output Layer: The output layer refers to the number of actions (output data).

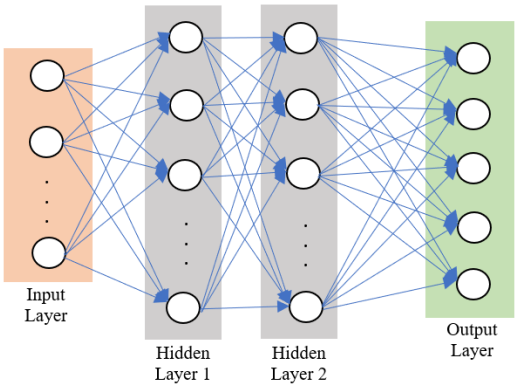

Figure 3. Architecture of a typical neural network for intrusion detection system

In Deep Reinforcement Learning, neural networks are function approximators that maps state to action-value pairs. The agent implements a classifier (Deep Q-Learning algorithm). The fully connected neural network is used for Q-function approximation. There are two phases: 1) the training phase for training the neural network, and 2) the test phase for testing the DRL algorithm [23]. During the training phase, the DRL will train the neural network, learns an optimal policy and stores the NN parameters in the replay memory. Once the training stage is completed, the RL agent will use the learned optimal policy for the best actions. The complete steps for training DQN are presented in Algorithm 1.

IV. METHODOLOGY

In this section, the proposed system using deep reinforcement learning is presented. Before presenting the processing steps of the system, the detailed description of various components of DQN architecture related to the system is introduced.

Agent: An agent is the one responsible for taking actions (or decisions) in the environment. The algorithm used for training is the agent.

Replay memory: It is a technique to store the agent's experiences at each time step. It has the ability to store past and new experiences. First, the observed transitions are stored. During the neural network training, samples are randomly extracted in order to update the network. It helps stabilize the learning algorithm by breaking the temporal correlations among parameters [24].

Q-Network (MainNet): It is a multi-layer neural network in DQN algorithm, used to approximate the action value function.

Target Network: It is copy of Q-Network, used to calculate a target value and is updated by the Q function at regular intervals. Training the main Q-network helps stabilize the learning process.

Environment: It is the world with which the agent interacts frequently. It can either be live or simulation. The current state is returned by the environment. Since the data is already available, the model is integrated with a simulation environment for automatic interactions between the agent and the environment.

The basic concepts in reinforcement learning that describes the interaction between an agent and the environment: state, action, and reward.

State (s): It describes the current situation of the network. The state (or observation) in the environment corresponds to each instance in the training data. Here, state 's' is taken as the network traffic samples (features).

Action (a): Action is the set of all possible decisions that an agent takes. Only discrete, possible actions are chosen by the agent. In the proposed work, the action 'a' corresponds to network intrusion label predictions.

 DOI : 10.35291/2454-9150.2021.0395

Reward (r): It is the feedback used to evaluate the agent's action. The reward is the immediate information given to the agent in the given state by the environment during learning. Here, the reward 'r' corresponds to the measures the success or failure of the prediction. 1/0 reward signifies how well is the agent's action. For every correct label prediction, a reward of 1 is obtained and for incorrect prediction, a reward of 0 is obtained.

Other important concepts:

Discount factor (γ): It describes the importance of future rewards to those in the immediate future. The value of discount factor is in the range 0 and 1.

Policy: The epsilon-greedy approach is followed throughout the training phase. It is used to select the best action with the highest estimated reward over time. This method balances exploration and exploitation by choosing between them randomly.

Loss: Loss function evaluates how well the algorithm models the dataset. It estimates the error between the predicted and actual value. Higher the loss if the predictions are not up to mark. It outputs low value if the predictions are good.

Algorithm 1: Deep Q-Learning with Experience Replay

Initialize replay memory D
Initialize networks Q and $\hat{Q}$ with random weights
for episode 1, M **do**
  Initialize state s
  for t = 1, T **do**
    Choose an action a using policy ε-greedy
    Take action a, observe reward r and next state s′
    Store transition (s, a, r, s′) in D
    Set s′ = s
    Sample a random minibatch of transitions from D
    Set $y_i = \begin{cases} r_i & \text{for } s' \\ r_i + \gamma \max_{a'} \hat{Q}(s_i', a') & \text{otherwise} \end{cases}$
    Perform a gradient descent step on parameters
    Update target network parameters
  end for
end for

The processing steps of the proposed system is as follows:

A. Data Collection

The NSL-KDD dataset used in this study is collected from an online resource called Kaggle. The data used for implementation shows how effective a DQN algorithm is when detecting the anomalies in the network. The data consists of 41 features and 1 label. The dataset comprises both numeric and categorical values. There are 39 attack types which are categorized into 5 class labels i.e., 'normal', 'DoS', 'Probe', 'R2L' or 'U2R'.

B. Data Pre-processing

Data pre-processing is an important step in data mining that defines the success rate of the prediction. It involves data cleaning, reduction and transformation. Since the dataset contains no duplicate records, only transformation is applicable. Technique such as Normalization is used for data transformation. This means the data is transformed into an appropriate format suitable for data mining. Since the dataset contains different datatypes, Normalization and One-hot encoding are carried out on the samples for speeding up the learning. In min-max normalization, the minimum values get transformed to 0, maximum values to 1 and the rest of the values between 0 and 1. Min-max normalization is performed on numeric values and One-hot encoding on categorical values.

C. Training phase

The input layer corresponds to the number of states in the environment**.** The input to the neural network is the one-hot encoded state-input vector got after data pre-processing step. There are two fully connected hidden layers with 80 units(nodes) per layer and Rectifier Linear Unit (ReLU) is used as an activation function in the hidden layer. Finally, the output layer comprises of 5 nodes (i.e., representing 5 class labels), corresponding to the number of actions in the environment**.**

The training process is performed in two parts. During the pre-training phase, the agent selects action according to the ε-greedy policy. Probability ε selects the action randomly while probability 1− ε selects the action according to the maximum Q-value [25]. In the process, agent's experience (state, action, rewards, next state) is stored into the replay memory at each time step.

During the training phase, we randomly sample a batch of experiences from Replay memory. Each batch is called as "minibatch". Mean Squared Error loss is computed between Q-network and Target network. Q-value is said to be the total cumulative reward for an action at a given state. The loss function is the square of difference between the predicted and the target Q-value. Loss function outputs lower value if the predictions are not up to mark. Higher the loss function if the predictions are good. Only the Q-network is trained in the process and the Target network is used to train the Q-network in order to obtain the maximum Q-value. The target value is calculated using the prediction for the new state based on the Bellman equation. The target Q-value is given by the sum of immediate reward and the discounted max Q-value for the next state over time.

$$\text{Target Q-value} = r + \gamma \max_{a'} Q(s', a')$$

$$\text{Loss } [\{(r + \gamma \max_{a'} Q(s', a')) - Q(s, a)\}^2]$$

Target Prediction

Stochastic gradient descent is an optimization algorithm used in this study to minimum the loss and update the network weights. By periodically fixing the weights and updating the network parameters, maximum value of Q-

 DOI : 10.35291/2454-9150.2021.0395

values can be obtained. This determines the corresponding action to be outputted.

D. Testing phase:

The Q-table scores from the training phase is used for testing. The trained model is used for testing on test dataset. The model's performance is the evaluated. The flowchart is depicted in Figure 4.

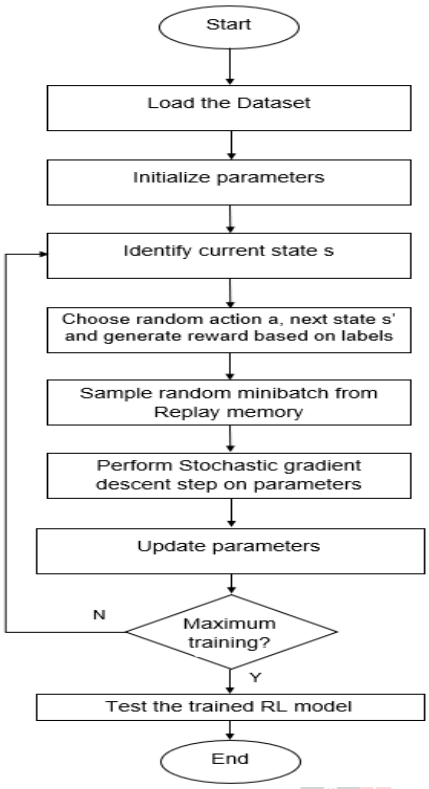

Figure 4. Flowchart illustrating the training and test phase of the proposed system

V. EXPERIMENTS AND RESULTS

In this section, the proposed DRL method is evaluated based on the methodology discussed in the previous section. The experiments are conducted to evaluate whether the model is successful in intrusion detection and how well it learns the patterns with respect to existing as well as new types of attacks. This section describes the experimental setup covering the dataset description, study requirements and the result analysis.

A. Dataset Description:

The selection of an appropriate data plays an important role in data analysis. Different datasets have different impact on the performance of the system and some result in generating poor prediction outcomes. In order to address this problem, NSL-KDD, a new dataset has been used in this work. NSL-KDD is the most commonly used data for network intrusion detection. Network Security Laboratory Knowledge Discovery and Data Mining is abbreviated as NSL-KDD. It is the modified version of KDD99 dataset.

Table 1. The class distribution of NSL-KDD dataset for training and testing

Dataset	Normal Class		Anomaly Class	
	Label	Instances	Label	Instances
Train	Normal	67,343	DoS	45,927
			Probe	11,656
			R2L	995
			U2R	52
Test	Normal	9711	DoS	7458
			Probe	2421
			R2L	2754
			U2R	200

The advantages of NSL-KDD dataset over KDD99 dataset are as follows:

a. No redundant records in the train set making the classifier to not produce any biased result

b. No duplicate record in the test set resulting in better detection rates

c. The number of train and test set records being rational makes the system run on complete set without having to worry about selecting the subset of it randomly

NSL-KDD dataset comes in two separate files for training and testing. There are 1,25,973 training samples and 22,544 test samples, both containing 41 features and 1 label. Each record is labelled as either normal and specific attack type. Table 1 shows in detail the class distribution of NSL-KDD dataset. In total, there are 38 attack types and 1 normal type in the dataset. The training dataset consists of one normal type and 22 attack types as shown in Table 2. Along with the attack types in training dataset, additional attack types (making it in total 38 attack types) and one normal type are present in test set. Thus, this determines how effective the model is in handling the unknown attacks. The attack types are categorized into 5 class labels: normal, DoS, Probe, U2R and R2L.

Table 2. NSL-KDD attack types and classes

Attack Labels	Attack types in training dataset	Additional attack types in test dataset
DoS	back, land, neptune, pod, smurf, teardrop	worm, apache2, udpstorm, mailbomb, processtable
Probe	ipsweep, nmap, portsweep, satan	saint, mscan,
U2R	buffer_overflow, loadmdoule, perl, rootkit	ps, sqlattack, xterm
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster	http_tunnel, named, sendmail, snmpgetattack, snmpguess, xclock, xsnoop

 DOI : 10.35291/2454-9150.2021.0395

B. Experimental setup:

The experiment study was implemented on an AMD Ryzen 2.10 GHz processor and with 8 GB of RAM. Keras, a higher-level library along with TensorFlow was used to build the algorithm with the neural network. The experiment was carried out on Anaconda Spyder IDE via Python. Python version 3.8 is used. The complete training period is divided into episodes. Episode refers to the number of iterations taken by the algorithm during the training procedure to arrive at a desired solution. Inside this loop, another loop referring to the number of steps is chosen to mark the end of each iteration. Network parameters are said to update their weights after every iteration. Necessary hyperparameters for setup: 200 steps and 400 episodes.

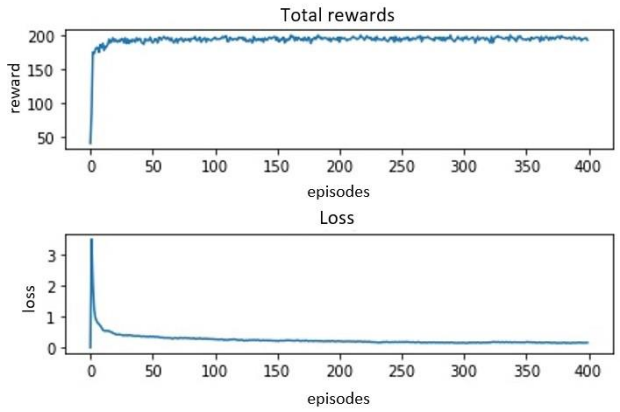

Figure 5. Plot of a) cumulative reward per episode, and b) loss per episode

In the experiment, 125973 instances of records have been used as training data to build the training models for the DQN classifier. The testing phase is implemented based on 22544 instances of records.

With exploration and exploitation strategy, the experiment demonstrates that agent learns better policies by exploiting actions and modifying the Q-values. Results are reported based on the agent's transformation of maximizing rewards during the training period. After the first few episodes, performance improvement of the model can be seen. Figure 5 shows decrease in loss function over time increased agent's ability to handle efficiently all the types of attacks by attaining higher rewards.

After the training, the robustness and adaptiveness of the trained model was tested on a set of 22544 test samples that have an additional types of attacks present. The DQN model has an 82.99% accuracy on the test set. The scores represent the maximum total reward over episodes during testing time. The final output conveys information about DQN's scores of actual labels present in the samples over correctly predicted labels. Figure 6 depicts the test set scores' comparison of total labels and correct label predictions.

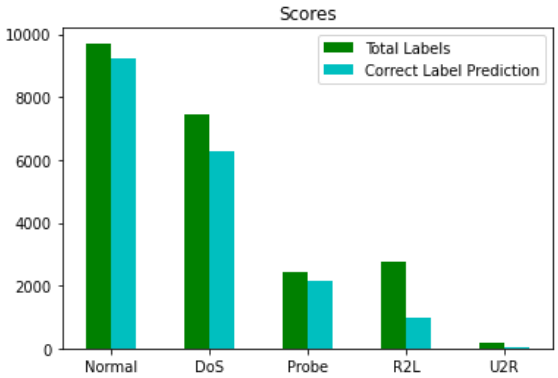

Figure 6. Comparison of scores on test dataset

VI. Conclusion and Future Work

Most of the state-of-the-art IDSs designed using existing approaches to intrusion detection suffer from performance degradation due to the lack of autonomous learning. To handle the complex traffic patterns, a deep reinforcement learning method which implements frequent updates from the system is proposed. With the continuous interactions of agent and environment, the system is capable of autonomously learning new attack patterns and give proper responses. The approach not only detects if there is an attack or not, but also diagnoses the exact class of attacks with its optimal decision-making solution.

In future studies, this approach can be implemented using adversarial examples, which is small perturbations added in input samples to fool the model, and check the effectiveness of the system. NIDS using several deep reinforcement learning algorithms, such as Double DQN, Dueling Q-Network, Prioritized experience replay, Policy-based and Actor-critic algorithms can also be explored.

References

[1] Abrar, Iram et al. "A Machine Learning Approach for Intrusion Detection System on NSL-KDD Dataset." 2020 International Conference on Smart Electronics and Communication (ICOSEC) (2020): 919-924.

[2] Revathi, S. and A. Malathi. "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection." International journal of engineering research and technology 2 (2013)

[3] Meena, Gaurav and R. Choudhary. "A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA." 2017 International Conference on Computer, Communications and Electronics (Comptelix) (2017): 553-558.

[4] Ingre, B. and Anamika Yadav. "Performance analysis of NSL-KDD dataset using ANN." 2015 International Conference on Signal Processing and Communication Engineering Systems (2015): 92-96.

[5] Ahmed, Mohiuddin et al. "A survey of network anomaly detection techniques." J. Netw. Comput. Appl. 60 (2016): 19-31.

 DOI : 10.35291/2454-9150.2021.0395

[6] Vinayakumar, R. et al. "Deep Learning Approach for Intelligent Intrusion Detection System." IEEE Access 7 (2019): 41525-41550.
[7] Ferrag, M. et al. "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study." J. Inf. Secur. Appl. 50 (2020).
[8] Jang, Beakcheol et al. "Q-Learning Algorithms: A Comprehensive Classification and Applications." IEEE Access 7 (2019): 133653-133667.
[9] Mousavi, Sajad et al., "Deep Reinforcement Learning: An Overview. Lecture Notes in Networks and Systems", 2018. 426-440. 10.1007/978-3-319-56991-8_32.
[10] Y. Zhu, R. Mottaghi, E. Kolve, J.J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning. In Robotics and Automation (ICRA), IEEE International Conference on, pp. 3357–3364, 2017.
[11] Zhang, Fangyi et al. "Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control." ArXiv abs/1511.03791 (2015).
[12] Lillicrap T. et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:150902971. 2015.
[13] Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," IEEE transactions on neural networks and learning systems, vol. 29, no. 6, pp. 2042–2062, 2018.
[14] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," IEEE Transactions on Neural networks, vol. 12, no. 2, pp. 264–276, 2001.
[15] Paudel, Prabesh, "Exploring Game Playing AI using Reinforcement Learning Techniques", 2020, 10.13140/RG.2.2.14522.62400.
[16] Hosu, Ionel-Alexandru and Traian Rebedea. "Playing Atari Games with Deep Reinforcement Learning and Human Checkpoint Replay." ArXiv abs/1607.05077 (2016).
[17] Chen, Ao et al. "The Use of Reinforcement Learning in Gaming The Breakout Game Case Study.pdf." (2020).
[18] Giannakopoulos, Petros and Y. Cotronis. "A Deep Q-Learning Agent for L-Game with Variable Batch Training." ArXiv abs/1802.06225 (2017).
[19] Mnih V, Kavukcuoglu K, Silver D, et al. "Human level control through deep reinforcement learning". Nature 2015; 518(7540): 529–533
[20] Carta, Salvatore & Ferreira, et al., "Multi-DQN: An Ensemble of Deep Q-Learning Agents for Stock Market Forecasting. Expert Systems with Applications", 2021. 164. 10.1016/j.eswa.2020.113820.
[21] Deep Q-Learning Tutorial: minDQN - A Practical Guide to Deep Q-Networks https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc
[22] Reinforcement Learning with Neural Network https://www.baeldung.com/cs/reinforcement-learning-neural-network
[23] Huang, Qiuhua et al. "Adaptive Power System Emergency Control Using Deep Reinforcement Learning." IEEE Transactions on Smart Grid 11 (2020): 1171-1182.
[24] Bruin, T. D. et al. "The importance of experience replay database composition in deep reinforcement learning." (2015).
[25] Sutton, R. and A. Barto. "Reinforcement Learning: An Introduction." IEEE Transactions on Neural Networks 16 (2005): 285-286.

 DOI : 10.35291/2454-9150.2021.0395