



# A Study on the Efficacy of Deep Reinforcement Learning for Intrusion Detection

 Halim Görkem Gülmez<sup>1</sup>,  Pelin Angin<sup>2</sup>

<sup>1</sup>Middle East Technical University; halim.gorkem.gulmez@gmail.com

<sup>2</sup>Corresponding Author; Middle East Technical University; pangin@ceng.metu.edu.tr

Received 30 November 2020; Revised 14 December 2020; Accepted: 26 December 2020; Published online 03 February 2021

## Abstract

The world has witnessed a fast-paced digital transformation in the past decade, giving rise to all-connected environments. While the increasingly widespread availability of networks has benefited many aspects of our lives, providing the necessary infrastructure for smart autonomous systems, it has also created a large cyber attack surface. This has made real-time network intrusion detection a significant component of any computerized system. With the advances in computer hardware architectures with fast, high-volume data processing capabilities and the developments in the field of artificial intelligence, deep learning has emerged as a significant aid for achieving accurate intrusion detection, especially for zero-day attacks. In this paper, we propose a deep reinforcement learning-based approach for network intrusion detection and demonstrate its efficacy using two publicly available intrusion detection datasets, namely NSL-KDD and UNSW-NB15. The experiment results suggest that deep reinforcement learning has significant potential to provide effective intrusion detection in the increasingly complex networks of the future.

**Keywords:** security, deep reinforcement learning, intrusion detection

## 1. Introduction

The fast-paced developments in computing and network infrastructures in the past two decades have led to the rise of the Internet of Things (IoT) paradigm with ubiquitous connectivity along with increasingly widespread usage of cloud computing. While these developments have greatly facilitated daily operations in many industries and enterprises in addition to touching the daily lives of people in positive ways, the resulting cyber security issues have created deterrents for the more widespread adoption of IoT due to an enlarged attack surface with many security vulnerabilities. The number of zero-day attacks, which are security incidents whose signatures were not previously observed, is rising every day with the increasing number of vulnerabilities in these networked systems. Some of these attacks can have devastating consequences, as they are now capable of destroying not only software, but also hardware components through IoT connections.

Modern network intrusion detection and prevention systems (IDPS) have the purpose of detecting and mitigating various attacks on networked systems with sub-second response times. While IDPS in legacy systems mostly relied on attack signature-based solutions, which would create rules for each observed attack pattern and compare incoming traffic with the rules in the IDPS's database, this solution is not sufficient to cover the variety of attacks in today's complex systems both because of the high volume of traffic that needs to be analyzed in real time and due to the inability to generalize and detect attacks with unknown signatures. Security researchers thus have turned to machine learning (ML) and deep learning (DL) techniques that are capable of learning patterns of attacks and normal behavior of systems so that anomalous network traffic can be detected and classified in real time, and the IDPS can adjust itself to deal with new types of attacks over time.

Reinforcement learning (RL) algorithms, which are based on agents interacting with a runtime environment under a variety of states to learn to maximize their rewards, has been a popular technique for many learning-based tasks since their introduction. More recently, deep reinforcement learning (DRL) algorithms, which utilize deep neural networks within RL to facilitate representation of many possible state-action pairs and provide generalizability, have been applied successfully to a variety of

problem domains. Among successful applications of DRL are Atari games [1], chess [2], solving arithmetic problems [3], medication treatment plans [4], optimization of chemical reactions [5], and extraction of biological sequence data [6] among many others. Despite its success in various fields, the application of DRL to network security has been rather limited so far.

In this paper we propose a DRL-based approach for network intrusion detection and evaluate its effectiveness on two real-world benchmark datasets that have been commonly used in the evaluation of ML-based approaches for detecting cyber attacks in legacy networks, namely NSL-KDD and UNSW-NB15. The evaluation results demonstrate that DRL is a promising method for network intrusion detection, achieving F-1 scores of over 96% on both datasets. We also show that the effectiveness of the algorithm is significantly affected by the structure of the embedded deep neural network, i.e., the number of hidden neurons, as well as the number of training iterations. Performance comparison of the model with various state-of-the-art ML/DL models demonstrates its promise, especially in terms of F-1 score, on the two benchmarks.

The remainder of this paper is organized as follows: Section 2 provides an overview of related work in ML-based intrusion detection systems. Section 3 provides details of the proposed DRL model for network intrusion detection. Section 4 provides an experimental evaluation of the model on two public network intrusion detection datasets. Section 5 concludes the paper with future work directions.

## 2. Related Work

The advances in the field of machine learning have paved the way for their use in the field of cyber security for the past two decades. Most existing anomaly-based intrusion detection systems rely on ML techniques. Beehive, a successful solution for detecting intrusion from network logs, was proposed in [7]. Beehive uses four types of features and utilizes k-means clustering to detect anomalies. One downside is that it does not work in real time. Another successful approach utilizing k-means clustering includes the work of [8]. While k-means clustering can be effective for detecting anomalies, predefining the value of k can be a problem in many settings.

Balogun and Jimoh [9] proposed a method utilizing the k-nearest neighbor (KNN) classifier and decision trees. Their approach was shown to be capable of detecting new attacks with high accuracy. [10] utilized a variety of ML algorithms including k-means clustering, isolation forest, histogram based outlier score and cluster-based local outlier factor in their approach called CAMLPAD, and achieved an accuracy of 95% in an intrusion detection task. Pervez and Farid [11] proposed using Support Vector Machines (SVM) for intrusion detection on the NSL-KDD dataset. Although SVM was successful on the training set, it failed to detect many attacks in the test set. In [12], a multi-layer perceptron based model with 3 layers was proposed, which achieved 81% accuracy for binary classification on NSL-KDD. Kamel et al. [13] proposed an AdaBoost-based intrusion detection model and reported 99.9% accuracy on NSL-KDD, however their training and test sets consisted of subsets of the whole dataset, which were not clearly described. Hu et al. [14] also applied AdaBoost for intrusion detection on the KDD Cup'99 dataset and achieved 91% detection rate. Engly et al. [15] evaluated the performance of Gradient Boosting Machines on NSL-KDD and achieved successful results with an ensemble model. Moustafa and Slay [16] applied Expectation-Maximization Clustering, Logistic Regression (LR) and Naive Bayes classification on the UNSW-NB15 datasets, and achieved the best results with an accuracy of 83% for LR.

Following the success of deep learning in many fields in recent years, security researchers have started employing it in many intrusion detection systems. [17] and [18] proposed using recurrent neural networks (RNN) in intrusion detection on data with time dependencies and achieved successful results. A variant of the RNN-based intrusion detection model was proposed by Yin et al. [19], achieving over 83% accuracy on the KDD Cup'99 dataset. An LSTM-based model, which is a special RNN-structure, was proposed by Li et al. [20], which achieved 83% accuracy and F-1 score on NSL-KDD. Behera et al. [21] also proposed the use of convolutional neural networks (CNN) for intrusion detection and achieved high accuracy on the NSL-KDD dataset. They also stated their approach can be adapted to detect zero-day attacks. Another CNN-based intrusion detection model was proposed by Li et al. [22],

which also achieved successful results on NSL-KDD. Lopez-Martin et al. [23] proposed a conditional variational auto-encoder based model for unsupervised intrusion detection, which achieved 80.10% accuracy on NSL-KDD. A two-stage stacked auto-encoders based model was proposed by Khan et al. [24], which achieved 89% accuracy on the UNSW-NB15 dataset. A hybrid model consisting of deep neural networks and spectral clustering was proposed by Ma et al. [25], which achieved around 72% accuracy on NSL-KDD. A scalable, hybrid intrusion detection approach, which utilizes deep neural networks (DNNs), was proposed by Vinayakumar et al. [26]. The distributed DNN-based model was shown to achieve better performance than traditional ML-based classifiers on a set of benchmarks. Gao et al. [27] developed a deep belief networks-based model for intrusion detection, and demonstrated its superior performance in comparison to SVM and MLP.

Researchers have also utilized RL for detecting attacks in networks. Various types of log files were used in the solution of [28] where a rule-based approach was taken to create association rules signaling attacks. Their approach utilized RL as a helper rather than basing the solution on it. [29] also proposed an RL-based approach with multiple agents watching over the network states in a hierarchical manner, which was shown to provide accurate results, although it was not evaluated with different datasets. A cyber security simulation was set up in [30] to apply RL for finding the best strategies of both attackers and defenders in a Markov game. Their experiments demonstrated the tool can be used both for intrusion detection systems and for launching successful cyber attacks on systems. The approach we propose in this work differs from existing RL-based approaches in that it utilizes fully connected deep neural networks for allowing the RL agents to make decisions based on unstructured input data, obviating the need to manually create large state spaces.

### 3. Proposed Intrusion Detection Approach

In this section, we describe our proposed DRL-based approach for network intrusion detection. We first provide a brief overview of deep neural networks, and continue with an explanation of how they are integrated into RL to achieve a highly accurate intrusion detection model.

#### 3.1 Deep Neural Networks (DNN)

Neural networks are a special category of ML models the design of which resembles the functioning of the human brain in the sense that it simulates the processing and transmission of information through the complex networks of neurons, which get excited or inhibited by the signals in the network [31]. One of the first examples of neural network structures is the *perceptron*, which contains a single input layer connected to an output. The perceptron represents the simplest processes in the brain's neurons using an activation function and a set of weights, as depicted in Figure 1(a). Machine learning with a perceptron involves random assignment of weights to each of the input nodes, and the passage of the weighted sum of the input values through an activation function to produce the output value. The weights are adjusted throughout the training process in multiple iterations and the goal of the training process is to minimize the aggregate error in the output. The error is calculated as the difference between the ground truth output, and the output that is calculated by the model.

Multi-layer perceptrons (MLP) are feedforward neural networks containing a number of hidden layers in between the input and output layers, as demonstrated in Figure 1(b). The figure shows a fully-connected deep neural network with one hidden layer, with every input node connected to every hidden node and likewise, every hidden node connected to every output node. When the fully-connected neural network consists of more hidden layers, each node in a hidden layer will be connected to each node in the following hidden layer. As seen in the figure, each edge connecting the nodes has a weight that is updated throughout the training process to achieve minimum output error. The number of hidden neurons in each layer can be different from the number of input and output layer neurons. Training of the network involves running a back-propagation algorithm [31] updating the weights of the edges in each iteration. While the number of input nodes is decided by the dimensionality of the input feature vector, the number of output nodes is decided by the specific learning task, e.g. multi-class classification,

regression, binary classification etc. Among commonly used activation functions in MLP are sigmoids including  $y(v_i) = \tanh(v_i)$  and  $y(v_i) = \frac{1}{1 + e^{-v_i}}$ .

DNNs are yet more complex artificial neural networks with many more hidden layers than MLPs. Their complexity allows them to express more complex hypotheses by better modeling the nonlinear relationships in the network. DNNs provide the inherent ability to learn higher level representations from possibly unstructured data, which makes them very valuable for a variety of machine learning tasks. In this work, we utilize fully connected DNNs integrated into the RL process as described below to achieve highly accurate intrusion detection.

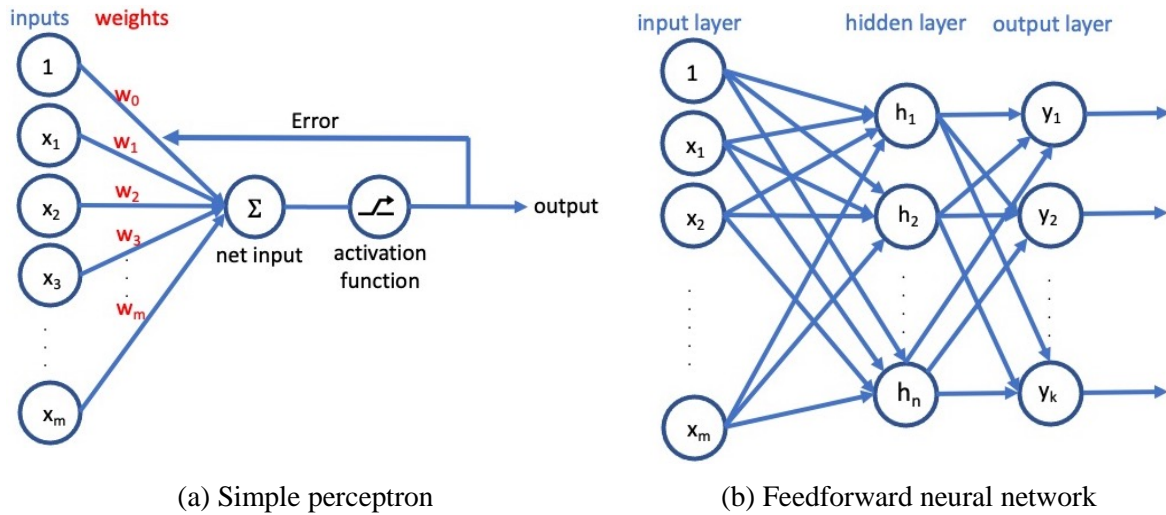


Figure 1 Structure of neural networks

### 3.2 DRL-Based Intrusion Detection

Reinforcement learning is an ML technique based on an agent learning through rewards and punishments it receives through its interactions with the environment. Each state of the agent is associated with a set of actions that could have different rewards, and the agent learns over time what action to perform based on its history of actions-rewards at that same state. An agent in RL takes actions from among a set of possible actions for its current state, and receives a positive or negative reward for taking that particular action, which it saves in its memory. These rewards are then used by the agent to decide which action to take in later states, where the ultimate goal of the agent is to maximize its total reward value. Agents are connected to their environments with action and recognition as described by Kaelbling et al. [32]. Picking a certain action at a certain state results in an output, which modifies the state of the agent, and the agent receives the value of this change with a reinforcement signal. The agent learns to choose the most rewarding action over time by trial and error using different algorithms. The environment is not always deterministic, i.e. choosing the same action can have different consequences at different points in time in the same state.

As apparent from the description above, RL is quite different from supervised learning. While supervised learning utilizes training datasets consisting of labeled input/output pairs, an agent in RL receives immediate rewards based on its actions after performing the action. Learning does not actually stop in RL; it is a continuous process in which the agent keeps receiving new rewards or punishments as it interacts with its environment, however it is expected that the rewards will keep increasing over time, as the agent learns which actions provide the greatest rewards at each state.

The state-action space could get very large in RL in complex environments, causing the algorithm not to generalize well. DRL is an improvement of RL algorithms that provides improved generalization power by augmenting RL with deep neural networks in the state-action input formation. i.e., DRL utilizes deep neural networks for function approximation in policy and value functions in RL. This

capability is important in a network intrusion detection setting, as generalizability matters especially for cases like zero-day attacks.

In an RL algorithm based on Q-learning, the value function is as follows:

$$Q(s,a)=r(s)+\gamma \max_{a'}\Sigma P(s'|s,a)Q(s',a') \quad (1)$$

Equation 1 is the Bellman Equation. Here  $s$  represents the state,  $a$  represents the action,  $r$  represents the reward, and  $P$  represents state change possibility. Based on the equation, the Q value of a state-action pair is equal to the sum of the current reward and potential future Q-values. While this equation is discrete, many real-life applications involve continuous actions and states. Thus, we need an effective function approximation technique for the value function. This requirement is met by integrating DNNs into RL. In the value function using DNNs, every state and Q-value are calculated by utilizing hidden layers of neural networks, which are trained using backpropagation.

Algorithm 1 Deep Q-learning

1	Initialize replay memory $D$ to capacity $N$
2	Initialize Q-function with random weights
3	<b>for</b> episode = 1, $M$ <b>do</b>
4	Initialize neural network from a random state $s$
5	<b>for</b> $t = 1, T$ <b>do</b>
6	Find Q-values for all actions using DNN algorithm: $a_t = \max_a Q^*(s_t, a; \theta)$
7	Choose an action $a_t$ for current state $s_t$ by using e-greedy exploration
8	Move to the next state $s_{t+1}$ with action $a_t$ , pick reward $r_t$
9	Store transition $(s_t, a_t, r_t, s_{t+1})$ in $D$
10	Sample random minibatch of transitions $(s_t, a_t, r_t, s_{t+1})$ in $D$
	Set $y_j =$
11	$\begin{cases} r_j, & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta), & \text{for nonterminal } s_{j+1} \end{cases}$
12	Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$

We describe deep Q-learning [1] in Algorithm 1 above. As the algorithm describes, DNNs are used as part of the RL, forming the DRL algorithm. In RL, immediate rewards are valued more than distant rewards in the future. DNNs provide the capability for the Q-functions to more accurately take future rewards into account when deciding about the actions to take. Another advantage of using DNNs in RL is that the number of interactions needed is reduced by sampling, resulting in better performance and efficiency.

In this paper, we propose a simple DRL model for network intrusion detection, where the learning agent has two different states, i.e. under attack or normal traffic, and four possible actions. Table 1 provides a high-level overview of states, actions and corresponding reward values. The main difference of this approach from state-of-the-art ML/DL models for intrusion detection is the overall learning process, which involves exploration of the different classification options by the learning agent, which is penalized when it incorrectly classifies an instance and rewarded for correct classification. Through this process, the agent learns to take the optimal actions over time to maximize its reward. Here the states refer to the network traces. Unlike traditional deep learning, DNNs are only used as part of the process in DRL to enable representation of the policies of the agent, i.e. actions to be taken to achieved the maximum reward at a specific state, without having to enumerate all possible states manually. The learning process continues throughout the lifetime of the agent. This is an important feature for especially online learning systems, which will be instrumental in successful intrusion detection in the era of ever increasing zero-day attacks. Figure 2 shows an activity diagram of the deep Q-learning algorithm.

Table 1 RL States, Actions and Rewards

State	Action	Reward
Normal	No Alarm	+1
Normal	Alarm	-1
Attack	Alarm	+1
Attack	No Alarm	-1

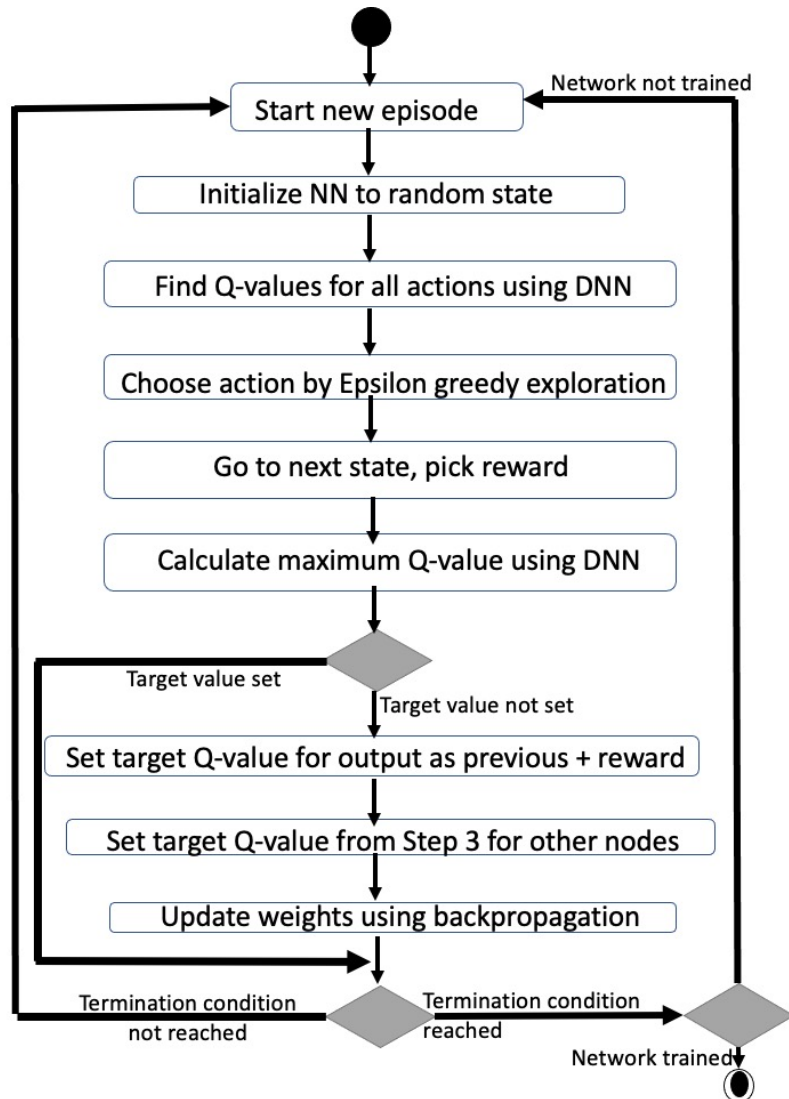


Figure 2 Deep Q-learning activity diagram

#### 4. Experimental Evaluation

We have evaluated the effectiveness of the proposed DRL model for intrusion detection using two benchmark datasets, UNSW-NB15 and NSL-KDD, where the task was to perform binary classification of records into attack and normal classes. Below we describe the datasets and provide results of the performed experiments.

##### 4.1 Datasets

*UNSW-NB15:*

The UNSW-NB15 [33] dataset was created by the University of New South Wales in 2015, using the IXIA tool for generating network traffic including attacks. It has 49 features, two of which are labels for binary classification (i.e. attack or normal traffic) and multi-class classification (i.e. type of attack). The dataset contains 9 types of attack traffic in addition to normal traffic, where attacks include DoS, DDoS, fuzzing, backdoor, analysis, worm, exploit, shellcode and generic. The remaining fields include network packet and connection details like IP addresses, ports, communication protocols. A subset of the features of this dataset are listed in Table 2 below. The dataset consists of about two million network packet traces, which is quite extensive.

Table 2 UNSW-NB15 Features

Feature	Type	Description	Feature	Type	Description
srcip	nominal	Source IP	sloss	integer	Source packets retransmitted or dropped
sport	integer	Source port	dloss	integer	Destination packets retransmitted or dropped
dsip	nominal	Destination IP	service	nominal	http, ftp, ...
dsport	integer	Destination port	Sload	float	Source bits/sec
proto	nominal	Protocol	Dload	float	Destination bits/sec
dur	float	Total duration	Spkts	integer	Source-to-destination packet count
sbytes	integer	Source-to-destination transaction bytes	Dpkts	integer	Destination-to-source packet count
dbytes	integer	Destination-to-source transaction bytes	stime	timestamp	Record start time
sttl	integer	Source-to-destination time to live value	ltime	timestamp	Record last time
dttl	integer	Destination-to-source time to live value	label	binary	0 for normal, 1 for attack
...					

#### NSL-KDD:

KDD CUP'99 [34] has been one of the most frequently used datasets in the evaluation of ML-based intrusion detection techniques since it was released in 1999. This dataset was generated by extracting features from DARPA98 [35], which is a dataset consisting of traffic obtained from the U.S. Air Force LAN. The dataset consists of 41 features and 4 attack categories: probing, denial of service (DoS), R2L, U2R. Despite its age, this dataset is still used by many researchers due to its large size (about 5 million records) and its modeling of a variety of conditions obtained from real network traffic. It also has some drawbacks including the presence of many duplicate records, unbalanced numbers of records from different classes in the training set, which could create biased classification models and the unbalanced distribution of records in the training and test sets [36].

The NSL-KDD dataset [36] was created to solve the abovementioned issues with KDD CUP'99. It involved removal of duplicate records and balancing of the number of records for different classes to prevent bias in the classification. The researchers also provided more balanced training and test sets. All original features from KDD CUP'99 were retained. Different sets were provided in the dataset, including sets with binary classification labels as in UNSW-NB15, sets with attack type labels and difficulty levels, as well as sets not including the hardest-to-detect cases.

## 4.2 Experimental Results

To evaluate the effectiveness of the solution, we have utilized metrics commonly used to in ML to judge the goodness of algorithms, which are precision, recall, accuracy, and F-1 score. The description of each metric is provided in Table 3 below. The abbreviations used in the table are as follows:

TP (true positive): The number of instances correctly classified as attacks

TN (true negative): The number of instances correctly classified as normal traffic

FP (false positive): The number of instances incorrectly classified as attacks

FN (false negative): The number of instances incorrectly classified as normal traffic

Table 3 Evaluation Metrics

Metric	Formula
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Precision	$\frac{TP}{TP + FP}$
Recall	$\frac{TP}{TP + FN}$
F-1 Score	$\frac{2 * Precision * Recall}{Precision + Recall}$

Here, the recall is quite important, as it demonstrates the ability of the algorithm to detect attack traffic. However, equally important is precision, which will ensure that the system will not block legitimate traffic by creating many false positives. Therefore, the value of the F-1 score, which combines both metrics, is a good measure for the efficacy of the algorithm. Accordingly, for the optimization of the models, the F-1 score has been taken as the main performance measure. In the below subsections, we provide performance results of the DRL model on the two benchmark datasets discussed above and compare them with the results of previous work that have utilized the same datasets for evaluation.

### 4.2.1 Experiments with NSL-KDD

The first set of experiments were performed using the NSL-KDD dataset. The learning system was set up in a Gym environment as explained by Koduvally [37]. Gym provides an environment for testing and comparison of RL algorithms. We used the full training and test datasets for the experiments. We experimented with different numbers of training iterations to evaluate the effect of the number of training iterations on the accuracy of the algorithm. Table 4 lists the precision, recall, accuracy and F-1 score values for the experiments with a low number of training iterations (1) and a high number of training iterations (20). As seen in the table, the algorithm achieves very high precision and recall when the number of training iterations is high.

Table 4 Precision, Recall and Accuracy for Varying Number of Training Iterations in NSL-KDD

	Precision	Recall	Accuracy	F-1 Score
Low Iterations	0.715	0.719	0.725	0.72
High Iterations	0.951	0.925	0.940	0.93

We also experimented with different DNN architectures to see the effects of the number of hidden neurons on the performance of the algorithm. As opposed to the number of iterations, we observe that increasing the number of hidden neurons in the DNN does not always lead to better performance. We have tried five different settings and the results are reported in Table 5 below.

In the first experiment, we set the number of neurons at the hidden layers to be 2/3 of the input layer's size. We achieved satisfying results with an accuracy close to %97. In the second experiment, the number of hidden neurons was set equal to the size of the input layer. The performance was much lower than that of the first setting.



In the third experiment, the number of hidden neurons was one and a half times the input layer's size. This made the performance degrade even further. In the fourth experiment the number of hidden neurons was half the size of the input layer, and while the precision and recall values were quite balanced, this setting did not achieve the performance of the first setting either.

In Experiment 5, we used the square root of the input layer's size as the number of hidden neurons. This provided an increase in performance over the previous settings except for the first experiment.

Table 5 Precision, Recall, Accuracy, and F-1 Score for Varying Number of Hidden Neurons in NSL-KDD

	#of hidden neurons	Precision	Recall	Accuracy	F-1 Score
Experiment 1	$\frac{2 * Input\ size}{3}$	0.98	0.96	0.97	0.97
Experiment 2	Input size	0.65	0.91	0.70	0.76
Experiment 3	$\frac{3 * Input\ size}{2}$	0.72	0.54	0.68	0.62
Experiment 4	$\frac{Input\ size}{2}$	0.77	0.79	0.79	0.78
Experiment 5	$\sqrt{Input\ size}$	0.89	0.92	0.89	0.90

After the initial set of experiments with different numbers of hidden neurons, we optimized the training process by automating the setting of hyperparameter values for the DNN component of the model. The optimization process performs a grid search [38] over all given possible values of the different hyperparameters, calculates F-1 scores achieved with the specific hyperparameter settings on the validation dataset and reports the hyperparameter values resulting in the best F-1 score. Grid search is currently one of the most commonly used hyperparameter optimization techniques in DL, as it has been proven to find the most optimal parameter settings when compared to random search and function approximation techniques for hyperparameter optimization. It involves determining a range of possible values for each hyperparameter and training the model with all combinations of those values to find the combination with the optimal performance. In this work, we included the following hyperparameters for DNN in the automated grid search: (a) learning rate (in the range [0, 0.1]) (b) dropout rate (in the range [0, 0.4]) (c) number of hidden neurons (in the range [6, 60]). Adam optimizer and L2 regularization were used for DNN. The best performance was achieved with a learning rate of 0.01, dropout rate of 0.3 and 27 hidden neurons. Before performing grid search for the selected hyperparameters, we performed trials for the other hyperparameters including the number of epochs, batch size and reward decay rate, and the best performance was achieved with 30 epochs, a batch size of 1000 and a reward decay rate of 0.9. Note that although it is possible to include many hyperparameter types and hyperparameter values in the grid search, the more parameter values included, the longer it takes to train the model. For a large hyperparameter space, the optimization process could take days of training, which has been avoided in the DNN literature, as the resulting model could also overfit the training data, decreasing the usefulness of the model for real-world application. The increase in the training time would also hurt the performance of online learning, which is important in intrusion detection systems that need to continuously update their models with new data.

Table 6 provides performances of state-of-the-art ML algorithms in the literature in terms of precision, recall, accuracy, and F-1 score on the NSL-KDD dataset and example related works in the literature utilizing these algorithms. The models compared against include logistic regression, SVM with the Radial Basis Function (RBF) kernel, random forest, Gradient Boosting Machine (GBM), Adaboost, multi-layer perceptron (MLP), convolutional neural networks (CNN) (results of these are provided by Lopez-Martin et al. [39]), variational autoencoder, deep belief network and fully connected deep neural network (results of these are provided by Yang et al. [40]). All of the included models are state-of-the-art ML/DL models that have been utilized in a variety of intrusion detection systems.

As seen in Table 6, the proposed DRL model achieved good results in all of the performance measures for the NSL-KDD experiments. While models such as random forest, GBM, Adaboost, MLP and CNN achieved quite high precision values, their low recall values caused a lower F-1 score. As recall values demonstrate the ability of the models to detect attacks, it is quite an important metric for the goodness of the models in practice.

Table 6 Performance Comparison of Proposed Approach and Existing ML Approaches on NSL-KDD

ML/DL algorithm	Precision	Recall	Accuracy	F-1 Score	Example Work	Method
DRL (proposed)	<b>0.98</b>	<b>0.96</b>	<b>0.97</b>	<b>0.97</b>	--	
Logistic regression	0.90	0.55	0.71	0.68	Moustafa and Slay [16]	A network intrusion detection system using logistic regression in its decision engine along with association rule mining is proposed.
SVM	0.91	0.88	0.88	0.89	Lopez-Martin et al. [39]	Application of an optimized SVM model on intrusion detection datasets is evaluated.
Random forest	0.97	0.57	0.75	0.72	Lopez-Martin et al. [39]	Application of an optimized random forest model on intrusion detection datasets is evaluated.
GBM	0.97	0.63	0.78	0.76	Engly et al. [15]	The performance of GBM on intrusion detection datasets is evaluated by itself vs. in an ensemble with random forests and neural networks.
Adaboost	0.97	0.60	0.76	0.74	Hu et al. [14]	A computationally lightweight intrusion detection model based on direct application of the AdaBoost algorithm is proposed.
MLP	0.97	0.67	0.80	0.79	Ingre and Yadav [12]	An artificial neural network with Backpropagation (BFG) and tansig activation function is proposed for intrusion detection.
CNN	0.97	0.68	0.81	0.80	Li et al. [22]	An image conversion method for network data is proposed and the resulting data is fed into a convolutional neural network for intrusion detection.
Variational Autoencoder	0.95	0.80	0.80	0.87	Yang et al. [40]	A supervised variational auto-encoder with regularization is proposed, which utilizes Wasserstein GAN for learning latent data distribution.
Deep belief network	0.89	0.55	0.57	0.68	Gao et al. [27]	A DNN classifier comprising multilayer unsupervised learning networks, and a supervised backpropagation learning network is proposed for intrusion detection.
Fully connected DNN	0.89	0.61	0.62	0.73	Vinayakumar et al. [26]	A distributed, fully connected DNN architecture is proposed for intrusion detection in large networks.

#### 4.2.2 Experiments with UNSW-NB15

The second set of experiments was performed with the UNSW-NB15 dataset. As in the previous experiments, the optimal hyperparameters were found using grid search with the same set of possible values as in Section 4.2.1. The best performance was achieved with a learning rate of 0.01, dropout rate of 0.3 and 32 hidden neurons. We report the best performance results in Figure 3 below. We performed two different experiments, where we utilized the default training set consisting of 175341 records and test set consisting of 82332 records in the first experiment. In the second experiment we randomly selected training and test data over the dataset. 100000 records were selected for both sets. The results did not change much in this experiment as compared to the first experiment.

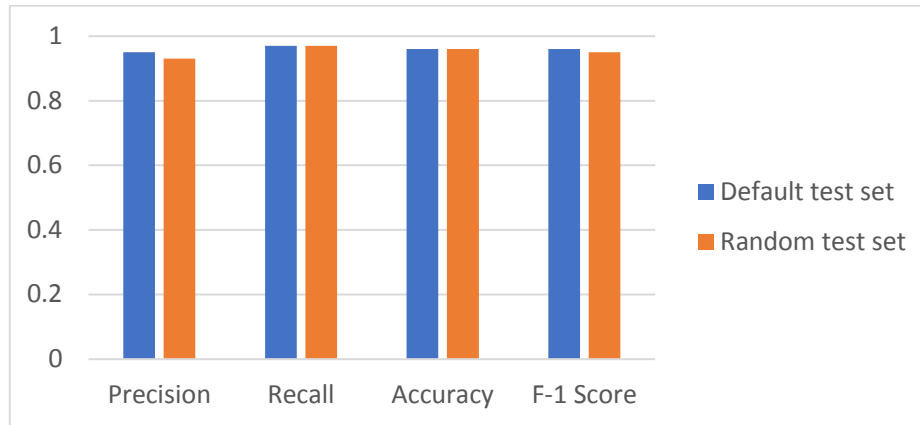


Figure 3 Precision, Recall, Accuracy, and F-1 Score on UNSW-NB15

Table 7 provides a performance comparison of the proposed approach with existing state-of-the-art ML-based approaches in the literature in terms of precision, recall, accuracy, and F-1 score on the UNSW-NB15 dataset. The models in the table are the same as those in Section 4.2.1 and their results are provided by Lopez-Martin et al. [39] and Yang et al. [40] as before.

Table 7 Performance Comparison of Proposed Approach and Existing ML Approaches on UNSW-NB15

Algorithm	Precision	Recall	Accuracy	F-1 Score
DRL (proposed)	<b>0.95</b>	0.97	<b>0.96</b>	<b>0.96</b>
Logistic regression	0.81	0.94	0.84	0.87
SVM	0.75	<b>0.99</b>	0.82	0.86
Random forest	0.83	<b>0.99</b>	0.88	0.90
GBM	0.80	<b>0.99</b>	0.86	0.88
Adaboost	0.80	0.98	0.85	0.88
MLP	0.81	0.98	0.87	0.89
CNN	0.86	0.98	0.90	0.91
Variational Autoencoder	<b>0.95</b>	0.92	0.93	0.94
Deep belief networks	0.85	0.97	0.89	0.91
Fully connected DNN	0.82	0.98	0.87	0.90

As seen in Table 7, high precision, accuracy and F-1 scores are achieved by the proposed DRL-based approach. While for this dataset SVM, random forest and GBM achieve higher recall values, their precision values are much lower than that of the DRL approach, which means they would create many false positives at runtime. The DRL model achieves a better balance between false positives and false negatives, with high precision, recall and F-1 values. This makes it promising for both accurately detecting attacks and achieving high network reliability by avoiding unnecessary interruption of traffic.

The good performance of the DRL model is attributable to the exploration of a wide set of network states and penalizing all incorrect classifications with the same penalty function, which limits the number of false positives and false negatives as the RL agent continues to learn.

## 5. Conclusion

In this work, we proposed a deep reinforcement learning based approach for network intrusion detection. The proposed approach overcomes the generalization shortcomings of reinforcement learning and achieves high performance on binary intrusion detection tasks trying to differentiate between normal and attack traffic. The efficacy of the model was evaluated with two widely used intrusion detection benchmark datasets and F-1 scores of over 96% were achieved for both datasets. We also demonstrated the effects of the number of hidden neurons and number of iterations on the performance of the proposed algorithm. This study has shown that deep reinforcement learning is a promising method for network intrusion detection. We aim to expand upon this study in future work by evaluating the performance of the model on additional datasets as well as creating extensions of the model with different reward functions to achieve optimal performance in a variety of settings.

## References

- [1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," pp. 1–9, 2013. Retrieved from <http://arxiv.org/abs/1312.5602>
- [2] M. Lai, "Giraffe: Using Deep Reinforcement Learning to Play Chess," September, 2015. Retrieved from <http://arxiv.org/abs/1509.01549>
- [3] L. Wang, D. Zhang, L. Gao, J. Song, L. Guo and H. T. Shen, "MathDQN: Solving arithmetic word problems via deep reinforcement learning," *32nd AAAI Conference on Artificial Intelligence*, pp. 5545–5552, 2018.
- [4] S. Nemati, M. M. Ghassemi and G. D. Clifford, "Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2016.
- [5] Z. Zhou, X. Li and R. N. Zare, "Optimizing Chemical Reactions with Deep Reinforcement Learning," *ACS Central Science*, vol. 3, no. 12, pp. 1337–1344, 2017.
- [6] M. Mahmud, M. S. Kaiser, A. Hussain and S. Vassanelli, "Applications of Deep Learning and Reinforcement Learning to Biological Data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2063–2079, 2018.
- [7] T. Yen, A. Oprea and K. Onarlioglu, "Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks," *Proc. 29th Annual Computer Security Applications Conference*, pp. 199–208, 2013.
- [8] A. Razaq, H. Tianfield and P. Barrie, "A big data analytics based approach to anomaly detection," *Proc. - 2016 IEEE/ACM 3rd International Conference on Big Data Computing Applications and Technologies (BDCAT)*, pp. 187–193, 2016.
- [9] A. O. Balogun and R. G. Jimoh, "Anomaly intrusion detection using a hybrid of decision tree

- and K-nearest neighbor," *Journal of Advances in Scientific Research & Applications (JASRA)*, vol. 2, no. 1, pp. 67-74, 2015.
- [10] A. Hariharan, A. Gupta and T. Pal, "CAMLPAD: Cybersecurity Autonomous Machine Learning Platform for Anomaly Detection," *Proc. Future of Information and Communication Conference (FICC), San Francisco, CA, USA*, pp. 705-720, 2020.
- [11] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," *SKIMA 2014 - 8th International Conference on Software, Knowledge, Information Management and Applications*, pp. 1-6, 2014.
- [12] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," *2015 International Conference on Signal Processing and Communication Engineering Systems*, pp. 92-96, 2015.
- [13] S.O.M. Kamel, N. Hegazi, H. Harb, A. ElDein and H. ElKader, "AdaBoost Ensemble Learning Technique for Optimal Feature Subset Selection," *International Journal of Computer Networks and Communications Security* vol. 4, no. 1, pp. 1-11, 2016.
- [14] W. Hu, W. Hu, and S. Maybank, "AdaBoost-Based Algorithm for Network Intrusion Detection," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 38, no. 2, pp. 577-583, 2008.
- [15] A. H. Engly, A. R. Larsen, and W. Meng, "Evaluation of Anomaly-Based Intrusion Detection with Combined Imbalance Correction and Feature Selection," *Proc. 14<sup>th</sup> International Conference on Network and System Security*, Melbourne, Australia, pp. 277-291, 2020.
- [16] N. Moustafa and J. Slay, "A hybrid feature selection for network intrusion detection systems: central points and association rules," arXiv:1707.05505, (2017) [cs.CR].
- [17] J. Kim and H. Kim, "Applying Recurrent Neural Network to Intrusion Detection with Hessian Free Optimization." In: Kim H., Choi D. (eds) *Information Security Applications. WISA 2015. Lecture Notes in Computer Science*, vol. 9503, 2016, Springer, Cham.
- [18] Y. Chuan-long, Z. Yue-fei, F. Jin-long and H. Xin-zheng, "A Deep Learning Approach for Intrusion Detection using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21954 - 2196, 2017.
- [19] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954-21961, 2017.
- [20] Z. Li, A. L. G. Rios, G. Xu, and L. Trajkovic, "Machine learning techniques for classifying network anomalies and intrusions," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 1-5, 2019.
- [21] S. Behera, A. Pradhan, and R. Dash, "Deep Neural Network Architecture for Anomaly Based Intrusion Detection System," *5th International Conference on Signal Processing and Integrated Networks (SPIN 2018)*, pp. 270- 274, 2018.

- [22] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Proc. Int. Conf. Neural Inf. Process.* pp. 858–866, 2017.
- [23] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT," *Sensors*, vol. 17, no. 9, p. 1967, Aug. 2017.
- [24] F. A. Khan, A. Gumaiei, A. Derhab, and A. Hussain, "TSDL: A twostage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.
- [25] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, Oct. 2016.
- [26] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
- [27] N. Gao, L. Gao, Q. Gao, and H. Wang, "An Intrusion Detection Model Based on Deep Belief Networks," *Proc. 2<sup>nd</sup> International Conference on Advanced Cloud and Big Data*, Huangshan, China, pp. 247-252, 2014.
- [28] B. Deokar and A. Hazarnis, "Intrusion Detection System using Log Files and Reinforcement Learning," *International Journal of Computer Applications*, vol. 45, no. 1919, pp. 28–35, 2012.
- [29] A. Servin and D. Kudenko, "Multi-agent reinforcement learning for intrusion detection: A case study and evaluation," *Frontiers in Artificial Intelligence and Applications*, vol. 178, pp. 873–874, 2008.
- [30] R. Elderman, L. J. J. Pater, A. S. Thie, M. M. Drugan and M. A. Wiering, "Adversarial reinforcement learning in a cyber security simulation," *ICAART 2017- Proceedings of the 9th International Conference on Agents and Artificial Intelligence*, pp. 559–566, 2017.
- [31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
- [32] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, 1996.
- [33] N. Moustafa, J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network i Intrusion Detection Systems (UNSW-NB15 Network Data Set)," *Proceedings of the 2015 IEEE Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, 2015.
- [34] KDD Cup 1999. Available online: <https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html> (Accessed on 20 November 2020).
- [35] 1998 DARPA Intrusion Detection Evaluation Dataset. Available online: <https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset> (Accessed on 20 November 2020).

- [36] M. Tavallaee, E. Bagheri, W. Lu, and A.A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.
- [37] H. Koduvely, "Github repository, gym-network\_intrusion," Retrieved from [https://github.com/harik68/gym-network\\_intrusion](https://github.com/harik68/gym-network_intrusion), 2018.
- [38] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "An Experimental Study on Hyper-parameter Optimization for Stacked Auto-Encoders," *Proc. IEEE Congress on Evolutionary Computation*, Rio de Janeiro, Brazil, pp. 1-8, 2018.
- [39] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Shallow neural network with kernel approximation for prediction problems in highly demanding data networks," *Expert Systems with Applications*, vol. 124, pp. 196-208, 2019.
- [40] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, "Network intrusion detection based on supervised adversarial variational auto-encoder with regularization," *IEEE Access*, vol. 8., pp. 42169-42184, 2020.