Enabling intrusion detection systems with dueling double deep *Q*-learning

Youakim Badr
The Pennsylvania State University, Malvern, Pennsylvania, USA

Received 22 May 2022
Revised 19 June 2022
23 June 2022
Accepted 23 June 2022

Abstract

Purpose – In this research, the authors demonstrate the advantage of reinforcement learning (RL) based intrusion detection systems (IDS) to solve very complex problems (e.g. selecting input features, considering scarce resources and constrains) that cannot be solved by classical machine learning. The authors include a comparative study to build intrusion detection based on statistical machine learning and representational learning, using knowledge discovery in databases (KDD) Cup99 and Installation Support Center of Expertise (ISCX) 2012.

Design/methodology/approach – The methodology applies a data analytics approach, consisting of data exploration and machine learning model training and evaluation. To build a network-based intrusion detection system, the authors apply dueling double deep *Q*-networks architecture enabled with costly features, k-nearest neighbors (*K*-NN), support-vector machines (SVM) and convolution neural networks (CNN).

Findings – Machine learning-based intrusion detection are trained on historical datasets which lead to model drift and lack of generalization whereas RL is trained with data collected through interactions. RL is bound to learn from its interactions with a stochastic environment in the absence of a training dataset whereas supervised learning simply learns from collected data and require less computational resources.

Research limitations/implications – All machine learning models have achieved high accuracy values and performance. One potential reason is that both datasets are simulated, and not realistic. It was not clear whether a validation was ever performed to show that data were collected from real network traffics.

Practical implications – The study provides guidelines to implement IDS with classical supervised learning, deep learning and RL.

Originality/value – The research applied the dueling double deep *Q*-networks architecture enabled with costly features to build network-based intrusion detection from network traffics. This research presents a comparative study of reinforcement-based instruction detection with counterparts built with statistical and representational machine learning.

Keywords Cybersecurity, Reinforcement learning, Deep learning, Convolution neural network, Double deep *Q*-network, Intrusion detection, Supervised machine learning

Paper type Research paper

1. Introduction

With the exponential increase in the use of internet applications, the internet-connected systems have an increasing threat from cyber-attacks. The complexity and dynamics of cyber-attacks require intrusion detection systems (IDS) to intercept cyber-attacks at large-scale.

We are witnessing an exponential advancement in technology and its usage and with that, the data generated has been growing at an increased pace as well. With the increased data,

This search was generously supported by The Penn State Center for Security Research and Education (CSRE) – 2021 Impact grant.

Digital Transformation and Society
Vol. 1 No. 1, 2022
pp. 115-141
Emerald Publishing Limited
e-ISSN: 2755-077X
p-ISSN: 2755-0761
DOI 10.1108/DTS-05-2022-0016

the security emerges as a major source of concern and thus cyber-security becomes an important department in any industry. Therefore, it is important for any organization to have reliable systems in place to ensure security of their databases and network systems. To this end, IDS are used to monitor traffic moving on networks to identify suspicious activity and known threats. They are conceived as software applications that scan a network or a system to detect malicious activity or policy breaches. Furthermore, an ideal IDS monitors network packets inbound to the system and sends out a warning signal whenever a harmful activity is detected.

According to the National Institute of Standards and Technology (NIST) Scarfone and Mell (2007), IDS can be classified into 5 types:

(1) **Network Intrusion Detection Systems (NIDS)**: NIDS are established at a planned point while setting up a system to monitor traffic from all the devices on the network. It observes the incoming traffic and compares the same with the collection of known errors or checks for abnormal behavior. When detected, the system sends a signal to the administrator for further investigation.

(2) **Host Intrusion Detection System (HIDS):** HIDS runs on hosts or devices on the network and examines incoming and outgoing network packets. The administrator is sent a warning signal if the current snapshot of system files becomes different from a previous snapshot.

(3) **Protocol-based Intrusion Detection System (PIDS)**: PIDS incorporates a system or an agent running on servers to control and interpret protocols (like HTTPS (hypertext transfer protocol secure), etc.) between remote devices and servers to secure the web server from any malicious activity.

(4) **Application Protocol-based Intrusion Detection System (APIDS)**: APIDS is similar to the PIDS in terms of operation but it predominantly resides within a group of servers. It identifies intrusion by interpreting the exchanged packets between application specific protocols.

(5) **Hybrid Intrusion Detection System (HIDS)**: HIDS is made by a combination two or more approaches discussed above. In HIDS, the system and network data are combined to have a complete network system and thus the Intrusion Detection System is more effective when compared with the traditional approaches.

Alternatively, IDS can also be classified based upon the detection method incorporated. The two primary methods of detection are signature-based and anomaly-based. Signature-based IDS use a predefined set of instructions to look for known attack patterns. These systems can easily identify the attacks whose information is already stored in instructions, but it becomes a daunting task to identify new malware attacks on the system. On the other hand, Anomaly-based IDS first establish the baseline performance under normal operating conditions to observe the normal behavior of the network.

Supervised machine learning are extensively used to build IDS by training models on network packets (e.g. transmission control protocol (TCP), user datagram protocol (UDP)) to classify normal traffics from abnormal activities caused by unknown malware attacks. Machine learning based methods have a better generalized property in comparison to signature-based IDS. They are trained from data collected from communications or generated syntactically for simulations. On the other side, supervised machine learning requires large amount of data and hyper-tuning to obtain classifiers that generalize well on unforeseen data. In addition, they are subject to model and data drifts when they are deployed for inference in real time. Drifts imply degradation of intrusion detection classifiers' prediction power due to changes in their environments, and thus the relationships between the packet's input features and the corresponding labels (e.g., normal or abnormal). The most common ways to address the drifts are periodically refit the classifiers with updated training datasets, learn the change or weight input features.

Reinforcement learning (RL) is an alternative approach to build IDSs. RL is a branch of machine learning that trains an agent (model) to choose an optimum solution for a problem. In fact, RL agent learns from interactions with its environment to choose an optimal action not from a label as in the supervised learning case but from a time-delayed label called a reward (a scalar value). The reward informs whether the decision of classifying an action such as detecting abnormal behavior is good or bad decision. Hence, the goal of RL is to process a sequence of actions in order to maximize the reward. In contrast to supervised machine learning algorithms which require huge amount of data, RL build the training data from interactions of its agent with the environment (e.g. network).

In this study, we explore *Q-learning,* an off-policy RL algorithm to build IDS agent that detects the anomaly in network packets. These anomalies can be categorized into various attack types based on the network configuration. A *Q*-learning-based agent takes actions at random, learns the value of an action in a particular state and aims at choosing the best action to take given the current state and by maximizing the reward. In addition, we propose novel IDS agent-based on double deep *Q*-network (DQN) with dueling architecture to maximize a cumulative reward by detecting intrusions and attacks. To this end, we train deep neural networks to approximate the *Q*-value function. The state is given as input and the *Q*-value of all possible actions are generated as output. The action space is defined as the selection of right set of features and classification of network packets. The reward function is thus defined to give a negative score whenever the RL agent misclassifies a network packet or a partial negative score when it selects a set of features which do not improve the model performance.

We have also compared the performance of our proposed double dueling deep *Q*-network with retrace architecture and the results from two machine learning models: convolution neural networks (CNN) and *K*-nearest neighbors (*K*-NN). We thus observed that the convergence obtained by ouble dueling DQN architecture is faster and better. Our results are also aligned with similar studies (Lopez-Martin, Carro, Sanchez-Esguevillas, & Lloret, 2019).

The remaining paper is organized as follows. Section 2 introduces basic concepts of RL and presents algorithms used in this study. Section 3 sheds light on the related works on IDS and forces on Deep Reinforcement Learning (DRL) based intrusion detection. Section 4 presents the data analytics methodology, introduces the proposed dueling double deep *Q*-networks architecture enabled with costly features to build a RL based IDS architecture. In addition, Section 4 presents the exploratory data analysis and data preprocessing. In Section 5 covers a range of models from the classical supervised machine learning trained on the knowledge discovery in databases (KDD) Cup 99 dataset (*K*-NN, (Support-Vector Machines) SVM) and installation support center of expertise (ISCX) datasets (CNN, *K*-NN and SVM), and the double dueling deep *Q*-network with retrace models and compares the results from different models. Finally, Section 6 concludes the study and provides future research directions.

2. Reinforcement learning preliminaries

RL is a branch of machine learning based on rewarding desired behaviors from sequences of decisions taken throughout the course of training. As shown in Figure 1, RL is described by concepts of state, action and reward and environment. An agent learns to behave in an environment by choosing actions and observe the results from taking such actions. For each good action, the agent gets a reward, and for each bad action, the agent receives penalty. The

RL's architecture is thus based on interactions between the agent and the environment, and the cumulative reward obtained at the end of the entire training episode. RL training consists of trials and errors to come up with a solution to a problem. The goal of RL training is to maximize the total reward.

DRL use deep neural networks to tackle problems that considered too complex for classic RL algorithms.

2.1 Elements of reinforcement learning

For each and every RL system, there are six components to define in order to learn to solve problems. **According to Sutton and Barto (2018), these components include:**

(1) **Environment:** Environment (E) provides information about the *state* of the system. The environment is assumed to be stochastic, which means it is random in nature.

(2) **Agent:** An entity that can perceive or explore the environment and act upon it by selecting an action.

(3) **Actions**: Actions (A) causes the environment to transit from current state to a new state. Actions could be discrete (e.g. turning on/off) or continuous (e.g. turning a wheel).

(4) **States:** A state (S) is an immediate situation returned by the environment after each action taken by the agent.

(5) **Reward:** Based on the current state and the performed action, the agent receives rewards from the environment as a result of the agent's action.

(6) **Policy:** Based on the current state, a policy (π) is a strategy used by the agent to choose the next action.

There are four categories of RL algorithms to map states to actions in a way that maximizes rewards, namely policy-based algorithms (e.g. REINFORCE algorithm), value-based algorithms (e.g. State Action Reward State Action [SARSA] and DQN), model-based algorithms (e.g. Monte-Carlo Tree Search) and combined methods (i.e. Actor-Critic algorithms).

In this paper, we focus on the value-based functions by which the RL agent evaluates the possible long-term return rewards based on the current state and actions under a policy π. The value-based functions are divided into two categories: Q-values and V-values. Put it simply, the Q-value represents the value of choosing a specific action at a given state whereas the V-value represents the value of the given state regardless of the action taken. In addition, the advantage-value (A-value) estimates how advantageous selecting an action is relative to the other actions at the given state. The A-value is the difference between the Q-value and the V-value. Q functions estimate the expected return on state-action pairs. Based on Q functions, state-action pairs could easily transform into an RL policy. Finally, V functions estimate the value of states.

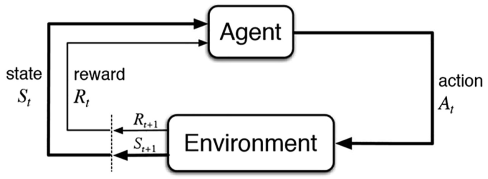


Figure 1.
The agent-environment interaction in RL

2.2 Q*-learning and deep* Q*-network*

Most existing RL algorithms work with combinations of finite or discrete actions (action space) and continuous or discrete states (states space). A typical intrusion detection system requires discrete actions (e.g. classification into two or more classes) taken in response to high-dimensional real-valued input features (e.g. continuous states space).

Q-learning is an off-policy RL algorithm which is constrained to discrete states and discrete actions and generates a table of state-action-reward values based on the interactions between the environment and the agent. One approach to formulate problems with continuous states space and continuous states space is to extend *Q*-learning with deep neural networks (called Deep *Q*-Networks or DQN for short) to process continuous input features and approximate the table of state-action-reward values.

As depicted in Figure 2, the *Q*-learning function learns from discrete actions that are outside the current policy to choose the best action to take given the current state. By such, *Q*-learning function learns a policy that maximizes the total reward.

Based on Mnih *et al.* (2013), the parameters used in updating the *Q*-value process are:

(1) α: refers to the learning rate taking values between 0 and 1. If the learning rate has the value of 0, the *Q*-values are never updated and the agent is learning nothing. The value of 0.9 indicates that the learning can occur quickly.

(2) γ: refers the discount factor and its value indicates that future rewards are worth less than immediate rewards. The value of 0 means that the agent cares for its immediate reward only whereas the value of 1 indicates that the agent cares for all future rewards.

(3) $\mathbf{max}_{\alpha}$: refers to the maximum reward which is attainable in the state following the current one.

Q-learning builds a memory table $Q[s, a]$ to store *Q*-values for all possible combinations of *s* and *a*. By sampling an action from the current state, the *Q*-learning function finds out the reward (*R*) and the new states. The *Q* table determines the next action *a'* to take which has the maximum $Q\,(s', a')$. By running the agent several episodes, the *Q*-learning function maintains a running average for *Q* values which converge to optimal values. *Q*-learning performs well when dealing with simple environments where the number of states and actions are relatively small.

However, nondeterministic and complex environments, where the combined number of actions and states can reach a large number, require extensive computation and storage resources to evaluate every possible state-action pair. In addition, the amount of time required to explore each state and manage large *Q*-table would be unfeasible. To deal with these cases, *Q*-learning could be supported by deep neural networks, which yields DQ, to approximate *Q* values and learn an optimal policy without going through all possible states. The *Q*-learning's

```
Algorithm 1: Q-learning: An off-policy TD control
Initialize $Q(s, a)$ arbitrarily
Repeat foreach episode do
    Initialize S;
    Repeat foreach step of episode do
        Choose $A$ from $S$ using policy derived from $Q$ (e.g., ε-greedy);
        Take action $A$, observe $R$, $S'$;
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_{\alpha} Q(S', a) - Q(S, A)]$;
        $S \leftarrow S'$;
        until S is terminal
    end foreach
end foreach
```

Figure 2.
Q-learning: an off-policy temporal difference (TD) control algorithm

table is thus replaced by deep neural networks and denoted as $Q\ (s,\ a;\ \theta)$, where θ represents the trainable parameters of the neural network.

3. Related work

There have been numerous works on building IDS based on different techniques, including classical supervised machine learning and RL to detect anomalies in network traffics. The review in (Bhuyan, Bhattacharyya, & Kalita, 2013) covers a range of anomaly-based network IDS and methods. Furthermore, it presents different detection techniques and discusses several evaluation criteria for testing performances and accuracy. Similar work was developed in the context of IoT by (Hussain, Hussain, Hassan, & Hossain, 2020) in which various machine learning and deep learning techniques were presented and discussed within the context and IoT applications and their cyber-security. A basic workflow of practical guideline to explore machine learning paradigms for networking research is discussed in (Wang, Cui, Wang, Xiao, & Jiang, 2017). In this work, a comprehensive study of advanced machine learning techniques for networking was presented, including networking measurement, prediction and scheduling.

An area of active research has emerged and aims to apply machine learning techniques to IDS by collecting and learning models from networking packets. For example, authors in (Costa, Papa, Lisboa, Munoz, & Albuquerque, 2019) present several intelligent techniques and their applied intrusion detection architectures in computer networks within the context of IoT. Yet another survey on anomaly based IDSs using unsupervised techniques (Nisioti, Mylonas, Yoo, & Katos, 2018) and exhibits the ability to detect unknown attacks, based on packet features that describe each attack class without any prior knowledge or the need of labeled data for training and testing.

Authors in (Tsai, Hsu, Lin, & Lin, 2009) focus on various machine learning (ML) models which include simple classifiers such as (K-NN), SVMs, decision trees, as well as ensemble classifiers to address intrusion detection problems. Deep learning techniques such as recurrent neural networks (RNN) and generative adversarial networks (GAN) also used to detect a wide variety of cyber-security attacks that target networks, application software, systems and identify anomalies in early stages (Berman, Buczak, Chavis, & Corbett, 2019).

One of the prominent work on the KDD Cup 1999 dataset was developed in Shone, Ngoc, Phai, and Shi (2018) where authors implemented nonsymmetric deep auto encoders and deep learning-based stacked nonsymmetric deep auto encoders for the attack classification. They describe the different attack classes and their groups. Authors in Lopez-Martin *et al.* (2019) proposed a shallow linear architecture with *multiclass hinge loss* and a k*ernel approximation-based feature transformation* to train on the NSL-KDD dataset, which is an updated version of KDD cup99 to tackle IDS trained.

Authors in Javaid, Niyaz, Sun, and Alam (2016) introduced a self-learning approach based on deep learning incorporated in intrusion detection system. This approach relies on two stages, including feature selection and attack classification to improve the performance.

Apart from classic machine learning approaches, RL also used to build IDS. Authors in Sukhanov, Kovalev, and Stỳskala (2015) used temporal-difference RL enabled with observed states stored in look up tables. This approach is inspired by the Markov reward modeling with least-square temporal-difference learning.

A kernel-based RL approach has been also proposed by Xu and Luo (2007) to detect anomalies in host-based IDS which improved the generalization capabilities of RL in large and nonlinear spaces, a multi-agent RL for IDS proposed by Servin and Kudenko (2005) with look-up tables.

Recently, many approaches have been proposed, exploring different techniques for different types of IDS (Phadke, Kulkarni, Bhawalkar, & Bhattad, 2019; Lansky *et al.*, 2021). In particular, anomaly-based network intrusion detection emerged as an important research and development direction of intrusion detection (Bhuyan *et al.*, 2013; Yang *et al.*, 2022). We also observed the emergence of specialized IDS in domains such as IoT (Tiwari & Narain, 2021), smart phones (Ahmad, Shah, & Al-Khasawneh, 2021), vehicles (Badukale, Thorat, & Rojatkar, 2021) and smart grids (Liu, Hagenmeyer, & Keller, 2021) just to mention a few.

Within the context of IDS, the side-by-side comparison and evaluation of these representative approaches and techniques is a drastic challenge. Firstly, these approaches revolve around different datasets collected from real-world scenarios or synthetically generated by simulators. Secondly, IDS, which are conceived with classic machine learning or RL, are completely different. Machine learning-based intrusion detections are trained on historical datasets which make them subject to drift and lack of generalization to detect new attacks whereas RL trains agents with data collected through interactions with the environment. This leads to problems if agents collect poor quality data from trajectories with no rewards.

To the best of our knowledge, an end-to-end study to compare these approaches with the same datasets is not yet available. In addition, the shortcomings of DQN based intrusion detection to handle the overestimation of Q-values remains an open problem. We develop our study by considering two benchmark datasets used by the intrusion detection community: the KDD Cup 99 dataset (Tavallaee, Bagheri, Lu, & Ghorbani, 2009) and the ISCX 2012 dataset (Shiravi, Shiravi, Tavallaee, & Ghorbani, 2012) to build classical machine land RL models and compare their performance. In addition, recent variants of deep Q networks (DQN), like double DQN, dueling DQN Sewak (2019) are not yet applied to intrusion detections. In our study, we build dueling double deep Q-Learning based intrusion detections and compare its performance with classical machine learning.

4. Methodology

Our methodology considers two benchmark datasets; the KDD Cup 99 dataset and ISCX datasets to build and compare NIDS built with classical supervised machine learning (CNN, K-NN and SVM) and the double DQN with dueling architecture. These datasets are often selected by researchers for the purpose of simulation for network IDS and used in many studies during last two decades along with the citation in many studies. The ISCX is a benchmark intrusion detection dataset of synthetically recorded packets during seven days, replicating real time network traffics whereas the KDD Cup 99 is seven weeks of network traffics acquired from raw TCP dump data, simulating a typical United States (U.S.) Air Force local-area network (LAN). The methodology also applies a data analytics approach, consisting of data exploration and preprocessing, machine learning training and evaluation.

In this section, we introduce our double DQN with dueling architecture for intrusion detection and the preprocessing steps of datasets; KDD Cup 99 and ISCX 2012. In Sections 5 and 6, we respectively train classical machine learning models and the double DQN with dueling system, and compare their performance to detect attacks.

4.1 Double DQN with dueling-based IDS

The global architecture for anomaly-based intrusion detection using DRL is illustrated in Figure 3. The architecture aims to build robust IDS by implementing a double DQN with dueling. The architecture involves the observation and interpretation of network traffics in real time. By integrating, deep learning and RL, it becomes possible to detect unknown malware rapidly. As demonstrated in the evaluation section, the double DQN with dueling tends to convergence faster with improved capabilities of detecting anomalies in network traffics.

In Figure 3, networking packets received or transmitted from devices are intercepted and forwarded for data preprocessing and featuring engineering before stored in a database for

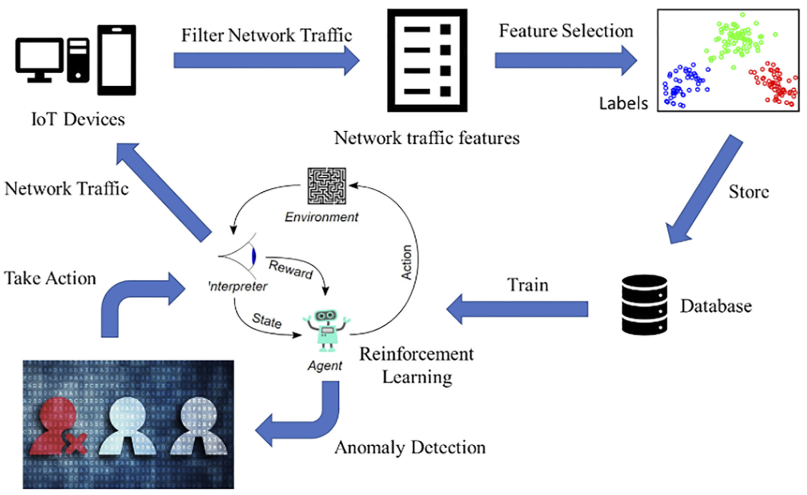


Figure 3.
Global architecture for IDS enabled with DRL

training. The RL based IDS system learns from packets to detect anomalies. The learned policy is then used for evaluation against realistic scenarios and deployment to automate decision-making or inform network administrators the occurrence of an anomaly.

More precisely, we propose a double DQN with dueling, retrace and classification with costly features (CwCF) based neural networks to implement the RL based IDS system in the global architecture. The proposed architecture is depicted in Figure 4 and consists of four sub-neural networks to respectively process the input features with masks, and estimate *V*-values and *A*-values and finally output the *Q*-values.

In the context of intrusion detection, the huge amount of data collected from network packets and state-action pairs make the *Q*-learning a computationally intensive task and the state-action table (*Q* table) is extremely large. The previous section discusses DQN, a variant of *Q*-learning, by utilizing neural networks to approximate *Q*-values and thus replacing the *Q* table. A major problem with *Q*-learning is the overestimation bias due to the *max* operator which overestimates *Q-Values* for certain actions and leads to a poor performance in some stochastic environments. The solution for this problem is double *Q*-learning (Hasselt, Guez, & Silver, 2016) which uses two estimators Q^A and Q^B instead of one estimator, *Q-Value,* for each state-action pair. Double *Q*-learning focuses on finding action a^* that maximizes Q^A in the state next states*'* and then uses a^* to get the value of *QB-Value* (see Figure 5).

From an implementation perspective, the double *Q*-learning combines the two target networks both with and without delay to reduce the bias induced by the max function. The

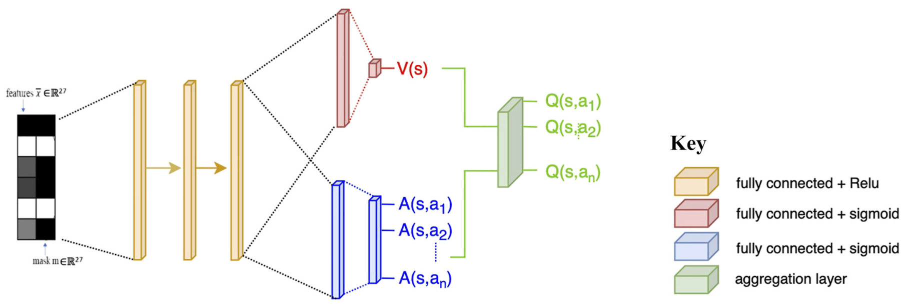

features $\bar{x} \in \mathbb{R}^{27}$
mask $m \in \mathbb{R}^{27}$
V(s)
Q(s,a1)
Q(s,a2)
Q(s,an)
A(s,a1)
A(s,a2)
A(s,an)
Key
fully connected + Relu
fully connected + sigmoid
fully connected + sigmoid
aggregation layer

Figure 4.
Double DQN with dueling, retrace and classification with costly features

double Q-learning maintains two Q-value functions Q^A and Q^B, each one gets update from the other for the next state. The update consists of finding the action a* that maximizes Q^A in the next state (Q (s', a^*) = max Q (s', a)), then use a^* to get the value of Q^B (s', a^*) in order to update Q^A (s, a).

In addition to a Double Q-learning we have included a **Dueling** architecture Wang *et al.* (1995) which is a type of Q-network with two streams to separately estimate the representation of state values and state-dependent action advantages. The motivation behind the dueling architecture is that for advanced IDS, it is unnecessary to know the value of each action at every time step or where actions might not always affect the environment in meaningful ways.

The proposed double Q-learning with dueling is also coupled with retrace Munos, Stepleton, Harutyunyan and Bellemare (2016) which is a method to efficiently utilize long traces of experience with truncated importance sampling.

As depicted in Figure 4, the right side of the architecture shows the dueling network with two branches to separately estimate state-value (red neural network) and the advantages (blue neural network) for each action; the last module (green neural network) combined the two branches with an aggregating layer to output an estimate of the state-action value function Q. Formally, the forward mapping to combine them is given by:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \max_{a'\in|\mathcal{A}|} A(s,a;\theta,\alpha)\right) \tag{1}$$

Alternatively, Wang *et al.* (1995) represents this formulation in their experiments as:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a;\theta,\alpha)\right) \tag{2}$$

This formulation shows that the advantage function has zero advantage for the chosen action. In addition, the average operator is used instead of a maximum to maintain the optimization stable.

Moreover, the optimal action a^* is given by:

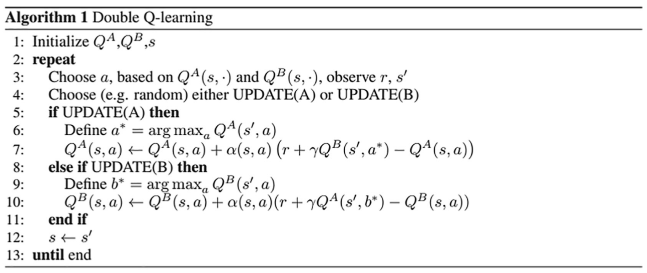
$$ErrorconvertingfromMathMLtoLaTeX \tag{3}$$

The dueling architecture and the standard DQN architecture have a similar input-output interface, and thus, their training remains similar.

The loss function to update the neural networks parameters θ is given by:

$$L(\theta) = \frac{1}{N}\sum_{i\in N}\left(Q_\theta(s_i,a_i) - Q'_\theta(s_i,a_i)\right)^2 \tag{4}$$

Where $Q'_\theta = R(s_t,a_t) + \gamma \max_{a'_i} Q_\theta\left(s'_i,a'_i\right)$



```
Algorithm 1 Double Q-learning
1: Initialize $Q^A$,$Q^B$,$s$
2: repeat
3:   Choose $a$, based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$, observe $r$, $s'$
4:   Choose (e.g. random) either UPDATE(A) or UPDATE(B)
5:   if UPDATE(A) then
6:     Define $a^* = \arg\max_a Q^A(s',a)$
7:     $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\left(r + \gamma Q^B(s',a^*) - Q^A(s,a)\right)$
8:   else if UPDATE(B) then
9:     Define $b^* = \arg\max_a Q^B(s',a)$
10:    $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s',b^*) - Q^B(s,a))$
11:  end if
12:  $s \leftarrow s'$
13: until end
```

Figure 5.
Algorithm taken from double Q-learning by Hado van Hasselt

Since the intrusion detection is a classification problem, each feature from raw packets captured from the network communication can be acquired for a cost and the goal is to optimize a trade-off between the expected classification error and the feature cost. The cost come in the form of time required to process packets by several layers of machine learning algorithms, or performance such as sending queries to vulnerability databases with limited number of queries (scarce resources) etc. Our double Q-learning with dueling and retrace is extended to incorporate CwCF proposed in (Janisch, Pevný, & Lisý, 2019).

At each step, the agent takes a decision and takes two types of actions: selecting a feature or making a classification. The agent receives a negative reward for requesting to acquire a new feature. In this case, the reward equals to the feature cost. The agent also receives rewards for correct and wrong classification.

For this model we will consider a sample $(x, y) \in D$ where the set (x, y) resembles a sample and D represents the entire data distribution.

Let X and Y denote respectively the packet features and their corresponding labels (binary classification).

Vector $x \in X \subseteq \mathbb{R}^n$ contains feature values, where x_i is a value of feature $f_i \in F$, where $F = \{f_1, \ldots, f_n\}$, n is the number of features and $y \in Y$ is a class.

Let $c: F \rightarrow \mathbb{R}$ be a function mapping a feature f into its real-valued cost $c(f)$, and let $\lambda \in [0, 1]$ be a cost scaling factor.

The neural network for classifying one sample for example, is a parameterized couple of functions (y_θ, z_θ) such as $y_\theta: X \rightarrow Y, z_\theta: X \rightarrow c(F)$, where y_θ classifies and z_θ returns the features used in the classification.

As defined by Janisch *et al.* (2019), the objective function is to find parameters θ that minimize the expected classification error (l) along with λ scaled expected feature cost:

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{|D|} \sum_{(x,y) \in D} \left[l(y_\theta(x), y) + \lambda \sum_{f \in z_\theta(x)} c(f) \right] \tag{5}$$

Given the state space S, the set of actions A, and the reward r transition function t, the states s based on the sample (x, y) is defined as $s = (x, y, \bar{F}) \in S$ where $\bar{F}$ is currently selected set of features.

An action $a \in A = A_c \cup A_f$ could be one of the classification actions $A_c = Y$, or feature selecting actions $A_f = F$. Obviously, the set of available feature selecting actions is limited to features not yet selected by the agent.

In the sequential steps, the agent selects a feature at each step to make a class prediction. By such, the agent receives only an observation $o = \{(x_i, f_i) \mid \forall f_i \in \bar{F}\}$, that is, the selected parts of x without the label.

(1) Classification actions A_c terminate the episode and the agent receives a reward of 0 in case of correct classification or a reward of -1 in case of wrong classification.

(2) Feature selecting actions A_f reveals the corresponding value x_i and the agent receives a negative reward of $-\lambda c(f_i)$.

Reward function $r: S \times A \rightarrow R$ is defined by Janisch *et al.* (2019) as:

$$r\left((x, y, \bar{\mathcal{F}}), a\right) = \begin{cases} -\lambda c(f_i) & \text{if } a \in \mathcal{A}_f, a = f_i \\ 0 & \text{if } a \in \mathcal{A}_c \text{ and } a = y \\ -1 & \text{if } a \in \mathcal{A}_c \text{ and } a \neq y \end{cases} \tag{6}$$

The CwCF is implemented in the left side of Figure 4, the input layer consists of the feature vector x concatenated with binary mask m to specify the number of features selected for a particular episode. The input *x* and *m* are fed into the neural networks which gives out the output as Qf and Qc corresponding to *Q* value for feature selection and *Q* value for classification respectively.

In the next sections we explore the datasets and discuss each part of the global architecture in details, including network datasets selecting, data preprocessing steps, feature selection and the DRL model.

4.2 Intrusion detection datasets

In order to build robust RL-based IDS based on supervised machine learning, datasets should not only be labeled (dependent variable) to classify the status of network as normal or under attack state but also consider the following requirements (Ring, Wunderlich, Scheuring, Landes, & Hotho, 2019):

(1) An inherent problem with most network traffic flow is the presence of more normal user behavior than attack traffic. Imbalanced datasets should be balanced by appropriate techniques such as weighted classes to have the same number of data points from each class.

(2) Network-based datasets should be publicly available to be used a benchmark for comparing intrusion detection methods from across different approaches and techniques.

(3) The datasets should also have a high-data volume to obtain significant results when training with any machine learning model (Shone *et al.*, 2018; Injadat, Salo, Nassif, Essex, & Shami, 2018).

Considering these requirements, two datasets are considered in this study: the KDD Cup 99 dataset (Tavallaee *et al.*, 2009) and the ISCX 2012 dataset (Shiravi *et al.*, 2012) which satisfy most of the above-mentioned requirements. These datasets consist of a number of features, numeric and categorical which are tagged with a dependent variable that determines whether the given state is normal or under attack.

4.2.1 KDD Cup 1999 dataset. The KDD Cup 1999 dataset is a well-recognized standard in the research of IDS (Tavallaee *et al.*, 2009). The dataset consists of high data volume of samples and includes a wide variety of intrusions simulated in a military network environment.

The original KDD Cup 1999 dataset is described in (Lopez-Martin *et al.*, 2019) and initially consists of 42 features in total out of which 34 features are continuous while the rest of the features represented as symbolic features. The entire dataset consists of 4,898,431 records, representing 7 weeks of network traffic. For our research we generate samples from the original dataset with dependent variable consisting of 39 classes to build a training dataset of 311,029 samples and testing dataset of 494,021 samples. The training and testing dataset do not have overlapping samples. All samples are unique. Unlike binary classification detection developed in Xu and Luo (2007), Servin and Kudenko (2005) and Javaid *et al.* (2016), we are implementing the multi-class classification by considering all labels during the training process.

Figure 6 shows the frequency of classes in the KDD Cup 1999. The number of records is transformed to a logarithmic scale to better visualize the imbalanced training and testing datasets. Table 1 summarizes attacks into four categories in both training and testing dataset.

4.2.2 ISCX 2012 dataset. The ISCX 2012 dataset consists of labeled network traces, including full packet payloads in the *pcap* format (Shiravi *et al.*, 2012). The dataset was generated by capturing traffic in an emulated network environment over one week with

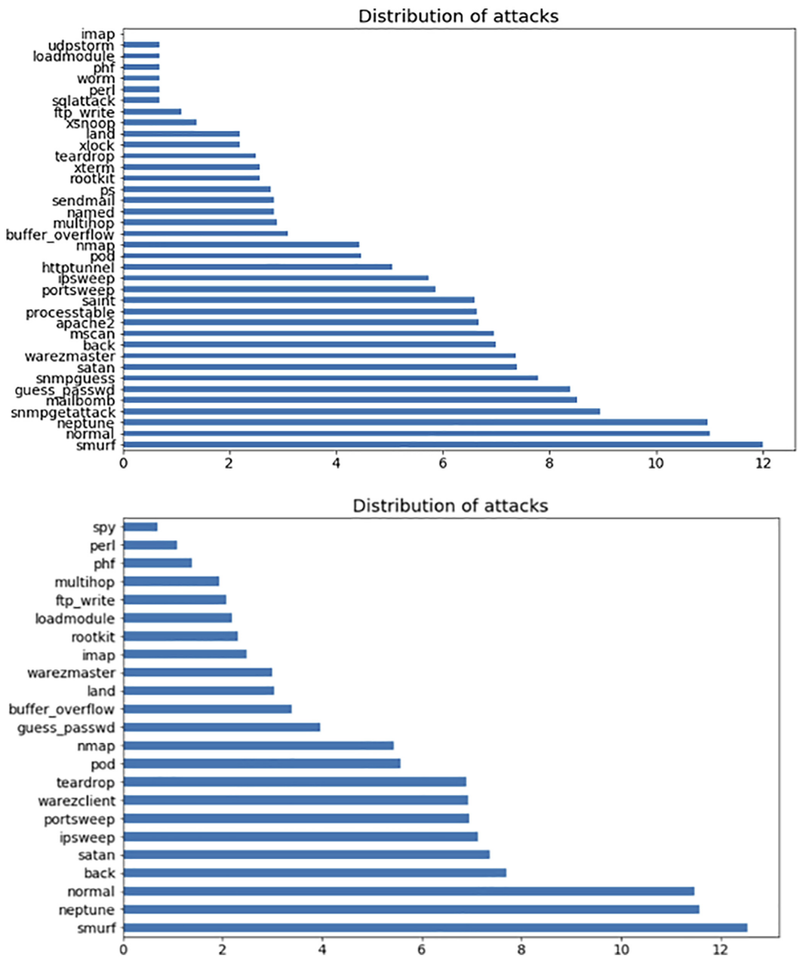


Figure 6.
Frequency distribution among classes in train and test KDD Cup 1999 dataset (logarithmic)

normal as well as malicious network behaviors. The dataset is based on the concepts of profiles, which contain descriptions of intrusions and abstract distribution models for applications, protocols or network entities. For example, α profiles define attack scenarios while β profiles characterize normal user behavior like browsing the Web. The ISCX 2012 dataset consists of 481,677 records for training and 160,519 records for validation and testing.

Figure 7 shows the frequency of classes in the ISCX 2012 training and test datasets. In addition, Wireshark, a network protocol analyzer visualizes the entire conversations and the exchanged packets from the ISCX 2012 in Figure 8.

4.2.3 Data preprocessing – KDD Cup 1999 dataset. The data preprocessing of KDD Cup 1999 dataset consists of removing duplicate rows, checking skewness, transforming categorical variables to numeric variables, applying features importance and splitting data into training dataset and testing dataset.

In this phase, the dependent variable has 38 classes for the 311,029 records. The distribution of classes per samples is displayed in Figure 6 and shows that the majority of the

Attack category	Attack type	KDD Cup 99 Test set	KDD Cup 99 Training set
Denial of service	Neptune	107,201	58,001
	Smurf	164,091	28,0790
	Pod	264	87
	Teardrop	979	12
	Land	21	9
	Back	2,203	1,098
	Apache2	–	794
	Udpstorm	–	2
	Process-table	–	759
	Mail-bomb	–	5,000
User to root	Buffer-overflow	30	22
	Load-module	9	2
	Perl	3	2
	Rootkit	10	13
	Spy	2	–
	Xterm	–	13
	PS	–	16
	HTTP-tunnel	–	158
	SQL-attack	–	2
	Worm	–	2
	Snmp-guess	–	2,406
Remote to local	Guess-password	53	4,367
	FTP-write	8	3
	IMPA	12	1
	Phf	4	2
	Multihop	7	18
	Warezmaster	20	1,602
	Warezclient	1,020	–
	Snmpgetattack	–	7,741
	Named	–	17
	Xlock	–	9
	Xsnoop	–	4
	Send-mail	–	17
Probe	Port-sweep	1,040	354
	IP-sweep	1,247	306
	Nmap	231	84
	Satan	1,589	1,633
	Saint	–	736
	Mscan	–	1,053

Table 1. Distribution of attacks into four attack categories

samples are observed across smurf, normal and neptune classes. It is worth noting that several classes in the network status column are not attacks like Internet Message Access Protocol (IMAP), perl, multihop, etc.

Supervised learning and RL trained on this dataset consider all these classes which make our approach is slightly different to the prior work in the state of the art where the different classes are clubbed into a binary classification of normal or attack classes.

After examining attack types in the dependent variable, all the remaining features were then carried forward to perform skewness and kurtosis test to understand the data distribution. Skewness values between 0.5 and 0.5 are considered to represent normal distribution. Features like d, src_bytes, dst_bytes, hot have skewness values over 70 which indicates highly skewed data may lead to biased models in prediction. To treat the skewness, box-cox transformation is applied to get normal distributions.

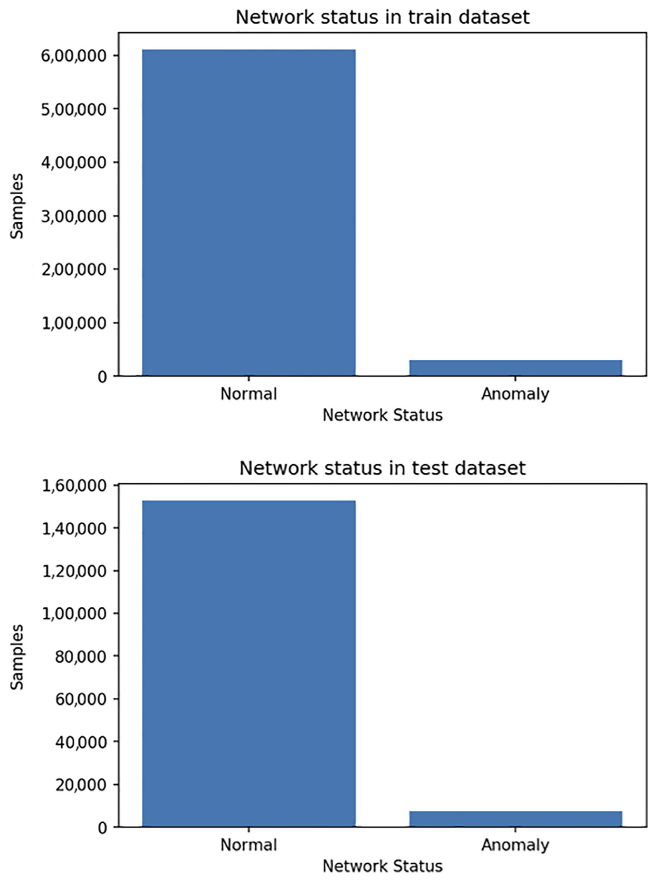


Figure 7.
Frequency distribution among classes in train (left) and test ISCX 2012 dataset

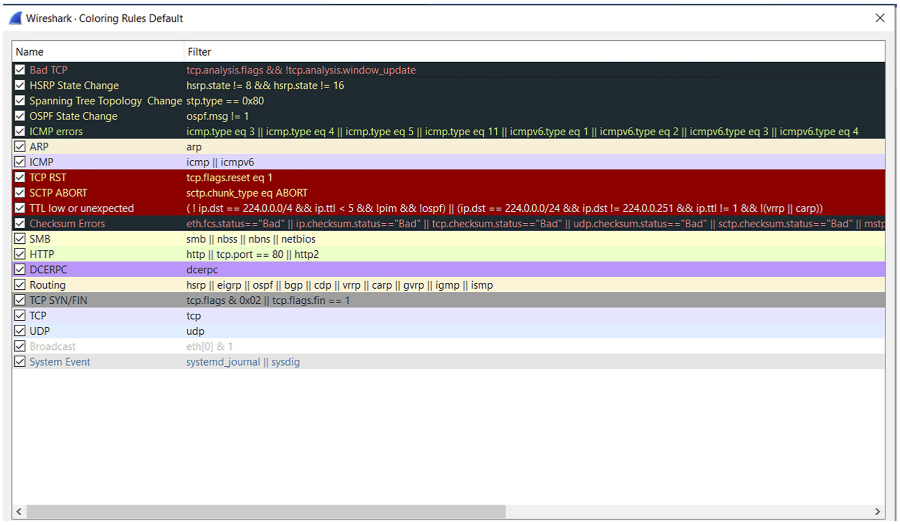
Since many features have different ranges of values, we standardize features by removing the mean and scaling to unit variance. In addition, qualitative features were encoded as numeric features since the machine learning models expect numeric features for training and evaluation. By such, the dataset has 39 features in total with all variables, consisting of numerical features.

The next important step in the exploratory data analysis is to identify the most important features to reduce the complexity of trained models and improve their performance. The application of the principal component analysis (PCA) identifies a minimum set of features that can explain the main axes of variances within the data. Figure 9 shows that the top 27 features explain 99% of the variance in the data and thus these features are selected to build intrusion detection classifiers.

Finally, the preprocessed dataset is then divided into train and validation sets with a 75:25 ratio. The obtained datasets include 248,823 records for the train data, 62,606 records for the validation data and 492,999 records for the test data.

4.2.4 Data preprocessing – ISCX 2012 dataset. The ISCX 2012 dataset consists of labeled network traces, including packet payloads in pcap format.

As illustrated in Figure 10, the pcap file consists of the file header (24 bytes), followed by the Pcap packet header (16 bytes). The Pcap packet data can have a varying length. The



Wireshark · Coloring Rules Default

Name	Filter
☑ Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update
☑ HSRP State Change	hsrp.state != 8 && hsrp.state != 16
☑ Spanning Tree Topology Change	stp.type == 0x80
☑ OSPF State Change	ospf.msg != 1
☑ ICMP errors	icmp.type eq 3 || icmp.type eq 4 || icmp.type eq 5 || icmp.type eq 11 || icmpv6.type eq 1 || icmpv6.type eq 2 || icmpv6.type eq 3 || icmpv6.type eq 4
☑ ARP	arp
☑ ICMP	icmp || icmpv6
☑ TCP RST	tcp.flags.reset eq 1
☑ SCTP ABORT	sctp.chunk_type eq ABORT
☑ TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf) || (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !(vrrp || carp))
☑ Checksum Errors	eth.fcs.status=="Bad" || ip.checksum.status=="Bad" || tcp.checksum.status=="Bad" || udp.checksum.status=="Bad" || sctp.checksum.status=="Bad" || mstp
☑ SMB	smb || nbss || nbns || netbios
☑ HTTP	http || tcp.port == 80 || http2
☑ DCERPC	dcerpc
☑ Routing	hsrp || eigrp || ospf || bgp || cdp || vrrp || carp || gvrp || igmp || ismp
☑ TCP SYN/FIN	tcp.flags & 0x02 || tcp.flags.fin == 1
☑ TCP	tcp
☑ UDP	udp
☑ Broadcast	eth[0] & 1
☑ System Event	systemd_journal || sysdig

Figure 8.
An instance from the Wireshark to display different network packets

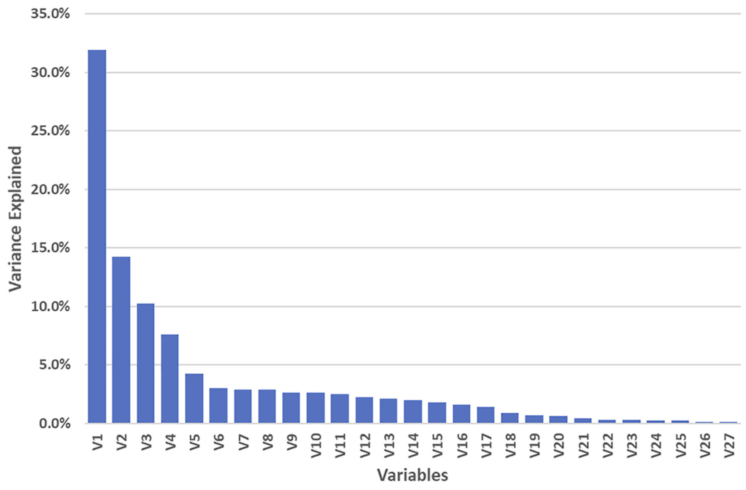


Figure 9.
PCA explained variances

global file headers are further divided into 7 segments, referring to magic number (4 bytes) followed by the major and minor numbers (4 bytes). The next set of elements includes GMT local, time stamp, max length and data link type of varying lengths. The pcap packet header consists of Timeval, capture length and data link type. An example of a pcap is shown in (see Figure 11).

In the data preprocessing, the Wireshark interface is used to load the ISCX 2012 dataset and export the required fields to CSV format. Wireshark also shows a typical analysis of network packets (see Figure 8) where different network packets are color coded based on their status of the network packet.

The dataset initially consists of 20 features ranging from appName, source bytes, destination bytes, destination packets, source packets, etc. These features also included

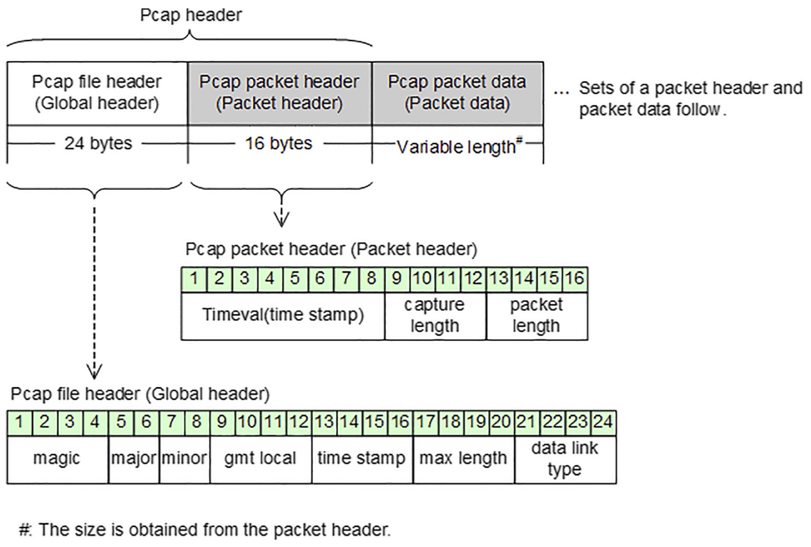
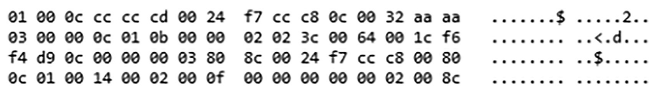

Pcap header
Pcap file header (Global header)
Pcap packet header (Packet header)
Pcap packet data (Packet data)
... Sets of a packet header and packet data follow.
24 bytes
16 bytes
Variable length#
Pcap packet header (Packet header)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Timeval(time stamp)
capture length
packet length
Pcap file header (Global header)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
magic
major
minor
gmt local
time stamp
max length
data link type
#: The size is obtained from the packet header.

Figure 10.
Structure of the pcap file format



```
01 00 0c cc cc cd 00 24  f7 cc c8 0c 00 32 aa aa   .......$ .....2..
03 00 00 0c 01 0b 00 00  02 02 3c 00 64 00 1c f6   ........ ..<.d...
f4 d9 0c 00 00 00 03 80  8c 00 24 f7 cc c8 00 80   ........ ..$.....
0c 01 00 14 00 02 00 0f  00 00 00 00 00 02 00 8c   ........ ........
```

Figure 11.
A pcap example

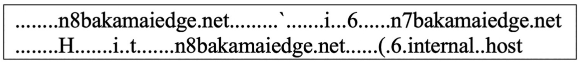
categorical variables like direction, protocol, etc. For this study, we are only concerned about the destination payload as Universal Transformation Format (UTF) feature and the network status as flag which becomes the dependent variable. The destination payload feature specifies packet's content and byte counts. The payload data is present in the pcap packet data which follows the pcap packet header in a network packet.

The length of the destination payload varies for each observation. To obtain a uniform data, we set an upper threshold of 7,500 units and concatenated its value with itself until length of the observation is just equal to the threshold of 7,500 units. Finally, we transform the obtained data into numeric by American Standard Code for Information Interchange (ASCII) transformation (Figure 12), and then to images (Figure 13). The rational to transform packets to visual presentations is motivated by our approach (see next section) to training CNN to detect patterns from images and classify then as normal or attacks or train RL policy. Each image has a shape of 50 × 50 pixels.

Finally, the ISCX 2012 dataset is divided into test, train and validation sets with a ratio of 20:60:20 which results in data splits of 160,519, 480,677 and 160,599 records respectively.

5. Supervised learning-based IDS models

In the context of IDS, classical machine learning techniques are often used in the early classification of attacks. However, due to a large number of algorithms available, the selection of the best algorithm is a challenging task.



........n8bakamaiedge.net.........`.......i...6......n7bakamaiedge.net
........H.......i..t.......n8bakamaiedge.net......(.6.internal..host

Figure 12.
Destination payload sample for normal network status

In this section, we have considered and evaluated the performance of the SVM and the *K*-NN as baseline models for the KDD Cup 1999 dataset and the ISCX 2012 dataset. These classifiers perform binary and multiclass attacks in terms of detecting whether or not the traffic has been considered as benign or an attack. In addition, we evaluated a CNN based on Visual Geometry Group (*VGG)-19* using the ISCX 2012 dataset. Table 2 shows a summary of the parameters and the datasets used to train and evaluate these modes.

5.1 Supervised learning with the KDD Cup 1999 dataset

The *K*-NN algorithm is a nonparametric supervised learning algorithm. It learns nonlinear decision boundaries based on the *k* closest training examples in a dataset. In *K*-NN, finding the value of *k* is a challenging task. A small value of *k* means that noise will have a higher influence on the decision boundary whereas a large value of *K* requires extensive computations. One potential solution consists of iterating over a number of *K* values and identifying the highest performance metrics like accuracy, F1, precision and recall.

Table 4 shows the performance of the *K*-NN models trained and tested on the KDD Cup 1999 dataset. As a result, we observed that the K value of 3 produced the best result in terms of the performance metrics (see Table 3).

SVM is one of the most popular s*upervised* learning algorithms, which is used for classification as well as regression problems. SVM is particularly effective in high dimensional spaces. However, SVM requires hyper-tuning to find the optimal hyper-parameters which result in the most accurate classification. To this end, we perform *grid*-search with 3-fold cross validation to identify the best optimal hyper-parameters by iterating over 3 kernels namely, *radial basis function, sigmoid* and *linear kernels*. In addition, the *C parameter is set to 10 which* informs the *SVM* optimization how much to avoid misclassifying each training example. By setting the value of gamma to 0.1, the grid search returns the kernel *radial basis function*, which gives the best performance with the accuracy, F1, precision and recall metrics in Table 4.

The experimental results demonstrate that the *K*-NN model with *K*-value of 3 has produced an average accuracy of 92.6% compared to the accuracy of 88.5% in case of SVM model on the KDD Cup 1999 dataset. As shown in Figure 14, the AUC values also indicate constructive performance from the *K*-NN model when compared with the SVM model.

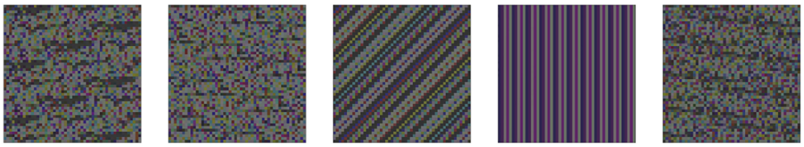


Figure 13. Image representation of the network data

Models	Parameters	KDD dataset	ISCX dataset
SVM	$kernel = linear,\ rbf$ $\gamma = 0.1\ \ C = 10,\ \ CV = 3$	X	X
K-NN	$k \in [1, 13]$	X	X
CNN	*Based on VGG*19	–	X
DQN	$epoch = 10,000,\ episode = 4,000$ $step = 5,000,\ learning\ rate = 5.0e-4$	X	–

Table 2. Hyper-parameters for supervised machine learning algorithms and DQN

Table 3.
Finding the value of K in K-NN

K-value	Accuracy	Precision	Recall	F1
2	92.3%	90.4%	84.1%	86.8%
3	*92.6%*	*90.5%*	*85.1%*	*87.4%*
4	92.1%	90.2%	83.7%	86.4%
5	*92.2%*	*90.1%*	*84.2%*	*86.8%*
6	92.1%	90.1%	83.7%	86.4%
7	92.1%	89.9%	84.1%	86.6%
8	92.1%	90.0%	84.0%	86.5%
9	92.1%	89.9%	84.1%	86.6%
10	92.1%	89.9%	83.9%	86.5%
11	92.1%	89.7%	84.1%	86.5%
12	92.0%	89.7%	83.9%	86.4%
13	92.0%	89.6%	84.1%	86.4%

Table 4.
Optimal SVM hyper-parameters with search grid

Kernel	Accuracy	Precision	Recall	F1
Radial basis function	88.5%	86.7%	74.4%	78.5%

5.2 Supervised learning with the ISCX 2012 dataset

In a similar process, we fit two binary classifiers; K-NN and SVM, to the ISCX 2012 dataset and compare their performance results. We also fit a CNN-based classifier with transfer learning on Red, Green, Blue (RGB) images generated from packets in order to detect malicious behavior using image recognition. By such, we discuss the strengths and limitations of these models with respect to ISCX 2012 dataset and KDD Cup 1999 dataset.

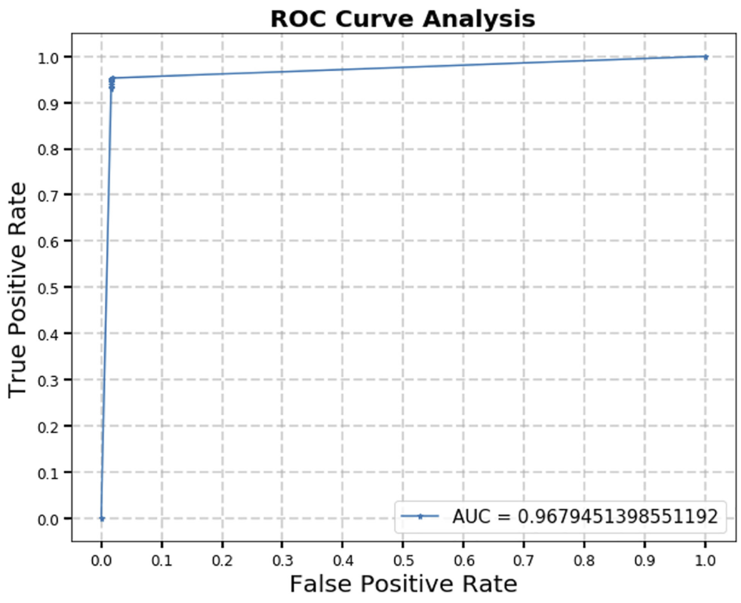
CNN-based models have been successfully applied to the KDD CUP 1999 dataset to classify individual attacks belonging to the same category such as the DoS attacks rather than distinguishing attacks from benign samples Kim, Kim, Kim, Shim and Choi (2020). Multistage deep learning with pretrained ResNet50 image recognition also proposed and trained on network features transformed into four-channel (Red, Green, Blue and Alpha) images and evaluated using two publicly available benchmark datasets, UNSW-NB15 and BOUN Ddos Toldinas *et al.* (2021). To the best of our knowledge, CNN model for binary classification on ISCX 2012 is not yet developed and compared with classical machine learning models such as SVM and K-NN.

To this end, we build an extended CNN model based on the VGG-19 pretrained model with the ImageNet database. Our model consists of 16 convolution layers, including the output with Rectified Linear Unit (ReLU) activation for classification and the binary cross-entropy loss function with RMSProp optimizer.

As part of the data preprocessing phase, each packet payload feature from the ISCX 2012 dataset is transformed into a 3-dimensional array by setting the row and width to a predefined shape of 50×50. Finally, the ISCX 2012 dataset is divided into train, validation and test sets.

The loss and accuracy plots for training our CNN model with 15 epochs are illustrated in Figure 15. In addition, Table 5 shows the accuracy, exceeding 99% along with other performance metrics. The AUC value reaches 98.58 which represents the degree or measure of separability and indicates how much the model is capable of distinguishing between normal and attack classes.

By considering the ISCX dataset, we train K-NN and SVM model and check their performance in comparison to the CNN model accuracy. In order to preprocess the dataset, we



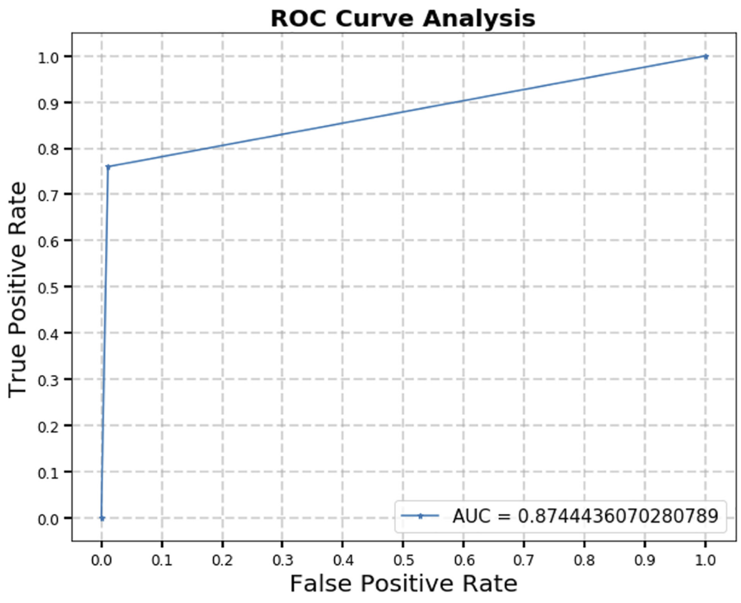


Figure 14.
ROC-AUC plots for best performing *K*-NN model (left) and SVM model (right) on KDD dataset

removed accumulative and redundant features (e.g., sourcePayloadAsBase64, destinationPayloadAsBase64, destinationPayloadAsUTF and sourcePayloadAsUTF) and selected 16 features for building the dataset based on their importance (Figure 16) (i.e., appName, Tag, destination, direction, source, protocolName, totalSourceBytes, totalDestinationBytes, totalDestinationPackets, totalSourcePackets, sourcePort, destinationTCPFlagsDescription, startDateTime, sourceTCPFlagsDescription, and

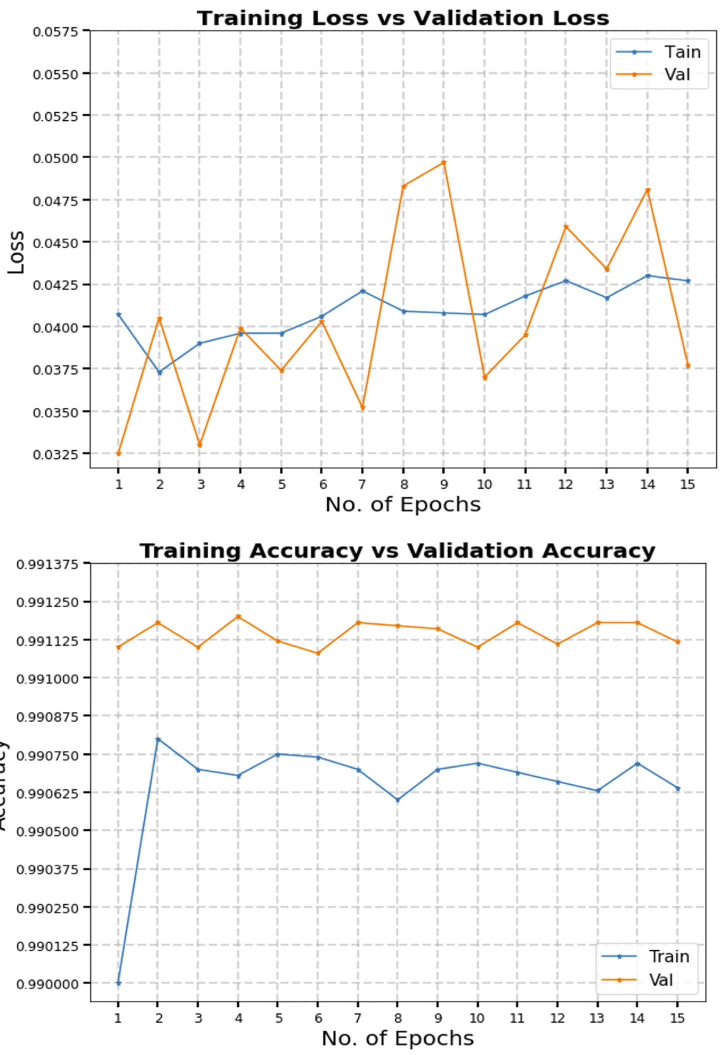


Figure 15.
Loss and accuracy plots on the ISCX 2012 dataset

Table 5.
Performance metrics for CNN on the ISCX dataset

Model	Accuracy	Precision	Recall	F1
CNN	99%	92%	98%	95%

destinationPort, stopDateTime). The preprocessing steps also include the transformations of date and time fields from string type to extract relevant information like hours and minutes, the transformation of IP addresses to integer format and removing skewness.

In order to find the best *K*-NN model, we compare different *K* values as depicted in Table 6 and observe from the model performance that there is significant difference based on the change of *K*-values. The *K* value of 2 gives the highest accuracy of 98.6%.

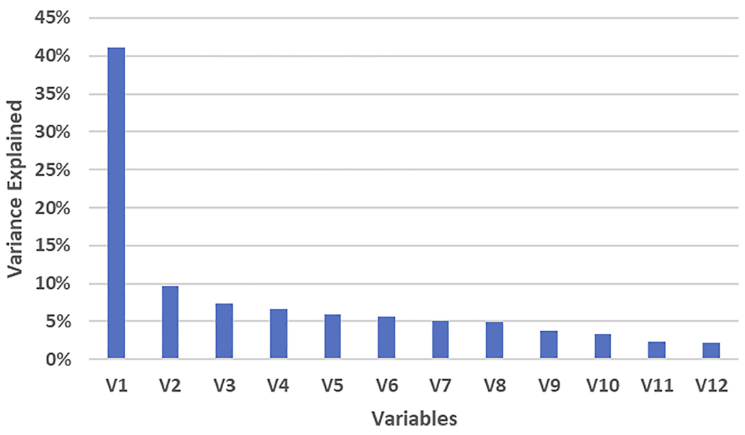


Figure 16.
PCA explained variance

K-value	Accuracy	Precision	Recall	F1
2	98.6%	90.9%	93.9%	92.4%
3	98.3%	92.4%	88.7%	90.4%
4	98.5%	91.8%	91.2%	91.5%
5	98.1%	92.8%	85.3%	88.6%
6	98.3%	92.6%	87.5%	89.8%
7	98.2%	92.6%	86.4%	89.2%
8	98.2%	92.3%	87.1%	89.5%
9	97.5%	90.6%	79.3%	83.9%
10	97.7%	91.0%	81.5%	85.5%
11	97.4%	90.5%	78.2%	83.2%
12	97.5%	90.7%	79.8%	84.3%
13	97.4%	90.8%	77.9%	83.0%

Table 6.
Performance metrics for *K*-NN on ISCX 2012 dataset

Similar results were obtained with the SVM model to predict the anomaly in the network traffic data of ISCX 2012 dataset. After an extensive grid search, the SVM algorithm returns the best parameters, which correspond to the Radial Basis Function Kernel (Table 7). The feature importance shows in Figure 16 that the destinationPayloadAsUTF feature plays an important role in predicting the anomalies in the dataset.

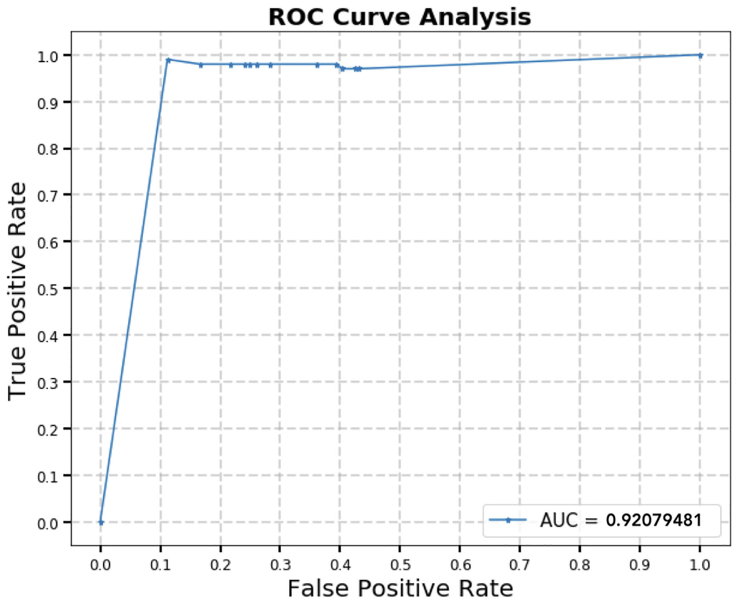
Finally, the receiver operating characteristic-area under the curve (ROC-AUC) plots in Figure 17 show the best performing *K*-NN model and SVM with AUC values of 0.9207 and 0.9374.

6. Dueling double DQN-based IDS

Since we are using the dueling double deep *Q* network to classify attacks with costly features, we generate a meta dataset, consisting of values between 0 and 1 for each feature in the KDD

Kernel	Accuracy	Precision	Recall	F1
Radial basis function	98.6%	92.3%	92.1%	92.2%

Table 7.
Performance metrics for SVM on ISCX 2012 dataset



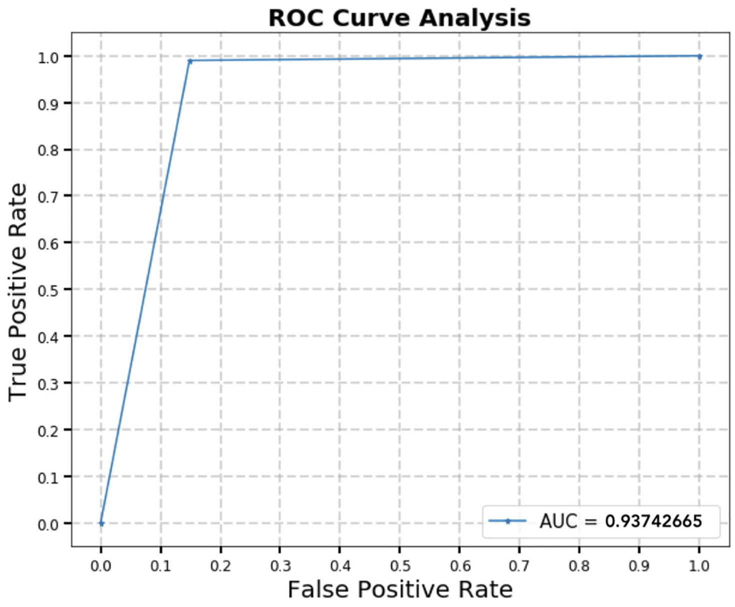


Figure 17.
ROC-AUC plots for best performing *K*-NN model (left) and SVM (right)

Cup 1999 dataset and the ISCX 2012 dataset. These values act as the cost associated with the selection of the specific features during the training.

For training the reinforcement agent, we select samples from each dataset to form a single episode. For each sample, the agent chooses a set of features and sequentially decides whether to acquire additional features and then decides how to classify the sample.

The deep Q-network consists of three fully connected hidden layers of neurons with ReLU activation functions (Figure 4). The number of neurons in each layer depends on the dataset used. All rewards are to be nonpositive, and therefore the Q-function is also nonpositive. Therefore, the agent objective throughout the reinforcement training is to achieve a total Q-value to be as close to zero as possible.

The agent has two sets of actions; action to select features and action to classify the network attack types (e.g. multi-class classification or binary classification). For a misclassification, the agent is awarded with 1 as reward. The agent is given a partial negative reward when selecting features that reduce the Q-values. For each correct classification, the agent is awarded with 0 as reward. The agent learns to bring the total reward as close to zero as possible.

The environments are set up as episodic with a short average length of episodes for short samples selected in each iteration. In addition, we use an undiscounted return with $\gamma = 1.0$ and ε – greedy policy to take random actions with the probability ε. The exploration rate is set to linearly decrease over time to its minimum value.

The hyper-parameters used in the KDD Cup 1999 dataset and the ISCX 2012 dataset are summarized in Table 8.

As depicted in Table 9, the dueling double DQN model performs fairly well on both datasets and reduces the computation time by multiple folds when its compared to Q-learning. A closer look at the performance of dueling double DQN with retrace and costly feature classification is shown in Figure 18 with reward and accuracy plots for the KDD Cup 1999 dataset.

The accuracy metrics obtained from the KDD Cup 1999 and ISCX 2012 datasets with RL and supervised machine learning (CNN, SVM and K-NN) show different performances and should not be considered as a determinant factor to build data-driven IDS.

The main difference between our proposed RL architecture and classical supervised learning is that the former is the process of learning from a training set and then applying that learning to unseen data in real time whether the latter is the process of dynamically learning by adjusting actions based on continuous feedback through trial and error in order to maximize a reward. As demonstrated, the RL approach can be used to solve very complex problems (e.g. selecting input features, considering scarce resources and constrains) that cannot be solved by classical machine learning. In addition, RL is bound to learn from its interactions with a stochastic environment in the absence of a training dataset. Conversely, supervised learning approaches simply learn from collected data and require less

Hyper-parameters	Values
Learning rate	5.00E-04
Batch size	4,000
Training length	4,000
Epochs	10,000
Exploration step	10 epochs
Exploration final	0.1

Table 8. Hyper-parameters used for initializing the dueling double DQN

Datasets	Acc. training	Acc. validation	Acc. testing
KDD Cup 1999	90.1%	93.9%	98.2%
IDS 2012	95.3%	95.3%	95.2%

Table 9. Dueling DDQN with classification with costly features (CwCF)

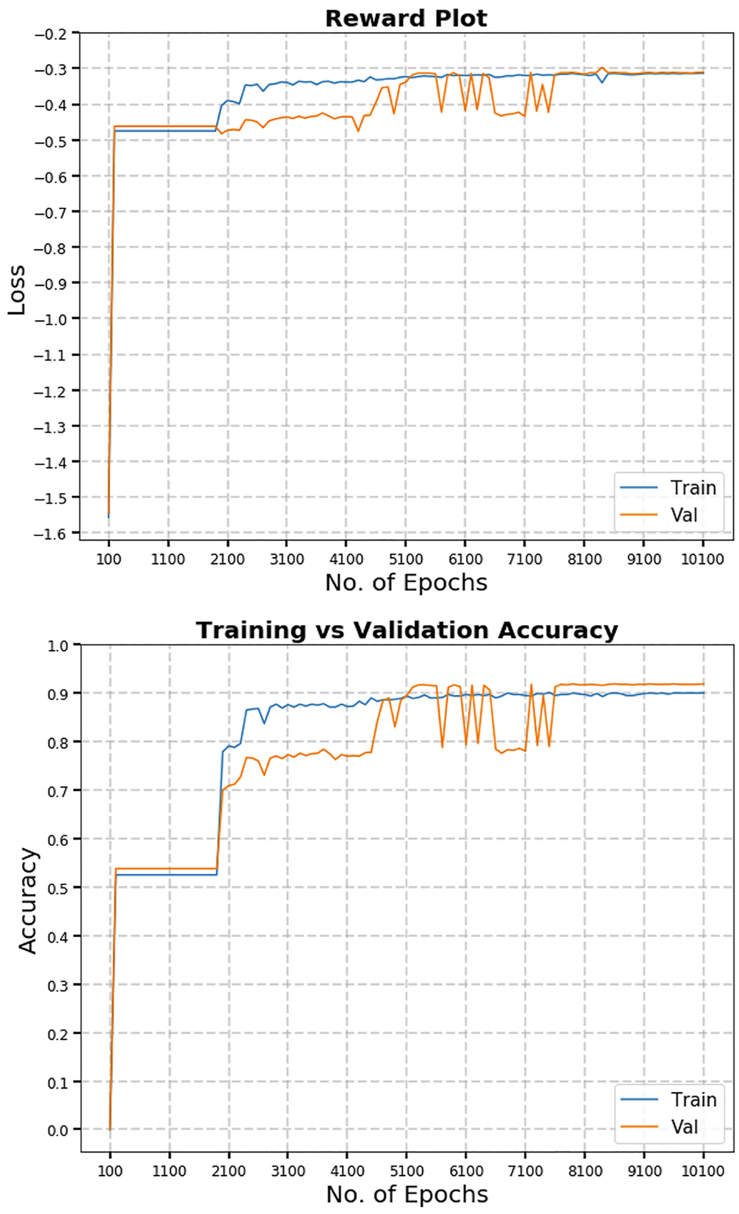


Figure 18.
Reward and accuracy plots for KDD Cup 1999 dataset

computational resources. For example, the baseline CNN model has achieved the accuracy of 99% on the IDS 2012 data training whereas the dueling double DQN enabled with classification with costly features achieved 95.3% on the training dataset. However, the accuracy of the CNN model can degrade within days of deployment and lead to incorrect predictions and significant risk exposure because production data differs from the training data and the model's accuracy will decrease (or drift) at run time.

7. Conclusion

IDS are core tools to protect computers and network-based systems by constantly monitoring protocols traffics for any potential invasions and attacks. IDS can be modified and changed according to outside and inner threats to computers and networks.

In this study, we compare performance side-by-side of machine learning models, namely *K*-NN, SVM and CNN and a dueling double deep *Q*-networks architecture feature using benchmark datasets; KDD Cup 1999 and ISCX 2012. Despite high accuracy values and performance, these models are fundamentally different yet complementary to each other.

In complex and uncertain environments, building IDS with DRL offers advantages in terms of computation as minimal updates are required when compared with traditional machine learning algorithms. Hyper-parameter tuning is another advantage while working with DRL as minimal update is required to improve performance.

Classical supervised learning models learn from historical data and continuously need to be retrained to detect new attacks. The limitation of this study is that both datasets are simulated and not realistic Ghurab, Gaphari, Alshamy, Othman and Suad (2021). Many criticisms of the KDD Cup 1999 for example are related the fact that no validation was ever performed to show that the dataset is actually similar to real network traffic Wang, Yang, Jing and Jin (2014). Modern attacks look different from attacks in the early ’90s. Realistic network traffics remain an open challenge to improve the development of network IDS.

Future work could address the development of a multi-class classification model to identify the specific attack type when malicious activity is detected. Additionally, considering recent datasets to build IDSs could offer insights into more recently developed attack types which do not currently exist in the two datasets, KDD Cup 99 and ISCX 2012, respectively published in 2009 and 2012.

References

Ahmad, I., Shah, S. A. A., & Al-Khasawneh, M. A. (2021). Performance analysis of intrusion detection systems for smartphone security enhancements. In *2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, 19–25. doi: 10.1109/ICSCEE50312.2021.9497904.

Badukale, P., Thorat, S., & Rojatkar, D. (2021). Sum up work on intrusion detection system in vehicular ad-hoc networks. *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*. 641–645. doi: 10.1109/ICOEI51242.2021.9452961.

Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L. (2019). A survey of deep learning methods for cyber security. *Information*, *10*(4), 122.

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2013). Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys and Tutorials*, *16*(1), 303–336.

Costa, K. A. D., Papa, J. P., Lisboa, C. O., Munoz, R., & Albuquerque, V. H. C. D. (2019). Internet of things: A survey on machine learning-based intrusion detection approaches. *Computer Networks*, *151*, 147–157.

Ghurab, M., Gaphari, G., Alshamy, R., & Othman, S. M. (2021). A detailed analysis of benchmark datasets for network intrusion detection system. *Asian Journal of Research in Computer Science*, *7*(4), 14–33.

Hasselt, H. V., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Hussain, F., Hussain, R., Hassan, S. A., & Hossain, E. (2020). Machine learning in IoT security: Current solutions and future challenges. *IEEE Communications Surveys and Tutorials*, *22*(3), 1686-1721. [9060970]. doi: 10.1109/COMST.2020.2986444.

Injadat, M., Salo, F., Nassif, A. B., Essex, A., & Shami, A. (2018). Bayesian optimization with machine learning algorithms towards anomaly detection. *IEEE Global Communications Conference (GLOBECOM)*. 1–6.

Janisch, J., Pevný, T., & Lisý, V. (2019). Classification with costly features using deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 3959–3966.

Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS)*. 21–26.

Kim, J., Kim, J., Kim, H., Shim, M. and Choi, E. (2020), "CNN-based network intrusion detection against Denial-of-service attacks", *Electronics*, *9*(6), 916. doi: 10.3390/electronics9060916.

Lansky, J., Ali, S., Mohammadi, M., Majeed, M. K., Karim, S. H. T., Rashidi, S., . . . & Rahmani, A. M. (2021). Deep learning-based intrusion detection systems: A systematic review. *IEEE Access*, *9*, 101574–101599. doi: 10.1109/ACCESS.2021.3097247.

Liu, Q., Hagenmeyer, V., & Keller, H. B. (2021). A review of rule learning-based intrusion detection systems and their prospects in smart grids. *IEEE Access*, *9*, 57542–57564. doi: 10.1109/ACCESS.2021.3071263.

Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., & Lloret, J. (2019). Shallow neural network with kernel approximation for prediction problems in highly demanding data networks. *Expert Systems with Applications*, *124*, 196–208.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Munos, R., Stepleton, T., Harutyunyan, A., & Bellemare, M. G. (2016). Safe and efficient off-policy reinforcement learning. *Proceedings of the 30th International Conference on Neural Information Processing Systems*, (1054–1062), Red Hook, New York, Curran Associates.

Nisioti, A., Mylonas, A., Yoo, P. D., & Katos, V. (2018). From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Communications Surveys and Tutorials*, *20*(4), 3369–3388.

Phadke, A., Kulkarni, M., Bhawalkar, P., & Bhattad, R. (2019). A review of machine learning methodologies for network intrusion detection. *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*. 272–275. doi: 10.1109/ICCMC.2019.8819748.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computers and Security*, *86*, 147–167.

Scarfone, K., & Mell, P. (2007). Guide to intrusion detection and prevention systems (IDPS). doi: 10.6028/NIST.SP.800-94.

Servin, A., & Kudenko, D. (2005). Multi-agent reinforcement learning for intrusion detection. In *Adaptive agents and multi-agent systems III. Adaptation and multi-agent learning*, 211–223.

Sewak, M. (2019). Deep Q network (DQN), double DQN, and dueling DQN.

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers and Security*, *31*(3), 357–374.

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(1), 41–50.

Sukhanov, A. V., Kovalev, S. M., & Stýskala, V. (2015). Advanced temporal-difference learning for intrusion detection. *IFAC-PapersOnLine*, *48*(4), 43–48.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed. 552). Cambridge, MA and London: The MIT Press.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications*. 1–6.

Tiwari, A. K., & Narain, B. (2021). IoT based intrusion detection system for criminal data analysis in deep learning. In *2021 IEEE International Conference on Technology, Research, and Innovation for Betterment of Society (TRIBES)*, 1–5. doi: 10.1109/TRIBES52498.2021.9751655.

Toldinas, J., Venčkauskas, A., Damaševičius, R., Grigaliūnas, Š., Morkevičius, N., & Baranauskas, E. (2021). A novel approach for network intrusion detection using multistage deep learning image recognition. *Electronics*, *10*(15). doi: 10.3390/electronics10151854.

Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, *36*(10), 11994–12000.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (1995). Dueling network architectures for deep reinforcement learning. *International Conference on Machine Learning*.

Wang, Y., Yang, K., Jing, X., & Jin, H. L. (2014). Problems of KDD cup 99 dataset existed and data preprocessing. *Applied Mechanics and Materials*, *667*, 218–243.

Wang, M., Cui, Y., Wang, X., Xiao, S., & Jiang, J. (2017). Machine learning for networking: Workflow, advances and opportunities. *IEEE Network*, *32*(2), 92–99.

Xu, X., & Luo, Y. (2007). A kernel-based reinforcement learning approach to dynamic behavior modeling of intrusion detection. In Liu, D., Fei, S., Hou, Z. G., Zhang, H., & Sun, C. (Eds), *Advances in neural networks*. 455–464. Springer Berlin Heidelberg.

Yang, Z., Liu, X., Li, T., Wu, D., Wang, J., Zhao, Y., & Han, H. (2022). A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers and Security*, *116*, 102675. doi: 10.1016/j.cose.2022.102675.

Corresponding author
Youakim Badr can be contacted at: yzb61@psu.edu