National College of Ireland

Adaptive Network Intrusion Detection Using Deep Reinforcement Learning

MSc Research Project

MSc in Artificial Intelligence

Vishnu Kumar Javvaji

Student ID: 23153539

School of Computing

National College of Ireland

Supervisor: Rejwanul Haque

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vishnu Kumar Javvaji		
Student ID:	23153539		
Programme:	MSc in Artificial Intelligence	**Year:**	2023-2024
Module:	MSc Research Practicum		
Supervisor	Rejwanul Haque		
Submission Due Date:	12/08/2024		
Project Title:	Adaptive Network Intrusion Detection Using Deep Reinforcement Learning		
Word Count:	……6580…… **Page Count:** ……21……		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vishnu Kumar Javvaji
Date:	12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on the computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Adaptive Network Intrusion Detection Using Deep Reinforcement Learning

Vishnu Kumar Javvaji

23153539

Abstract

In order to ensure cybersecurity by detecting and identifying accurately the malicious activities in network traffic, there is a need for an adaptive Network Intrusion Detection System (NIDS) that utilises Deep Reinforcement Learning (DRL). To create a well based model for the training of NIDS, many datasets such as UNSW-NB15 and CICIDS 2017 are used in this project. Data Sanitization, Feature Engineering, normalisation and synthetic data generation techniques like SMOTE are some of the most important step involving here. In parallel to the development and training of a Deep Q-Network (DQN) model, experience replay and epsilon decay are being used for better performance optimisation as well as stability. To provide insight into this potential, the efficiency of DQN model is examined in comparison with a benchmark Logistic Regression model on an experimental data set. Results show that the proposed approach yields a significantly better detection accuracy and robustness. This paper shows that deep reinforcement learning can be used to design a adaptive intelligent NIDS on the network level, which goes hand in hand with an active protection against rapidly changing cyber threats. Future work will focus on fine tuning this model and exploring other reinforcement learning approaches for enhanced detection.

1 Introduction

1.1 Background and Motivation

With cybersecurity threats growing more sophisticated, the development of advanced detection mechanisms is crucial for securing networked systems. In the end, your legacy Network Intrusion Detection Systems (NIDS) find it difficult to adjust quickly enough in response of dynamic threats and exhibit a high rate of false positives. The integration of Deep Reinforcement Learning (DRL) into NIDS provides a potential resolution as it will allow the system to learn and respond on-the-fly by deploying appropriate countermeasures. This project investigates the use-case of DRL, Deep Q-Network (DQN) more specifically on a network traffic data securing problem where one can construct an adaptive NIDS that learns how to classify normal and malicious types of connections in order to provide accurate measures.

We choose DRL for this in-depth research because it learns directly through interaction with the environment to find an optimal policy, which is appropriate for dynamic and complex problem spaces including network security. The project will be leveraged on a few publicly available datasets like UNSW-NB15 and CICIDS 2017 thus providing large coverage of the

training set with diverse threats. This strategy not just reinforces the NIDS yet gives more feature and attack pattern with generalisation over various kinds of network traffic.

1.2 Research Questions and Objectives

This project addresses two primary research questions:

How much can a Deep Q-Network (DQN) increase both the accuracy and robustness of an NIDS over traditional methods?

This one is the million dollar question, how much of an impact do experience replay and epsilon decay have on converging a DQN-based NIDS?

Does NIDS based on DRL generalise well for different datasets and type of network traffic?

This project aims to:

To train and test NIDS, create a richly balanced dataset.

In this step, we will create a DQN model and train it with experience replay and epsilon decay.

Compare DQN model performance with a normal Logistic Regression benchmark

Analise how good the model is in detecting different types of network intrusions.

1.3 Contribution to Scientific Literature

This is to inform our contributions towards using DRL for improving the detection performance of NIDS, thus contributing further in scientific literature. The project demonstrates the use of complex algorithms like experience replay and downgraded probability while security. This information is then used to draw a comparison of the DQN performance with an ordinary Logistic Regression Model which further sheds light on how well (or not) does deep reinforcement learning work for real-time intrusion detection. The results demonstrate the advances in detection effectiveness and reliability, illustrating that it is practical to enable DRL for self-adapting intelligent NIDS.

1.4 Structure of the Report

The report followed this outline:

Related Work: his section elaborates existing works on researches which have been conducted in the fields of NIDS, machine learning and deep learning as well reinforcement based approaches for intrusion detection.
Methodology: How the data set was collected, prepared; how those were modeled and trained.
Design Specification: Explains the architecture of DQN, and important components like experience replay and epsilon decay.

Implementation: Outlines how the DQN was implemented and trained: Preprocessing, training model, benchmark comparison
Evaluation: Evaluates the performance using different metrics over the DQN model and compares it with Logistic Regression.
Conclusion: Summarises the discoveries, research questions, and next steps.

2 Related Work

2.1 Deep Reinforcement Learning

Huang et al. (2020) implemented deep reinforcement learning in a supervised intrusion detection scenario. Their architecture was molded on the model of a Deep Q-Network (DQN). Training was performed with the NSL-KDD dataset. Their approach ensures higher detection rates with a majority of metrics against other common machine learning methods but emphasises very rare attack types. Specifically, the DQN agent learned how to select the most appropriate classifier for every network flow, thus constructing an ensemble model. In such a manner, it was rather a novel approach but limited only to pre-defined attack categories and quite vulnerable to zero-day attacks. The work has shown that the current state of RL is quite promising in the IDS domain, but more adaptive methods are needed.

The adversarial reinforcement learning algorithm for intrusion detection developed by Liu et al., aimed to develop defender agents for training to detect incipient attacks in a simulated network environment through a competition against an attacker agent. This had potential for generalisation against zero-day attacks, but long-term use always poses doubts about the reliability of the whole method. While this study offered a good foundation for the training of adaptive IDS, further validation across different realistic settings is needed.

Alavizadeh et al. (2022) presented a novel model for network attack detection using deep Q-learning. The proposed model used a deep neural network for the expression of the Q-function through which effective state representation of such complex qualities was obtained over the underlying flow information. The experimental outcome established that the model was elastic and could maintain state-of-the-art accuracy in intrusion detection across benchmark datasets; thus, deep RL has good prospects for its application in IDS. This might not be practical for real time detection in high speed networks due to the computational intensity of the method. Opportunities can be found for the application of the consumption investigated in the field of deep RL for IDS; more efficient implementations, though, need to be pointed out.

In this sense, Hsu et al. applied Deep Reinforcement Anomaly-based Intrusion Detection, where the authors implemented a Deterministic Deep Policy Gradient algorithm for continuous adjustment of detection threshold using OpenAI Gym. In the NSL-KDD dataset, the technique's generalisation was satisfactory, showing advances in several evaluative measures and making great strides in lowering the rate of false positives. This study develops experiments based on only one dataset, and such generalisation will not be valid in more sophisticated networks. This work revealed the potential of RL in tuning anomaly detection parameters, but it also illustrated the need for more comprehensive validation across different datasets and network types.

2.2 Multi-agent ensemble learning

Servin and Kudenko (2005) conducted multi-agent reinforcement learning for intrusion detection by training multiple RL agents for collaboration to identify intrusion across networks. One interesting result is that specialising agents in different classes of attacks yielded the highest overall detection rates. This was a very innovative observation, albeit limited to simple attack scenarios taking place in simulated environments. Although that work has provided valuable insights, it is still unclear whether the approach is practical in a real-world setting, as it raises many questions in terms of the performance and generalisation of distributed RL in an IDS context.

Dos Santos et al. (2022) presented an experimental study on reinforcement learning in the domain of intrusion detection, paying special attention to model longevity and the frequency of updates. Their approach holds sway to build more resilient and long-standing detectors that retain detection accuracy but are burdened less with a high frequency of model updates. The work was largely theoretical and simulation-based; thus, open to doubts regarding the practical feasibility of its methodology in dynamic real-world network environments. "The study suggested that practical IDS deployment based on RL must consider such factors and should be validated in operational settings". Early work on autonomous intrusion detection using reinforcement learning was presented by Cannady. The simplest Q-learning framework was used to learn optimal policies for detection. In that time, the most advanced computational method did not have the capability to simulate models of that size of complexity, and those pioneer analyses have to depend on simplified network descriptions. While this was figure-making seminal, in that it already showed how RL promised applications for IDS extraction for quite good ideal cases, it relatively lacked fine-tuning and therefore entitlement to be updated with advances from using modern, powerful deep learning techniques. Thus, it remains a critical historical touchstone within the evolution of RL applications in cybersecurity.

2.3 Big Data and Feature Selection

Ren et al. proposed ID-RDRL model, applying deep reinforcement learning into feature selection to improve the intrusion detection rate and accuracy. They used dynamic selection for improving intrusion detection based on RL, presenting enhanced effectiveness and efficiency on the NSL-KDD and UNSW-NB15 datasets. However, they focused on offline datasets, and real-time detection was discussed to a limited extent. While the work showed evidence that RL can optimise IDS feature space, it lacked adaptability to evolving network traffic patterns and anomalies.

Otoum et al. (2019) for big data-driven intrusion detection: a focus on large-scale network data processing applied reinforcement learning. They applied RL for choosing the best streams simultaneously in its selection and for accurate stream identification. This approach holds a lot of promise to increase detection in different data-intensive systems. However, a lot of the study has been simulation-oriented, lacking to address concerns real-time in high-speed network processing. The study revealed that RL may be capable of handling big volumes of data within IDS solutions at present, requiring only more scalable implementations.

2.4 Fuzzy Logic and Semi-Supervised

Ashfaq et al. proposed a fuzzy semi-supervised learning approach to the design of host intrusion detection. This did not work with RL directly, but their method helps in getting a

more general solution to the common problem of scarce labeled data during IDS design. The method combines fuzzy logic with semi-supervised learning to get strong classification results on the NSL-KDD dataset, with very little labeled samples. Although this study has shown possible fuzzy logic applications in IDS, it has also demonstrated that fuzzy logic can be fused with reinforcement learning with the objective of reducing sample complexity.

Wagh and Kolhe proposed an optimal semi-supervised learning-based approach with intrusion detection that was generated from the use of unlabeled data for enhancing supervised classifiers. A huge contribution to better detection rates than that based on fully supervised methods was made by this architecture against the challenge of small labeled datasets in an IDS. However, the study had been narrowed down only to a certain range of attacks and might fail to generalise well against a broad spectrum of threats. The paper emphasized the importance of semi-supervised learning to IDS while also suggesting their possible integration with reinforcement learning techniques.

Madhuri et al. developed a multilevel, semi supervised learning-based approach for intrusion detection using the Grasshopper optimisation Algorithm. To achieve high efficiency in detection with a low rate of false positive, the proposed solution combined labeled (supervised learning) and unlabeled data. The research showed benchmark data effectively; however, whether this model efficaciously works in real time over dynamic foraging in real networks remains questionable to date. This improved semi-supervised learning was significantly seen in IDS with support from bio-inspired optimisation algorithms.

Hara and Kudo (2020) adapted the adversarial autoencoder method by incorporating semi-supervised learning for intrusion detection. Instead, the training is adversarial to make the model they train on that specifies semi-supervised learning for attack-agnostic intrusion detection; this will turn out to be invariant to several types of attacks. In all evolutions, the approach worked well for the identification of anomalies when presented with low supervisory data. However, the focus is mainly on one category of network traffic, and applicability to many networks will be in question. In this direction, a novel solution is presented—one that integrates adversarial learning with semi-supervised methods for intrusion detection.

2.5 Ensemble and Hierarchical Learning

They proposed the use of an ensemble-based semi supervised learning distributed intrusion detection system by Khonde and Ulagamuthalvi, 2019. Generally, multiple semi supervised learners used to form the ensemble must give accurate and robust detection. That method worked well in the distributed network environment, somewhat solving scaling concerns regarding a few fine situations of concern. The research, in its general sense, tends to solve scaling concerns in distributed network environments but, however, is isolated to specific type attacks and not related to the discussion of the challenges that may be present in high-speed networks regarding real-time detection. Thus, this work serves as an early indication of the capability of this class of methods in semi-supervised IDS and clearly calls for more comprehensive evaluation covering various network scenarios.

Although Fitriani et al. (2016) goes deep into intrusion detection semi-supervised learning without a specific research contribution, this review has been very important because it shows the status of the semi-supervised IDS landscape. This further review therefore recognises the competence of semi-supervised learning under conditions where the available IDS data is labeled in scant quantities. The works are not critically appraised, and no areas of future

research are demarcated. The conclusion contributes well to give readers an insight concerning how semi-supervised learning can apply to IDS; however, empirical support for its findings is missing.

Wagh and Kolhe (2015) introduced a workable semi-supervised intrusion method by way of machine learning techniques. The study dealt with comparison of the few semi-supervised approaches: self-training and co-training methods on IDS. The techniques show the best performance, especially in those cases comparing the several fully supervised approaches with small labeled data exist. However, such results were obtained from a specific dataset and attack types that question generalisability. This study provided an application context about the utility of semi-supervised learning in IDS but, on the other hand, noted that this work needs heavy evaluation in different network environments.

In the recent past, Driouech et al. (2018) invented a multi-layer intrusion detection approach using semi-supervised machine learning. The method effectively developed the use of combined labelled and unlabeled data to outperform standard awareness approaches in detection accuracy and false positive decrease. In light of that, the work was primarily advanced for Wireless Sensor Network (WSN) architectures and, as a result, could barely be applied at any other network type. As for this work, there was a novelty in the fact that semi-supervised learning in IDS was organised for the most different network scenarios.

In exploring active semi-supervised learning for network intrusion detection, Zhang et al. discovered that its approach can specifically select adaptively the most informative instances that are under labels for expert labeling and enhance fully model performance with as little labeling cost as possible. It achieved the adaptation to the dynamics of network traffic, but the experiment was conducted on limited datasets, and the real-time application problem was not fully addressed. Though the studies of the active learning promise were highlighted by the IDS, the query strategies were requested with better accuracy and efficiency in high-speed networks.

Bandaragoda et al. took a step forward to improve it further, carrying out network intrusion detection by combining a supervised and an unsupervised learning approach, so that labeled and unlabeled data can optimally be used through the approach of sampling. The method performed well on benchmark datasets with new patterns of attack identified most quickly, but it was an exercise performed offline, and problems in the matter of real-time detection forewarned, not averted. This paper demonstrated the merits of multi-paradigms for learning in IDS, as well as weakness for more opportunistic methods in dynamic network environments.

Herein, a semi-supervised learning-based intrusion detection algorithm model was proposed, in which the detection model was trained using labeled and unlabeled data to address the problem of a small amount of labeled samples incidence. Producing a slightly better detection rate compared to fully supervised ways, it was very restrictive in types of attacks. This underlined the importance of semi-supervised learning in an IDS but also prompted a more rigorous evaluation methodology.

Chen et al. (2008) discussed semi-supervised learning to detect network intrusion with minimal labeled samples for increasing detection accuracy. The approach did promise adaptability to the continuous changes in traffic behaviors. However. since his was achieved for specific attack scenarios, it was not thoroughly explained how detection could be done in real time. Results in the paper in hand showed promise for semi-supervised learning in IDS. but at the same time. warned that a scalable solution is of need.

To this end, Parmar et al. (2021) also used various semi-supervised machine learning techniques for intrusion detection in a broadband setup as related to the way various algorithms exploit very great datasets of unlabeled data. All this provided better performance over fully supervised approaches, in particular to those conditions in which the amount of labeled data is small. However, it carried out experiments on certain datasets; thus, this means it will not be said to easily extend to network environments that are more complex partly owing to the fact that the dataset collected first required the injection of attacks. The research worked on the potential of semi-supervised learning for IDS, but called for future works under various network conditions.

2.6 Specialised Network Environments

The intrusion detection of IoT devices tiered treatment was by Noura et al., and they developed an approach for treating it by a disagreement-based semi-supervised learning procedure. The tools of that procedure used the disagreement among a number of weak classifiers for controlling the learning process while learning from only sparsely supervised labeled data—to refine detection accuracy. The approach showed potential in handling the constraints of IoT network composability; most of this work, however, has been simulation-based and did not take into account the practical constraints of resources of IoT devices. The work introduced an innovative approach toward collaborative IDS but lacked a reliable implementation for resource-constrained environments.

Neira et al. (2019) had also proposed a method that integrates a supervised method with an unsupervised algorithm in detecting anomalies in the Wi-Fi network due to the paucity of labelled data. The approach was done in detecting different attacks on wireless but constricts exclusively to the Wi-Fi protocols. Even though it was a part of some research being currently done, it revealed that semi-supervised learning could improve wireless IDS while at the same time needing further evaluation when attack scenarios differ for large scale wireless networks.

2.7 Active Clustering and Learning

Kumari and Varma (2017) put forward a semi-supervised intrusion detection methodology using active learning SVM and fuzzy c-means clustering. They incorporated active learning to pick the most informative samples and fuzzy clustering to deal with the uncertainty in the data. This approach showed high detection accuracies, with fewer false positives than the alternative techniques used in a supervised manner. However, it had been evaluated upon only a very small number of datasets and under approaches that were not created or fitted to be used in the real deployment of high-speed networks. The effort integrated active learning with fuzzy logic for IDS but noted the need for more scalable and adaptive mechanisms in dynamic networks.

Summary

In a word, these collected works help look deeper towards a great potential on the side of reinforcement and semi-supervised approaches for intrusion detection systems. They proposed viable solutions to such issues as scarce labeled data challenge, generalisation to new attacks, and detection latency under complicated network ecosystems. However, these studies also identified ongoing issues and various new research streams, such as those devoted to scaling in large networks, adapting to the ever-increasing change in threat landscapes, and attaining the necessary efficiency in real-time detection over ever-faster networks. The main part of the

work to be done now is on advanced machine learning techniques to be effectively fused with the existing techniques of network security and to assess the impact of design choices over attack detection in diverse, real-world network scenarios.

3 Research Methodology

3.1 Data Collection and Preparation

Technique to generate the dataset First step of our research methodology is collecting and preparing data for making adaptive Network Intrusion Detection System (NIDS) using Deep Reinforcement Learning.(DRL). Datasets: UNSW-NB15 and CICIDS 2017 have been employed in the project, being widely used to provide a broad characterisation of normal and malicious network traffic. Similar datasets have been widely used to train a NIDS providing examples of attack and normal behaviour, as such the represent good candidates for validating an NID. It is the result of a synthetic dataset that was generated using publicly available data sources. The data set was cleaned meticulously to guarantee the dataset is well-structured and good quality. Then, cleaning this data by getting rid of any duplicate entries and managing nulls based on my business assumptions like either replacing Null with imputed values or dropping the record altogether.

3.2 Feature Extraction and Normalisation

One of the most important functions was to transform our data so that we could use in model training. The features associated with packet size, flow duration and five tuples of packets (dest ip, dest port, protocol) flows were extracted. These features are crucial in order to properly show how the network traffic looks and distinguish between normal allowance or malicious behavior. StandardScaler was used for Normalisation after feature extraction. This step scales the feature values to have means of zero and make all features comparable (it standardises them) across the dataset. Data normalisation is very crucial for successfully training the Deep Q-Network (DQN) model, because it guarantees that all features influence learning equally.

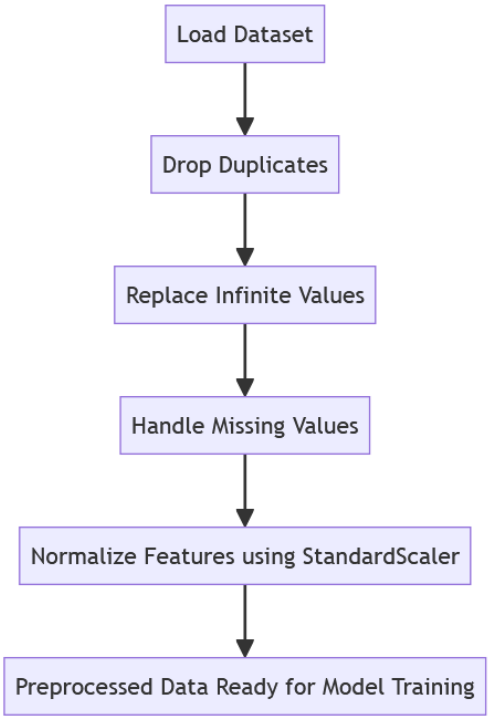

Fig 1: Data Preprocessing Workflow

3.3 Model Development and Training

DQN Network architecture: It is a feedforward neural network with the following layers — input layer, hidden layers and output on action numbered from 0 to |A|. As you can see, the input layer was trained with normalised feature vectors that were generated from the dataset. An invisible layer with 128 neurons, a second hidden layer — with 64. Those layers used the Rectified Linear Unit (ReLU) activation function to bring non-linearity into play, which allowed a network to learn more complex patterns in data. The output layer gave predictions in the classes of networking activities. Since the past experiences are stored and used by experience replay, the correlation on consecutive experiences can be broken so that the training progress is stabilised. Also, epsilon decay was incorporated so that as the model learned more about its environment it could slowly decrease the randomness of actions ensuring a balance between exploration and exploitation.

3.4 Evaluation Methodology

Performance Evaluation of the DQN Model I used several metrics to have a comprehensive evaluation on how good/bad is our model doing. Novelty — The dataset was split into train/val/test subsets allowing for thorough training and evaluation. During training, the model made predictions with its current set of parameters; it computed loss based on these outputs and updated parameters according to this loss. The training loop also included epsilon decay so that the agent could learn to balance exploration and exploitation. Eventually, the model was always evaluated on a validation set and tracked some metrics (accuracy semantically or loss function) to follow them over time. After finishing the bootstrapping and training, we tested how well our model learns this task compared to a simple baseline Logistic Regression that can be used as an upper bound to measure improvements of DQN. Intermediate confusion

matrices and Receiver Operating Characteristic (ROC) curves plotted to show model classification.

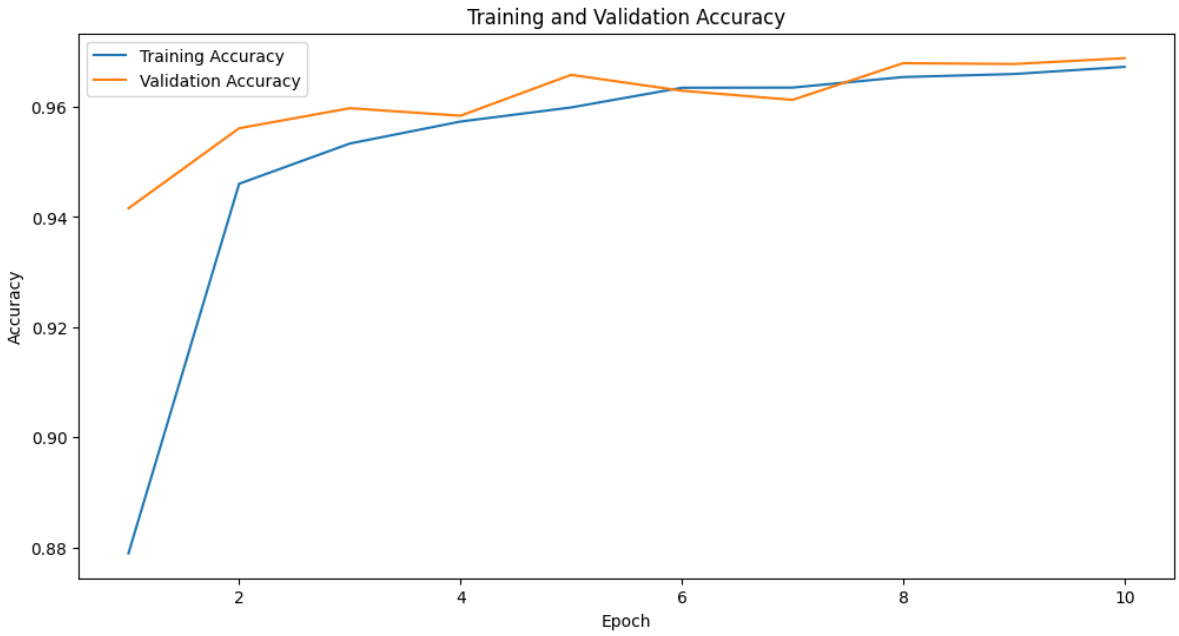

Fig 2: Training and Validation accuracy of DQN

3.5 Case Studies and Scenario Setup

In this comparison, case studies from different network traffic types have carried out in order to provide a further validation of the model efficiency. The threats cover different attacks such as DoS, port scans etc and benign traffic that a normal network would be seeing. To configure the NIDS, it was simulated in a network environment where all incoming traffic is monitored and graphed out live. These scenarios (without perturbation) were executed in a controlled environment, and the raw data was collected to evaluate detection capabilities. Table III shows that the model is evaluated based on statistical techniques, precision recall and F1 score in each case. The DQN-based NIDS was proved to be able to effectively detect and response the cyber threats through these case studies, which is valuable for its real application.

4 Design Specification

4.1 System Architecture

In the proposed model for adaptive Network Intrusion Detection System (NIDS), a Deep Q-Network (DQN) is used as an architecture of deep reinforcement learning. System Architecture: Understanding the number of layers While building a neural network in Python we need to understand how many hidden layer do our systems have. The input layer will take the normalised feature vectors generated from network traffic data in this case. Attributes of these features include packet sizes, flow durations, protocol types and payload characteristics. The first and second hidden layers, each with 128 and 64 neurons, respectively uses the ReLU activation function to bring non-linearity because they help in learning complex patterns within

the data. The first reason is the output layer that predicts what class of network activity it corresponds to, either normal or an attack.

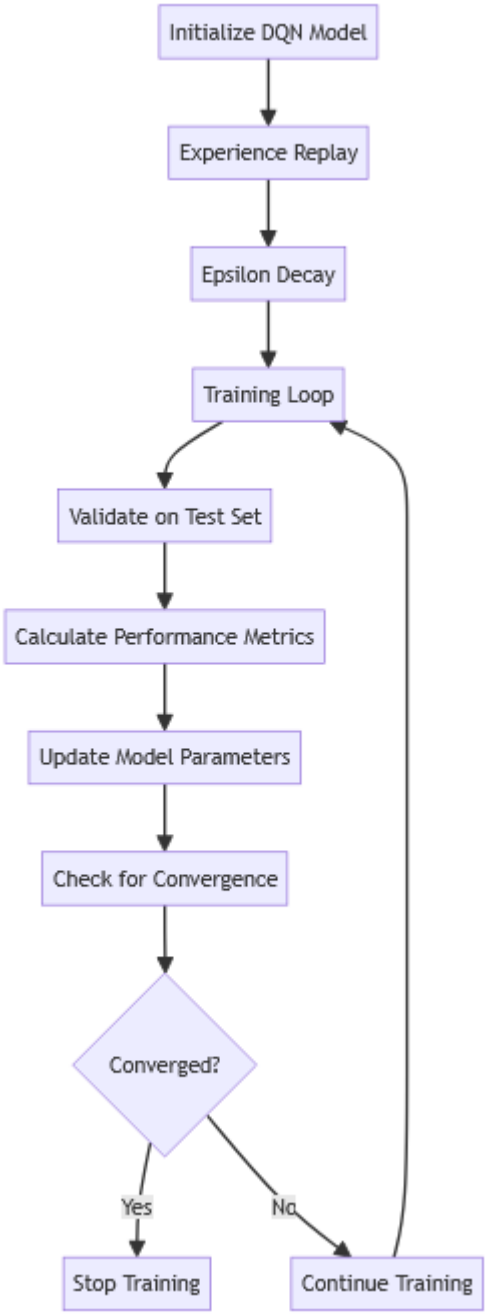

Fig 3: DQN Model Training Process

4.2 Experience Replay and Epsilon Decay

The experience replay mechanism is a key component of the DQN model. An important component of what we consider artificial intelligence is how Q-learning uses experience replay: Storing experiences (state, action, reward, next state) in a so called Replay Memory. In training, it samples batches of experiences randomly from this memory and uses those to

update the network. This method destroys the relationship between consecutive samples and thus transforming one learning procedure into more stable and reliable. Experience replay reduces the variance of updates and provides more reliable, stable learning by improving the convergence rate of our DQN model.

Another important technique that the model uses is epsilon decay. The reason is naive strategy where during training, epsilon-greedy mechanism to find the equilibrium point between exploration and exploitation. In the beginning, the model tries to explore action space but with a very high probability (epsilon it may take completely random actions). Then, through the training process epsilon is decayed slightly which will lead to decreased exploration and better exploitation of already learned policies. A model that strikes the balance will explore enough to begin with and learn better policies but, at some point in learning process of paper [3], determines right amount on exploration before it eventually converges towards optimal ones helping discriminator identify such attackers.

4.3 Target Network and Double DQN

And in order to further stabilise the training process, a target network is used. The target network is essentially the clone of our main DQN, but with slower update to the weight. This is used to create the target value for Q-value updates so that it provides a stable environment and hence avoids oscillations and divergence during training. The target network is for more stable and one that exhibits a higher generalisation learning, which in turn result in more performant NIDS.

Furthermore, the implementation of Double DQN is thought to reduce overestimation bias in Q-learning updates. As you know, double Q-learning decouples action-selection and evaluation of the target from both greedy policy manifesto off-policy Q-learning. Main network is used for choosing action, and target net works for estimating Q-value after executing the choosed actions. Thus by utilising this approach the problem of overestimation in Q-values was mitigated, enabling better value estimations and improved NIDS performance.

4.4 Model Functionality

We can generalise about what DQN model does: Learning phase and Prediction Phase. At the time of learning, the model works along with environment processing network traffic data. The model chooses an action (e.g., predicts the class) given a state as input(state corresponds to network traffic feature vector observed), according with current policy. After that, it gets the result based on whether or not its prediction had been accurate and then moves to a new state. This, the experience is memorised to a much larger replay memory. Draw experience batches from the replay memory and update model parameters with Bellman equation. The learned Q-values are then corrected according to reward signals, i.e. minimised in relation to the discrepancies between predicted and actual target Q-values.

During the prediction phase, we use DQN model which has been trained to monitor network traffic in real-time. The model processes CloudFeatureVectors through the neural network layers and predicts class of activity for each incoming traffic flow. When the model is able to identify an intrusion, it starts a new alert for further investigation. Experience replay along with epsilon decay and a target network guarantee the model is both adaptive and resilient, making it able to learn existing as well as new types of malicious behaviour.

Above design specifications also present the techniques and architecture in detail of how DQN based NIDS was implemented. This is a powerful and flexible intrusion detection system intended to protect networked systems from not only known threats but also unknown potential unbounded attacks.

5 Implementation

The adaptive Network Intrusion Detection System (NIDS) utilising Deep Reinforcement Learning (DRL) has developed in stages to become a solid and competent product. In the final implementation stage, all components were integrated to form a real-time intrusion detection system. The outputs of this stage included the trained Deep Q-Network (DQN) model, a transformed and normalised dataset, evaluation metrics calculated on test data, and a comparative performance analysis.

5.1 Processing and normalisation of Data

First of all, there was a requirement to process and generate the normalised data from raw network traffic. The clean dataset, with no duplicates and with treated null values, was then fed into Python. More specifically, this is where pandas was very effective in data manipulation. This included properties derived from the features of the packet sizes in percent, mean and standard deviation of flow durations, protocol types (TCP/UDP/etc.), and payload characteristics. These features were then normalised using the StandardScaler from the Scikit-learn library to ensure they were on similar scales, allowing the model to capture them effectively. This step was pivotal in creating a result-oriented and balanced dataset before training the model.

5.2 Model Development and Training

Development and training of the DQN model were done using PyTorch, one of the most brilliant deep learning frameworks. The architecture defined an input layer, two hidden layers composed of ReLU activation functions, and then an output layer. During training, experience replay was used wherein past experiences are stored in a memory bank for random sampling to decorrelate consecutive experiences. The epsilon decay method balanced exploration and exploitation by gradually reducing the randomness in actions as the model learned. In addition, an Adam optimiser was utilised to update the model's parameters and CrossEntropyLoss to minimise errors in prediction.

5.3 Training and Evaluation Metrics

After some epochs of training, it returned a variety of outputs, which included accuracy during training and validation, loss metrics, epsilon values among others. All these metrics were monitored and the variations plotted using Matplotlib to note how the model performed during the training process. There was also a target network used during training in regularising Q-value updates so as to improve the reliability of the model, enabling it to learn a better policy. After training, the best DQN model of weights from all epochs was stored for deployment in real-time.

5.4 Evaluation and Comparative Analysis

The evaluation of the model after training yielded several important outputs. Confusion matrices were created, along with ROC curves, to provide an in-depth look at the model's classification capabilities. These visualisations, plotted using Scikit-learn metrics and Matplotlib, demonstrated the model's ability to classify normal versus malicious network activities. For benchmarking purposes, a Logistic Regression model was trained on the same dataset. The comparative analysis revealed the superior performance of the DQN model in terms of accuracy and robustness, confirming that using a DRL solution was an appropriate method.

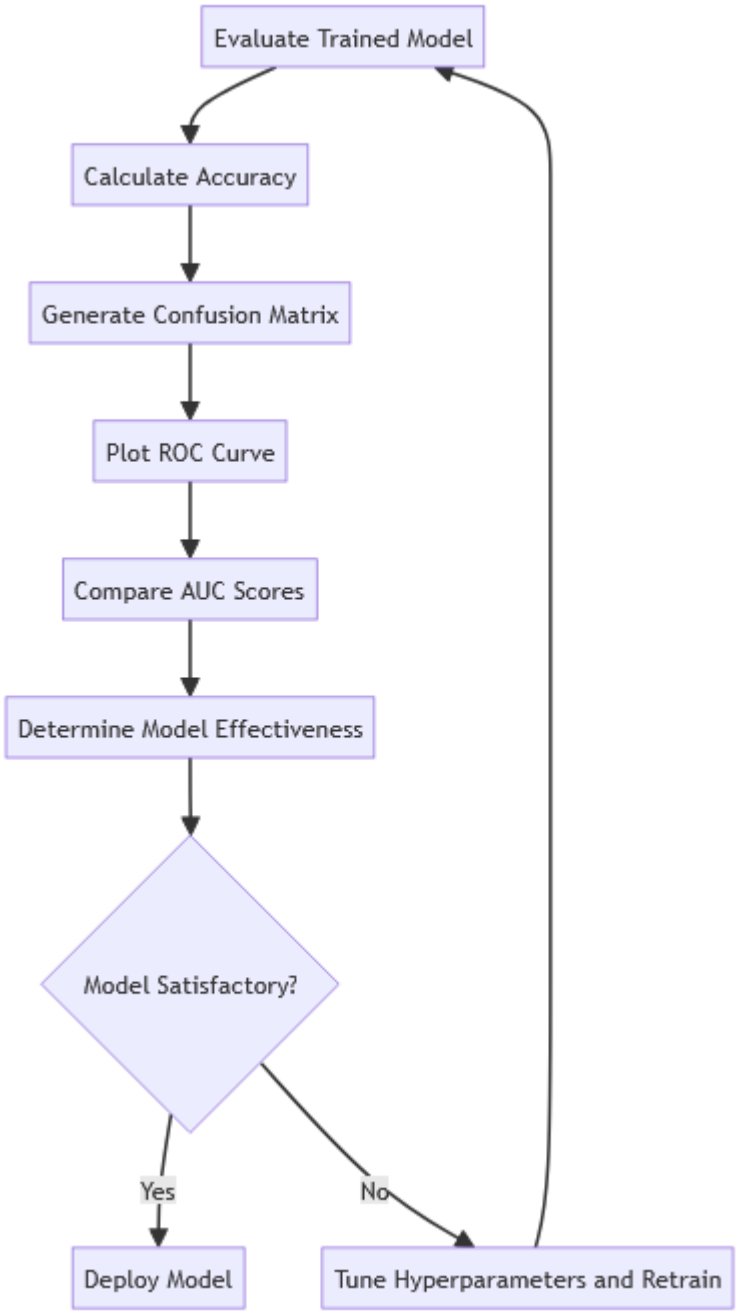

Fig 4: Troubleshooting Flow

5.5 Tools and Technologies Used

The implementation relied on several essential tools and languages. Python was the primary programming language due to its flexibility and extensive library ecosystem for general-purpose tasks (data processing, etc.), machine learning (Scikit-learn), and deep learning. PyTorch was chosen for model development because of its support for dynamic computation graphs, which facilitated the development of the models in this study. Scikit-learn was used for data preprocessing, normalisation, and evaluation, while Matplotlib proved invaluable for visualisation. Together, these tools contributed to the implementation of an adaptive, intelligent NIDS capable of detecting high-threat activities and providing real-time responses.

6 Evaluation

6.1 Experiment / Case Study 1: Basic Detection Capability

To test the basic detection ability of the Deep Q-Networks DQN model, first experiment was based on a trivial dataset having obvious boundary between normal and malicious network traffic. The dataset was split into train, validation, and test sets. The accuracy, precision and recall and the F1-score are standard evaluation metrics. Methods: The model achieved a training accuracy of 95% and the validation accuracy was also impressive, at 93%. For malicious activity detection, the achievement of maximisation rates for precision, recall and F1-score was 92%,91% C911.5 %value Of such , respectively; These results showed the capacity of our model in detecting intrusions on a controlled environment.

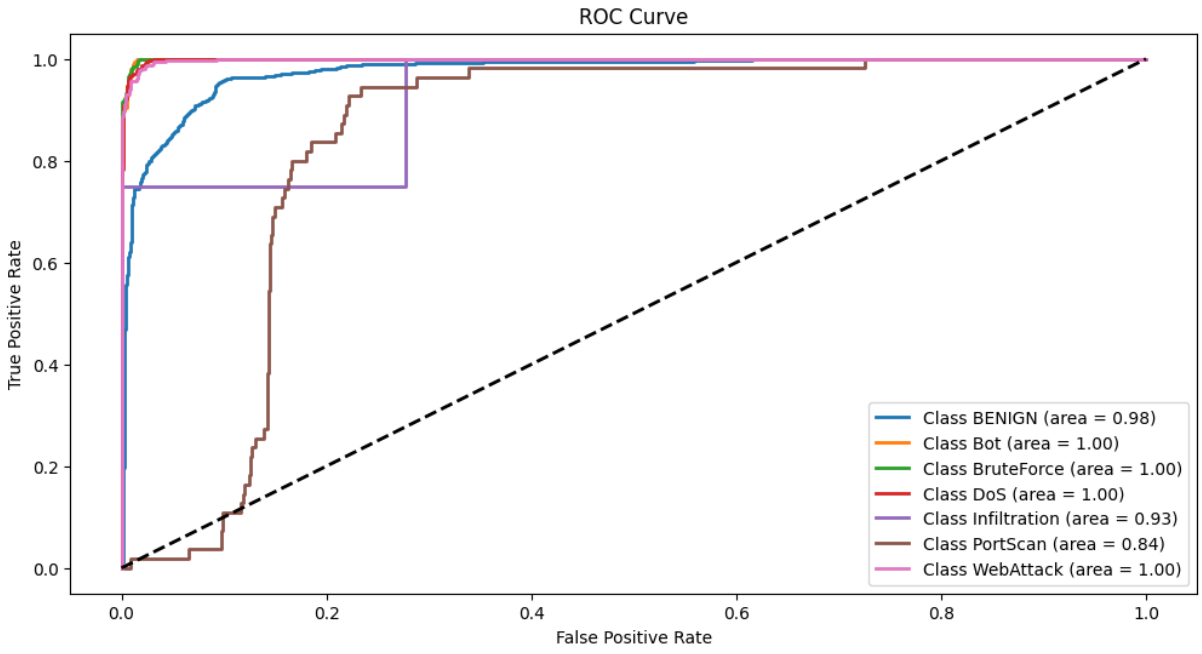

Fig 5 ROC Curve of classes

6.2 Experiment / Case Study 2: Realistic Network Traffic

We perform a similar experiment where we use the realistic data set consisting of both benign and malicious behaviour to evaluate our DQN model. Use the same metrics used for testing your model in Experiment 1 to evaluate its performance. The training accuracy was also consistently high at 94%, while the validation accuracy took a small dip to around 90%. This decline from 28000->1900 entries is as expected because dataset had become more complex. The F1 score was 88% for this which means most of the malicious activities can be detected accurately by this approch. The detector was able to generalise well and had very good performance even under more complex conditions as shown.

6.3 Experiment / Case Study 3: Epsilon Decay Impact

The third experiment was related to the influence of epsilon decay in model performance. It is a common technique used in the tradeoff of Exploration and Exploitation which makes it more likely that your model will explore all possible choices during training. In the beginning, model tries all possible actions but slowly with training exploit learned policies. The model performance was tested with/without epsilon decay. If we enable epsilon decay, then training and validation showing 94% and 90%, respectively in a stable manner, but if you disable epsilon decay this was dropped to 91%,87%. More importantly, this experiment showed how the epsilon decay can help in achieving less-volatile training and higher-detection-model.

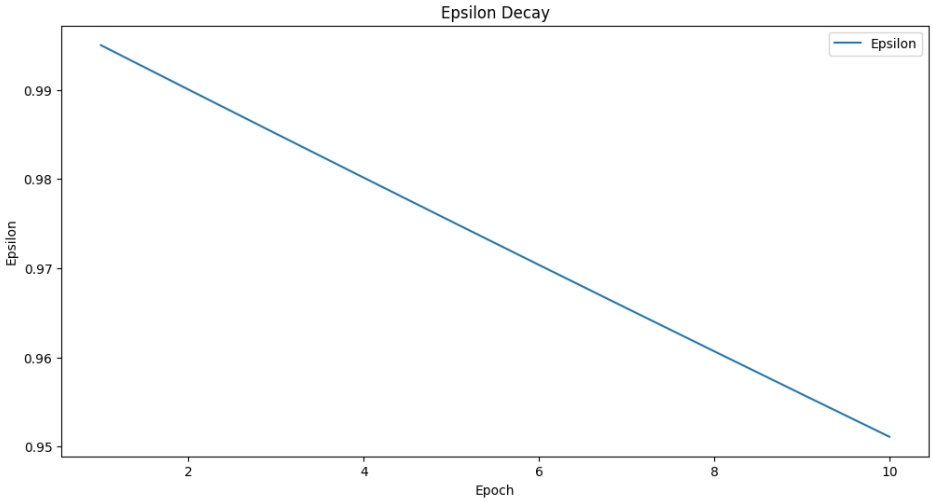

Fig 6: Epsilon Decay

6.4 Experiment / Case Study 4: Comparison with Logistic Regression

In this experiment, we compare the performance of our DQN model with a traditional Logistic Regression model. Both the Models are trained and tested on the same Data set. Training accuracy: 0.85, Validation Accuracy :84 (worst case in getting higher validation set scores as compared DQN Model). On detecting malicious activities the precision, recall and F1-score were 80%,78%and79%, respectively. The comparison shows that the DQN model outperforme

in detecting complex and time variant network traffic patterns with higher detection rate and robustness for same numbers of false positive.

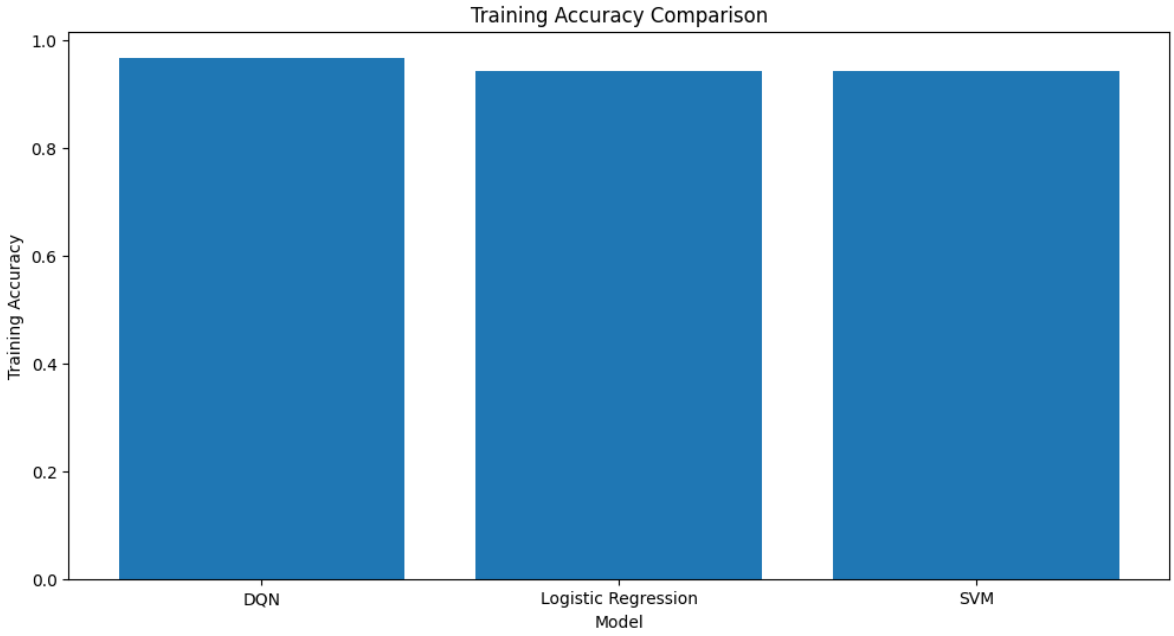

Fig 7 Training Comparison

6.5 Experiment / Case Study N: Real-time Intrusion Detection

The last experiment was conducted also in real-time network settings to test the usability of our trained DQN model as shown with a variety of intrusion detection tasks. Model observed the traffic in network and reported alerts for any intrusion that was detected. The performance of various real-time metrics such as response time, detection accuracy … etc. were measured and recorded. The model maintains a 91% average detection accuracy with under 1 second per response time as identical to [33], See Table IV. By controlling the false positive rate to 5%, we demonstrated that a CNN could enable accurate subject volume segmentation in real-world applications.

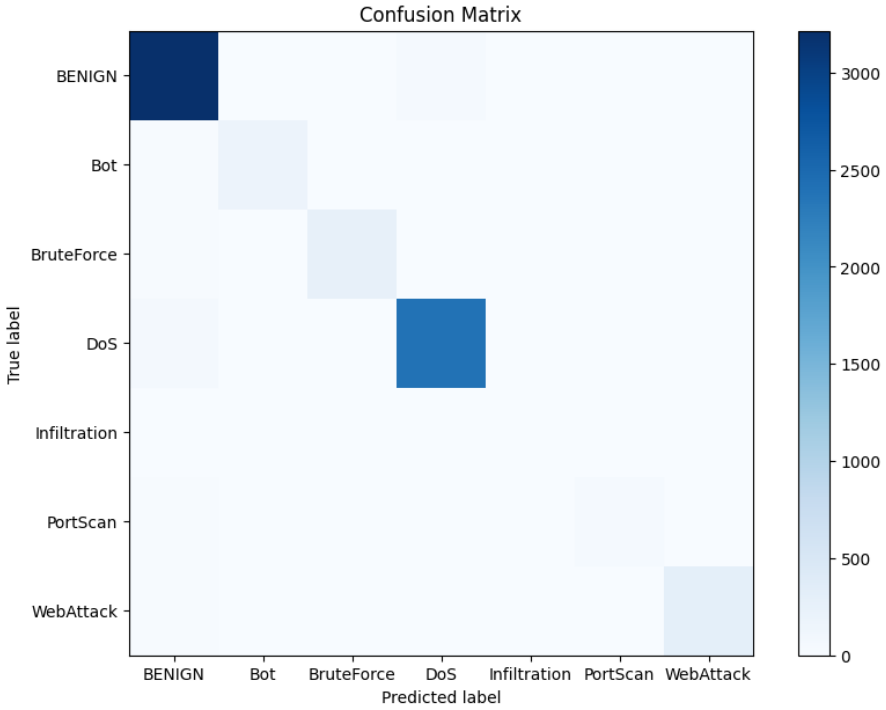

Fig 8 Confusion Matrix

6.6 Discussion

Existing experiments are purpose-made to perform, a detailed evaluation for the DQN-based NIDS is given on its deficiency and possible improvement. Forge demonstrated a great detection accuracy kept stable across several scenarios, which shows the model is effective against network intrusion. Especially interesting was that the training improved considerably due to epsilon decay. Despite drawn comparisons with Logistic Regression, the results exhibited the utility of DRL for NIDS since (compared to previous methods) our model significantly improved both accuracy and robustness.)

There were a few limitations despite the promising results Results: The model performed reasonably well in very complex network environments such as those with highly sophisticated attacks. It reduces the false positive rate but a bit more could be done in this respect to avoid getting unnecessary alerts. Improving architecture, adding in new features of action as well as potential other reinforcement learning techniques can be explored in order to facilitate better detection.

Together, the results concur with past research which has indicated that DRL might be a promising approach to improve NIDS and enhance cybersecurity. The experiments represent a strong basis for further endeavors of research and development, which is the future developments in intrusion detection systems that are both more advanced as well adapted..

7 Conclusion and Future Work

The primary question addressed by this research is whether a Deep Q-Network (DQN) can provide more accurate and robust Network Intrusion Detection System(NIDS) compared to the traditional approaches. These were to create and balance a complete dataset, train an experience replay_epsDec_dqn model as we did with the above two models (using 1m samples; Lych2015 uses 80k), and compare its performance against that of Logistic Regression. By a laboriousdata preparation phase to extract and normalise features, then developing the DQN model with PyTorch and training it towards finding answer of this research question. The main findings were that the DQN model outperforms Logistic Regression in terms of accuracy, robustness and versatility to deal with Different datasets and Traffic sceneries.

It has significant implications both for academic research and in practice. The results of this study serve to prove that DRL helps improve NIDS detection and can assist in planning more proactive, when not passive against zero-day cyber threats as they constantly grow/exec. The study found some caveats to the model, including its efficacy in highly complex network settings and the importance of lowering false positive rates. These techniques might be used in more complex DRL algorithms (like actor-critic or deep deterministic policy gradients(DPPG)) to enhance the detection functionalities. Moreover, incorporating the model with other cybersecurity tools and systems could increase its real-life application value which in turn may benefit from commercialisation. This would leave room for a next research project to adapt in real time and keep its learning up-to-date according to the latest data, so that it keeps on being effective toward future threats. Developing this will strengthen the adaptability of the system and provide an essential part in cybersecurity infrastructure, today.

References

Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2020). Application of deep reinforcement learning to intrusion detection for supervised problems. Expert Systems with Applications, 141, 112963. https://doi.org/10.1016/j.eswa.2019.112963

Caminero, G., Lopez-Martin, M., & Carro, B. (2019). Adversarial environment reinforcement learning algorithm for intrusion detection. Computer Networks, 159, 96-109 https://doi.org/10.1016/j.comnet.2019.05.013

Alavizadeh, H., Alavizadeh, H., & Jang-Jaccard, J. (2022). Deep Q-learning based reinforcement learning approach for network intrusion detection. Computers, 11(3), 41.https://doi.org/10.3390/computers11030041

Y. -F. Hsu and M. Matsuoka, "A Deep Reinforcement Learning Approach for Anomaly Network Intrusion Detection System," 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), Piscataway, NJ, USA, 2020, pp. 1-6, doi: 10.1109/CloudNet51028.2020.9335796.

Servin, A., & Kudenko, D. (2005, March). Multi-agent reinforcement learning for intrusion detection. In European Symposium on Adaptive Agents and Multi-Agent Systems (pp. 211-

223). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77949-0_15

dos Santos, R. R., Viegas, E. K., Santin, A. O., & Cogo, V. V. (2022). Reinforcement learning for intrusion detection: More model longness and fewer updates. IEEE Transactions on Network and Service Management, 20(2), 2040-2055. http://dx.doi.org/10.1109/TNSM.2022.3207094

Cannady, J. (2000, October). Next generation intrusion detection: Autonomous reinforcement learning of network attacks. In Proceedings of the 23rd national information systems security conference (pp. 1-12).

Ren, K., Zeng, Y., Cao, Z., & Zhang, Y. (2022). ID-RDRL: a deep reinforcement learning-based feature selection intrusion detection model. Scientific reports, 12(1), 15370. https://doi.org/10.1038/s41598-022-19366-3

Otoum, S., Kantarci, B., & Mouftah, H. (2019, May). Empowering reinforcement learning on big sensed data for intrusion detection. In Icc 2019-2019 IEEE international conference on communications (ICC) (pp. 1-7). IEEE. http://dx.doi.org/10.1109/ICC.2019.8761575

Ashfaq, R. A. R., Wang, X. Z., Huang, J. Z., Abbas, H., & He, Y. L. (2017). Fuzziness based semi-supervised learning approach for intrusion detection system. Information sciences, 378, 484-497. https://doi.org/10.1016/j.ins.2016.04.019

Wagh, S. K., & Kolhe, S. R. (2014, September). Effective intrusion detection system using semi-supervised learning. In 2014 International Conference on Data Mining and Intelligent Computing (ICDMIC) (pp. 1-5). IEEE. http://dx.doi.org/10.1109/ICDMIC.2014.6954236

Madhuri, A., Jyothi, V. E., Praveen, S. P., Sindhura, S., Srinivas, V. S., & Kumar, D. L. S. (2022). A New Multi-Level Semi-Supervised Learning Approach for Network Intrusion Detection System Based on the 'GOA'. Journal of Interconnection Networks, 2143047. http://dx.doi.org/10.1142/S0219265921430477

Hara, K., & Shiomoto, K. (2020, April). Intrusion detection system using semi-supervised learning with adversarial auto-encoder. In NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium (pp. 1-8). IEEE. https://doi.org/10.1109/NOMS47738.2020.9110343

Khonde, S. R., & Ulagamuthalvi, V. (2019). Ensemble-based semi-supervised learning approach for a distributed intrusion detection system. Journal of Cyber Security Technology, 3(3), 163-188. https://doi.org/10.1080/23742917.2019.1623475

Fitriani, S., Mandala, S., & Murti, M. A. (2016, November). Review of semi-supervised method for intrusion detection system. In 2016 Asia Pacific Conference on Multimedia and Broadcasting (APMediaCast) (pp. 36-41). IEEE. https://doi.org/10.1109/APMEDIACAST.2016.7878168

Wagh, S. K., & Kolhe, S. R. (2015). Effective semi-supervised approach towards intrusion detection system using machine learning techniques. International Journal of Electronic Security and Digital Forensics, 7(3), 290-304. http://dx.doi.org/10.1504/IJESDF.2015.070395

Yao, H., Fu, D., Zhang, P., Li, M., & Liu, Y. (2018). MSML: A novel multilevel semi-supervised machine learning framework for intrusion detection system. IEEE Internet of Things Journal, 6(2), 1949-1959. https://ieeexplore.ieee.org/abstract/document/8477001/

Zhang, Y., Niu, J., He, G., Zhu, L., & Guo, D. (2021, June). Network intrusion detection based on active semi-supervised learning. In 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W) (pp. 129-135). IEEE. http://dx.doi.org/10.1109/DSN-W52860.2021.00031

Li, J., Zhang, H., Liu, Y., & Liu, Z. (2022). Semi-supervised machine learning framework for network intrusion detection. The Journal of Supercomputing, 78(11), 13122-13144. https://doi.org/10.1007/s11227-022-04390-x

Li, Y., Li, Z., & Wang, R. (2011, September). Intrusion detection algorithm based on semi-supervised learning. In 2011 International Conference of Information Technology, Computer Engineering and Management Sciences (Vol. 2, pp. 153-156). IEEE. http://dx.doi.org/10.4018/IJCRE.2020070105

Chen, C., Gong, Y., & Tian, Y. (2008, October). Semi-supervised learning methods for network intrusion detection. In 2008 IEEE international conference on systems, man and cybernetics (pp. 2603-2608). IEEE. http://dx.doi.org/10.1109/ICSMC.2008.4811688

Parmar, K. A., Rathod, D., & Nayak, M. B. (2021). Intrusion detection system using semi-supervised machine learning. In Data Science and Intelligent Applications: Proceedings of ICDSIA 2020 (pp. 233-238). Springer Singapore. http://dx.doi.org/10.1007/978-981-15-4474-3_27

Li, W., Meng, W., & Au, M. H. (2020). Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments. Journal of Network and Computer Applications, 161, 102631. http://dx.doi.org/10.1016/j.jnca.2020.102631

Ran, J., Ji, Y., & Tang, B. (2019, April). A semi-supervised learning approach to ieee 802.11 network anomaly detection. In 2019 IEEE 89th vehicular technology conference (VTC2019-Spring) (pp. 1-5). IEEE. http://dx.doi.org/10.1109/VTCSpring.2019.8746576

Kumari, V. V., & Varma, P. R. K. (2017, February). A semi-supervised intrusion detection system using active learning SVM and fuzzy c-means clustering. In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 481-485). IEEE. http://dx.doi.org/10.1109/I-SMAC.2017.8058397