International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Intrusion Detection System using Deep Q-Network Approaches based on Reinforcement Learning

Ankit Chakrawarti *[1], Dr. Shiv Shakti Shrivastava[2]

Submitted: 17/11/2023 **Revised**: 29/12/2023 **Accepted**: 09/01/2024

Abstract: This study presents a comparative analysis of various algorithms for Intrusion Detection Systems (IDS), including KNN, RF, ANN, CNN, SVM, and a multi-method approach combining KNN, RF, NN, and NB. The proposed method, which integrates these techniques, achieves a notable accuracy of 96.8%. Additionally, the study explores a Deep Q-Networks (DQN) based IDS, detailing steps from data pre-processing and environment definition to model training and deployment. This DQN approach, with its structured learning and adaptation mechanism, complements the comprehensive analysis, highlighting the potential of combined and advanced techniques in enhancing IDS accuracy and effectiveness.

Keywords: *K-Nearest Neighbors , Random Forest, Artificial Neural Network, Convolutional Neural Network, Support Vector Machine, Deep Q-Networks , reinforcement learning, Intrusion Detection System.*

1. Introduction

Experts in the field of computers are now debating whether method is superior for data transfers across computer networks. ISPs are accountable for ensuring optimal network performance (ISPs). Network traffic is the initial step toward identifying and labeling uncharted groups inside the network [1]. Categorizing network data is critical for management and security applications like intrusion detection and QoS. In addition to a wide variety of other applications, this tactic may be used by network administrators to manage resources and prevent certain flows. The proliferation of network-based software is another possibility.

Classification, the process of putting things into distinct groups, occurs naturally. In each of these groups, we see how several data points are interconnected. In a network, communication takes place between nodes since they are connected. The term "traffic categorization" refers to a method for organizing the information sent through a network according to predetermined criteria. When these conditions hold, it becomes significantly less difficult to structure large data sets according to the relationships between them. If this technique is used, it may be less difficult to isolate the impacts of service network traffic. This, together with the use of protocols and ports, may help determine the load a node is under [2]. Different techniques are discussed below that may be used to secure network for IDS. Port identifiers, In addition to the IP address, The Third: Procedures, Network nodes are the building blocks of a computer system. Keeping an eye on network traffic might be useful for evaluating bandwidth use and server load. It may be enlarged all the way up to the detailed application design and the preliminary plan for the network upgrade. There are many types of network traffic, and they are as follows: One, very bandwidth-hungry traffic. Indulging in fake-traffic-watching throughout the workday, because there is no way to traffic, a high response rate is to be expected for interactive traffic yet may be disappointing. Sensitive traffic is a factor in the battle for bandwidth. This all-encompassing study concludes that rapid network growth causes an increase in both network traffic and resource usage. Network traffic categorization has the potential to enhance the precision of performance metrics while also reducing resource consumption. Analyzing bandwidth use allows for informed upgrades to existing systems.

Over the last two decades, several different categorization algorithms for network traffic have been published [3]. The first approach is the port-based method. This technology provides several opportunities for identifying network traffic. Ports need registration with the Internet Assign Numbers Authority before they may be used (lANA). It's possible that the widespread usage of P2P (Peer to Peer) programs in [5], which make use of dynamic port numbers, is to blame for the failure of this strategy.

To distinguish them from static port numbers, which have already been registered with the IANA, we use the term

[1]*Department of Computer Science and Engineering, Rabindranath Tagore University, Raisen (M.P.). chak03ankit@gmail.com*
[2]*Department of Computer Science and Engineering, Rabindranath Tagore University, Raisen (M.P.). shivshakti18@gmail.com*
** Corresponding Author: Ankit Chakrawarti*
Email: *chak03ankit@gmail.com*

"dynamic" (lANA). The second tactic puts more emphasis on the payload. When applied to network traffic classification, the results of this method are quite exact [4]. Deep Packet Inspection is an abbreviation for this technique. There is a catch, though, to this approach. There is no way to combine network apps that encrypt data to avoid interception with apps that utilize encrypted data.

The use of password-protected software also had a role in the strategy's downfall. The researchers devised a revolutionary method, which they called Machine Learning (ML), to categorize internet traffic and identify the various applications that are sent across the network. Incorporating machine learning techniques for network traffic classification improves accuracy significantly.

It is necessary to make use of a training set in addition to a test set in order to appropriately identify unknown classes [5]. Using standard methods of machine learning, such as classification of network data needs feature extraction to be carried out by subject matter experts. In recent years, deep learning-based systems for automatically extracting features have gained traction as a safer alternative to the traditional manual procedure due to the possibility of human error.

This is because of the nature of the patches being applied. However, in order for these algorithms to reach the training stage, a substantial quantity of data is necessary. These models contain many hidden layers, making it challenging to properly update their weights with a small sample of data. The findings of the research indicate that a sufficient amount of training data is essential for producing the best and most desired outputs, and that a model that lacks adequate training data performs badly.

However, deep network models have greatly improved the reliability of traffic classification via the discovery and incorporation of new data. Although it has become easier, it may still be challenging to find the right features and sufficient data for a given traffic [6]. Due to these challenges, accurate categorization of network data is challenging in most cases. Network traffic is classified by deep learning algorithms using an end-to-end deep learning-based classification strategy, eliminating the need for human feature extraction or further algorithm tuning.

In contrast to the standard practice of categorizing network traffic, this is a relatively new development. Improving the algorithm's classification performance isn't enough to make a network traffic classifier useful; it's also important to refine the approach used to arrive at those results. To get over the limitations of deep learning, the method was fine-tuned employing meta-heuristic techniques. Simple techniques and procedures allowed for many iterations, which in turn led to the identification of a workable and excellent solution.

Using reinforcement learning (RL) methods, a system may learn from its interactions with a novel environment and improve its performance over time. The agent's ability to learn from its experiences in the world is what makes this a reality. To control its actions, an RL agent will consult its policy, which may be seen of as a mapping from inputs to predetermined actions.

The term "supervised practice" refers to a kind of learning whereby a set of labeled examples is used to learn how to produce a desired result (in this case, an output). Learning by reinforcement is quite different from the aforementioned approach. The main difference is that the RL agent is never advised on what to do, but rather provided an assessment signal that shows it whether or not the action it chose was a good one. Determine the actions taken and the reward function in order to maximize the total cumulative discount rewards earned by reinforcement learning agents.

Using the provided template, the agent will be able to take part in the process of creating new memory cells. These memory cells initially store the agent's input vector, but the agent may later use other information it has stored there to make decisions on how to proceed. The challenge of traffic classification multiplies in a setting with a large amount of data, sometimes known as "Big Data" [7], and a high degree of diversity. Due to its large volume, wide variety, high veracity, and high velocity, big data has introduced a new dimension to the study of networking and traffic [8].

This research makes many recommendations for the big data ecosystem, including the categorization of network traffic using reinforcement learning. To solve this problem, researchers have created a new model for classification called DRL, which combines the decision-making powers of reinforcement learning with the modeling capabilities of deep learning.

As for the rest of the paper, it's organized as follows: In Section 2, numerous different approaches to traffic data are described along with a thorough analysis of the related literature. In section 3, we explained the reinforcement learning framework and offered a recommendation for applying RL to the problem of traffic classification. Section 4 presents the experimental results, while Section 5 wraps things up.

2. Background Study and Literature Review

In this part, we will do a cursory examination of the relevant literature. All aspects of machine learning, including methods, algorithms, performance assessments, traffic classification and forecasting, as well as machine learning in the internet of things and internet of things networks, are covered.

A. Machine Learning

Researchers in the field of "machine learning" compile massive volumes of data in order to train statistical models that may be applied to real-world issues. Through the use of learning algorithms and a data collection known as a training set, machines may acquire new skills without being specifically programmed to do so. Learn with supervision, with little supervision, without supervision, or with reinforcement [10, 11].

In the case of supervised learning, the dataset is made up of samples that have been labeled. An algorithm for supervised learning produces a model when it is provided with a dataset. This model takes in a feature vector and produces the label that corresponds to that vector. In supervised learning, the goal is to predict an output given an input. Unsupervised learning uses unlabeled samples. For this form of learning, training set labels are not necessary.

Teaching a model to reconstruct a feature vector or a real-world value from a given one is the ultimate objective of unsupervised learning. In unsupervised learning, the training set is not used until it has been labeled. The dataset used for semi-supervised training contains instances that have been tagged as well as those that have not been labeled. The vast majority of occurrences do not have names.

Although a supervised learning algorithm and a semi-supervised learning algorithm ultimately strive for the same thing, supervised learning algorithms have more direct human oversight, it differs in that it can potentially produce a more accurate model by making use of a large number of unlabeled cases. One branch of artificial intelligence is known as reinforcement learning. Feature vectors are used for training robots to operate in their intended surroundings. Your machine's ability to carry out your commands is not dependent on its current state.

Your efforts will be rewarded, and the robot will travel around the planet. Calculates the best course of action to maximize ROI in any given scenario. In time, a reinforcement learning algorithm may be trained to choose the best action, given a state and a property of that state. An ideal move is one that maximizes the mean average return [10, 11]. The labels in problems of classification in machine learning are limited. When the labels to be predicted are continuous in nature, the corresponding machine learning assignment is known as a regression problem [11].

Data analysis is the primary emphasis of Deep Learning (or DL for short), a branch of machine learning. It is a group of algorithms predicated on a deep ANN whose architecture mimics that of the human brain's biological neural network. [12] To draw inferences, deep learning models assess data in a way that is conceptually similar to human reasoning. Computer vision, NLP, voice recognition, visual object identification, bioinformatics, and medicine are just some of the many areas where DL is now being put to use. However, its use is restricted in the data network sector for a variety of reasons [9], including a lack of data, transparency, and computational resources.

Keep in mind that DL models have an insatiable appetite for data. They require access to a staggering amount of information in order to learn. Example: Due to the nano-restricted network's processing capabilities, it is difficult to implement Tesla's self-driving software on a nano-network since the program requires millions of pictures and video hours in order to operate effectively.

In addition, there is a broad variety of issues with how DL models are seen and understood. Their black box-like design makes it hard to grasp how they function [9], despite the fact that they are able to select features from input data and provide accurate predictions of output. The appropriate characteristics are chosen based on the provided data. Many computing resources are needed for deep learning. Training a deep learning network may now only take a few hours instead of a few weeks thanks to cloud computing and high-performance GPUs. However, the GPU is not optimal for implementing nano-networks, despite its higher training computation speeds [9].

Learn More About the Algorithms That Drive Machine Learning

Machine learning can solve practically every data challenge. There are benefits and cons to every algorithm. In the process of analyzing the traffic at the micro/nano gateway associated with electromagnetic nano-networks, which kind of machine learning model is going to be the most effective choice? In the following article, we'll investigate the most popular and often discussed techniques for analyzing and categorizing information gathered from wireless networks. The decision tree classifier, support vector machines, k-nearest neighbors, random forests, and neural networks are all examples of machine learning algorithms. It's possible that optimizing a learning algorithm's hyper-parameter settings is all that's needed to make the algorithm more effective.

First, a Decision Tree

Decision trees are choice-making acyclic networks. Inner nodes correspond to input vector attributes, while leaves indicate the final result. The left branch is followed if a characteristic value falls below a threshold. The terminal of the leaf node initializes the class of the example. Classification trees are widely used because they're easy to implement and can be modified for rule-based systems. They may also be drawn. Top-down greedy algorithm trains the model. This technique separates nodes frequently, and its optimization criteria is typically information obtained. Many DTC usage include categorization. It works for continuous and categorical dependent variables [12].

Seond Support Vector Machine-based Systems (SVM)

Separate-Variable Models (SVMs) are a kind of binary classifier that does not use probabilities. Their strategy entails representing each feature vector in a multidimensional space and looking for a linear separation between classes. Their systems cause this. There are cases when linear space partitioning is not only impractical, but also unable to give a workable solution. Therefore, the dimension of the space is increased using the kernel method [11], [12] to facilitate a separation that is much easier in a space with a substantially greater number of dimensions.

Third, K-Nearest Neighbors (K-NN)

Classification and regression may both benefit from the non-parametric K-NN technique. The K-closest training instances in the feature space will be the output in either scenario. When using K-NN to categorize data, a class membership is the result. An object is assigned to the category in which it is most popular among its K nearest neighbors, as decided by the votes of its neighbors [11].

In this paper, we present three new techniques that inject UAP into network traffic. By injecting a UAP into the packet content, the AdvPad attack may assess how well packet classifiers can handle unexpected data. The AdvPay attack modifies a faked packet by inserting a UAP into the packet's payload in order to gauge the efficacy of flow content classifiers.

The AdvBurst attack is used to test the robustness of flow time series classifiers by including a predetermined number of spoofed packets in the intended burst of a flow. These bogus packets have been crafted using statistical characteristics that were taken from an actual UAP. When even a little amount of UAP was introduced to the traffic, the overall performance of DL-based network traffic classifiers dropped dramatically, as shown by the findings [13].

The author of this piece examines and evaluates a number of well-known techniques for machine learning, any of which may be put to use in conjunction with information obtained from the network activity of Internet of Things (IoT) devices. We make use of a data collection that is available to the public and contains network traces spanning 20 days from 20 well-known Internet of Things devices. In order to extract useful features, first the network traces are evaluated.

In the next step, We conducted an analysis of recent survey papers to determine the most innovative machine learning approaches for the classification of Internet of Things traffic. In the next step, we compared several machine learning algorithms' results across a range of metrics, such as their classifying prowess, training duration, and overall processing speed. Finally, based on the data we gathered, we provided some guidelines for selecting the best machine learning algorithm for various applications [14].

In the last stage of the process, the hybrid suggested model is put into action by making use of the machine learning strategy known as Random Forest (RF) to choose relevant characteristics from the merged dataset (which includes V2V and V2R communications). The Gated Recurrent Unit (GRU) approach is used to forecast the flow of network traffic; it is the deep learning algorithm that has been shown to be the most accurate. The results of the simulations reveal that the proposed RF-GRU-NTP model beats the most advanced algorithms currently available for network traffic prediction [15] in terms of runtime and the number of inaccurate predictions.

Micro- and nano-gateway traffic from nano-networks will be categorized. The nano-network traffic will be evaluated and classified using five supervised machine learning methods. This study seeks to find the best classifier for nano-network traffic by testing suggested models, evaluating them, and comparing their accuracy and performance to other classifiers [16].

In SSDDQN's current network, an autoencoder reconstructs traffic characteristics and a deep neural network classifies. The present network handles both of these tasks. K-means clustering and deep neural network prediction are used in the target network. Training and testing use the NSL-KDD and AWID datasets. Additionally, a full comparison of different machine learning models is offered. Experiments show that SSDDQN is beneficial in terms of time complexity and yielded good results across a range of evaluation measures [17].

A detailed overview of the procedure for acquiring the data, which includes its preparation and anonymization, is presented here. To display the data on network traffic, we make use of t-distributed stochastic neighbor embedding (t-SNE). This makes it much easier to understand the dynamics of traffic and the communication channels [18].

The diagram in footnote 18 illustrates a high-level pipeline architecture and flow-based routing applications.

Our results show that even with massive imbalanced datasets, our technique is able to categorize network traffics accurately and quickly. Our research leads us to believe that it might help modern NIDS systems based on machine learning deal with the serious issue of imbalanced datasets. [19].

3. Proposed Method

3.1 Reinforcement Learning

In order to address problems that can't be handled by either machine learning or traditional learning alone, artificial intelligence researchers have developed a technique called reinforcement learning. To deal with unpredictable and ever-changing environments, reinforcement learning may be used (RL). The approach is based on the tactical utility function and is sequential and multi-step. The intended result of this strategy is maximum effectiveness. The actor network in an actor-critic architecture is the part responsible for acting on the basis of the system's interactions with its external environment and the states it is in at any given time. One possible approach to implementing real-time thinking is this (RL). The strategic utility function is built by the critic network and relies on the effectiveness of the actor network. Next, the critic network use this function to fine-tune itself and improve estimate precision [20].

Reinforcement learning has matured into a very sophisticated learning framework in comparison to the numerous models of machine learning that are now in use. During play, players craft a policy by combining various actions and incentives. Having this policy in place guarantees that the necessary educational traits are met. More and more scientists from many different technology fields have used it in the last decade. Resource management, intelligent systems, optimization problems, and image processing are all examples [21]. It is possible that difficult tasks requiring sequential decision-making might be simplified and made easier to solve by combining deep learning with reinforcement learning. This approach takes on one of the central problems in AI head-on: the development of self-sufficient creatures that learn to interact with their surroundings.

It is important to us to make decisions that will have lasting, far-reaching effects. With RL, the agents are the primary focus of the optimization. But unlike humans, RL agents can frequently learn how to do things well from scratch. This is one of the main reasons why there is a huge gap between how RL bots utilize data and how people do. This illustrates that agents can tackle a wider range of problems if given the means to make use of the knowledge they currently possess. Employing it to manage load balancing issues in distributed SDN controllers has been found to be beneficial in recent study [22].

Agents, states, and incentives all play a role in RL learning from its dynamic environment. To modify its state, the agent acts in response to input it receives through a critical evaluation of its surroundings; this feedback is the reward [23]. The agent's state is a representation of the environment's current state. The policy function for a state space S defines the conditional distribution of an action, indicated by a, in a Markov Decision Process (MDP).

$\pi\left(\cdot\middle|s,a\right)$ and the respective next state transition dynamics $\Gamma\left(\cdot\middle|s,a\right)$. The task function is defined as $p(\Gamma)$ where every task comprises of the initial state distribution $p(s_0)$, transition distribution and the reward function. The policy evaluation is performed through the reward function $r:S\times A\rightarrow \square$ where S and A represent the state space and action respectively. The objective of reinforcement learning is to derive a policy which maximizes the expected return $\eta\left(\pi,\Gamma\right):=\underset{s_0\sim p_0}{\mathrm{E}}\left[V^{\pi,\Gamma}\left(s_0\right)\right]$ where $V^{\pi,T}\left(s_0\right)$ is the value function for a policy π at a state s over the transition dynamics Γ, such that $\left[V^{\pi,\Gamma}\left(s\right)\right]:=\underset{a_t,s_t\sim\pi,\Gamma}{\mathrm{E}}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,|\,s_0=s\right]$. Here $\gamma\in\left[0,1\right]$ is the discount function to add flexibility

3.2. Proposed working flow

The model for network traffic is the topic of this section. Part of this model is a method with discrete operations, as in Figure 1. A step-by-step procedure for network traffic using machine learning is outlined here.

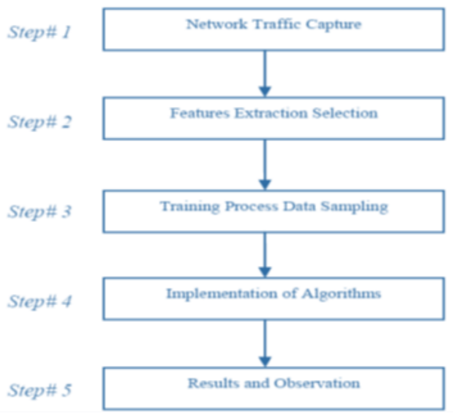
Step# 1
Network Traffic Capture
Step# 2
Features Extraction Selection
Step# 3
Training Process Data Sampling
Step# 4
Implementation of Algorithms
Step# 5
Results and Observation

Fig 1. Proposed working flow

A. The Capture of Network Traffic

The first step, which is also the most crucial, is to gather data. During this part of the process, real-time recordings of the network traffic are being made. The same activity may also be referred to by the phrase "data collection." Even though there are a number of other tools available, the Tcpdump application may still be used to capture live information from the network. This is the case despite the fact that there are other programs accessible. We monitor and record activities on the network with the help of the Wire Shark tool (IS), which does this by collecting and examining data packets. During that one minute, traffic from WWW, DNS, FTP, P2P, and Telnet applications was logged.

B. Feature Extraction

After collecting information on network traffic, the next step is to choose and extract features from the data. This process involves the extraction of characteristics such as packet length, inter-arrival time protocol, and so on. After the characteristics have been extracted, a machine learning classifier may be trained using them. It is feasible to extract features from a recorded dataset by using a script written in the programming language Perl. On the other hand, the 23 characteristics are extracted using the Netmate program. For the purpose of storing the dataset required by the Weka analysis tool, Comma Separated Values (CSV) files created in Microsoft Excel are used.

C. Taking Samples During the Training Process

The supervised learning method draws examples for its practice from both of these data sets. During the initial step of the supervised learning process, data are tagged in order to network applications that have not yet been classified.

D. The RL Algorithm Is Put Into Implementation

RL algorithms have the capability of learning how to swiftly complete a new job by making use of the information gained from past attempts. In this approach, examples from previously completed activities are utilized to assist in the learning of new ones. For meta-reinforcement learning approach, the family of tasks (Ψ) is defined through the MDP which have been parameterized as $\left\{\left(S, A, \Gamma, r_{\psi}, p(s_0), \gamma\right)\right\}_{\psi \in \Psi}$ and produces distinct reward functions while maintaining the same transfer dynamics. The value function on a task with policy π, dynamics Γ and reward r_{ψ} is denoted as $V_{\psi}^{\pi,\Gamma}$. Also, the respective expected return function is denoted by $\eta\left(\pi, T, \psi\right) = E\left\{V_{\psi}^{\pi,T}(s_0)\right\}$.

The meta-training method utilizes the history of past transition dynamics or context Ψ, which is referred to as c, in order to learn a policy from the specific task that is being considered and adapt to it accordingly. The sum total of the knowledge gained throughout the training period may be symbolically represented as: $c_{n=1:N}^{\psi}$ where c_n^{ψ} is one transition in a task. It is expected that the policy will be able to adapt to the new task selected from the family of tasks based on information gleaned from context variables during the testing phase. These parameters are sampled and recorded frequently during the whole event to increase the depth of the investigation. Priors of the trajectory over the context variables are created and used in the fast adaptation at the trajectory level [21]. The actor network in the RL paradigm is in charge of enacting action, while the critic network is in charge of evaluating the effectiveness of that action and deciding whether or not to reward it. The policy framework is derived from this structure via the application of several tuning rules.

3.3 Inverse Monte Carlo Learning

Learning about MDP transitions is not necessary thanks to the Monte Carlo method of reinforcement learning. Instead, it's via specific experiences that growth occurs. In this instance, the rate of return or reward is decided by chance.

Note that it is restricted to episodic MDPs, which is a significant caveat. It's fair to ask why now that things have gotten to this point. This is because we can't calculate profits until the show is over. In this section, we do not update for each and every single action or behavior; rather, we do it after each and every episode. We are able to determine the value by using the most basic concept imaginable, which states that the value is equivalent to the typical return of all of the sample trajectories for each condition.

It is essential to bear in mind the concept of multi-armed bandits that was presented in this article; each state represents a unique instance of the issue of multi-armed bandits, and the goal is to conduct actions that are optimum for all of the multi-armed bandits at the same time.

The value function for a particular random policy may be determined in a process called policy assessment, and the optimal policy can be determined in a process called policy improvement. These two procedures are quite similar to what's done in dynamic programming. Both of these phases will be discussed at length in the following two sections.

3.4 Monte Carlo Policy Evaluation

Again, the goal is to learn the value function vpi(s) by repeated practice under the constraints of some policy pi [24]. Please keep in mind that the return is calculated by subtracting the discount from the total award:

S1, A1, R2, ….Sk ~ pi

The issue that has to be answered is how these sample returns may be obtained. In order to do this, we will need to go through a number of episodes in order to produce them [25].

A progression of states and prizes will be available to us for each episode that we play. In addition, using these rewards, we are able to compute the return, which, by definition, is just the accumulation of all future rewards.

Only for the very first time that Monte Carlo is visited during an episode do average returns apply.

The following is an explanation of the method in step-by-step format:

1.Perform the initialization of the policy and the state-value function.

2.To begin, an episode should be created in accordance with the existing guidelines.

1. Maintain a record of the states that were visited during the episode.

3.Choose a state in section 2.1.

1. Include in a list the result that was obtained following the occurrence of this state for the first time.
2. Average over all returns
3. The value of the state should be set to the calculated average of those values.

4.Repeat step 3

5.Repeat 2-4 until satisfied

Every time you go to Monte Carlo: These are the average returns for each time you go there throughout an episode.

Altering the wording of this method's step #3.1 so that it reads "Add to a list the return received after every occurrence of this condition" is all that is needed to get the outcomes that are wanted.

In order to have a better grasp on this idea, let's look at a straightforward illustration. Imagine there is a setting in which we have access to two different states: A and B. Let's imagine that we watched two different episodes as a sample:

A+3 -> A+2 -> B-4 -> A+4 -> B-3 -> Terminate

B-2 -> A+3 -> B-3 -> Terminate

The transition from state A to state A, with a reward of +3, is denoted by the expression "A+3 => A." Let's figure out the value function by combining the two approaches:

First Visit	Evert visit
V(A) = ½(2+0)=1 V(B) = ½ (-3+-2) = -5/2	V(A) =1/4(2+-1+1+0)=1/2 V(B) = ¼(-3 + -3 + -2 + -3) = -11/4

3.5 Proposed working algorithm for IDS

1. Environment Setup: Define the network environment where the IDS will operate. This includes setting up network traffic data for training and testing, and defining the state space that represents different network scenarios.
2. Choosing a Reinforcement Learning Model: Select an appropriate RL algorithm Deep Q-Networks suitable for the IDS's objectives.
3. Defining Rewards and Penalties: Establish a reward system to reinforce desirable actions (correctly identifying threats) and penalties for undesirable actions (false positives/negatives).
4. Feature Extraction: Process the network traffic data to extract relevant features that the RL model can understand and use for decision-making.
5. Training the Model: Train the RL model using the network traffic data, allowing it to learn from interactions with the environment and improve its decision-making process over time.
6. Policy Development: Develop a policy for the model to decide what action to take in different states (e.g., raising an alert for potential threats).

3.6 Proposed Deep Q-Networks algorithm for IDS

1. Preprocessing: Collect and preprocess network traffic data. Normalize or standardize the features for effective learning.
2. Defining the Environment: Define the state space (network scenarios), action space (alerts, no action), and reward system (positive for correct detections, negative for false alarms).
3. Initialize Deep Q-Network: Initialize a neural network with input layers (matching the number of features in your data), hidden layers, and output layers (representing possible actions).

4. Set Hyperparameters: Set hyperparameters like learning rate, discount factor, and exploration rate.
5. Experience Replay Memory: Initialize a replay memory to store experiences (state, action, reward, next state).
6. Training Loop:
   - Collect Data: For each episode, collect state information from the network environment.
   - Select Action: Use the network to select an action based on the current state (employ exploration vs. exploitation strategy).
   - Execute Action & Observe Reward: Perform the selected action and observe the reward and new state.
   - Store Experience: Store this experience in the replay memory.
   - Sample Mini-batch: Randomly sample a mini-batch of experiences from the memory.
   - Compute Q-Value: Use the network to compute the Q-value for each mini-batch experience.
   - Update Network: Update the neural network weights using backpropagation to minimize the loss between predicted Q-values and target Q-values.
7. Model Evaluation: Regularly evaluate the model on a separate validation dataset to monitor its performance.

3.7 Advantages of proposed method

1. Efficient Learning from High-Dimensional Sensory Inputs: DQNs are particularly adept at processing and learning from high-dimensional data, a common characteristic of network traffic.
2. Stability and Convergence: The use of experience replay in DQNs helps in stabilizing the learning process and ensures convergence, which can be an issue in traditional reinforcement learning methods.
3. Handling Large Action Spaces: DQNs are effective in environments with large action spaces, making them suitable for complex IDS scenarios where numerous potential actions and responses are possible.
4. Integration of Deep Learning Advantages: By combining Q-learning with deep learning, DQNs leverage the pattern recognition capabilities of neural networks, enhancing the ability to detect sophisticated and novel intrusion patterns.
5. Overcoming Limitations of Traditional RL: Traditional reinforcement learning methods can struggle with correlated data and non-stationary distributions, issues that DQNs can handle more effectively.

4. Implementation and Result

4.1. Performance Metrics

It might be difficult to determine a model's quality without monitoring its progress as it is trained and tested. The identification of an error category, the extent to which the model is in agreement with the data, or some other relevant measure is commonly used to do this. Using a categorization strategy, we can make predictions about one of four possible outcomes. The performance metrics are controlled by a matrix constructed from these data; this matrix is termed the confusion matrix. Using this matrix, the efficacy of a classifier may be assessed.

Components off the diagonal represent data points for which the classifier made an incorrect label prediction, whereas diagonal elements represent the proportion of points for which the predicted and actual labels were identical. Having a big number of right guesses is an indication of a high confusion matrix diagonal value. While testing for network traffic prediction, accuracy is determined by how close network traffic samples are to one another. The precision with which one can identify network traffic is proportional to the closeness of a given number. Exponent 1 represents the accuracy of the data on network traffic.

Accuracy = (TP + TN) / (TP + TN + FP + FN) (1)

When a TP indicates a true positive and a TN indicates a true negative, an FP indicates a false positive and a FN indicates a false negative.

4.2 Dataset Details

Magnetic nano-network communication and its protocol stack may be simulated using NS2, also known as Simulator. The simulations in Simulator are based on events. The packet size is determined by the user, and the simulator's message processing unit generates random packets at regular intervals. Networking abstraction layer allows the inclusion of Fig. 2. Minimal Header Format for Network Messages [27]. Through transparent-MAC [27], packets are transmitted from the network layer to the physical interface of the nano-router with a network header that is not reliant on the routing approach and a MAC layer that follows a strategy over which it has no control.

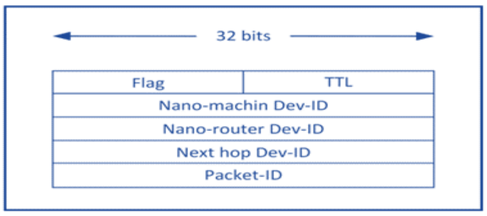

Fig 2. Template for the header of nano-network messages [27].

Table 2. Micro- and nano-gateway traffic categorization and forecasting input characteristics [33].

Field	Feature description
flag_id	Flag identification
ttl	Time to live
Source_dev_id	Source nano-device identification
Sender_dev_id	Sender nano_device identification
Next_hop_dev_id	Next hop nano-device identification
Packet_id	Packet identification
Source_IP	Source IP address
Destination_IP	Destination IP address
Transport_protocol	IP transport protocol number
Source_port	Source port number
Destination_port	Destination port number
Payload	Message
Payload_size	Message size
Header_size	Header size
Packet_size	Packet size
Source_mac	Source MAC address
Destination_mac	Destination MAC address

Table 3. Labels that are outputted for the purpose of micro- and nano-gateway traffic categorization and prediction.

Field	Label description
NN0	Nano-to-nano-communication packet
NN1	Nano-to-Internet-communication packet
TCP	TCP packet
UDP	UDPpacket

4.3 Decision Tree

Figure 3 depicts the variation in the learning curves of the unoptimized model when the training set and cross-validation set sizes are varied. This is the fruit of using accessible data to train a decision tree classifier. Overfitting is evident at 50 training points, when the model achieves perfect accuracy on both the training and testing data.

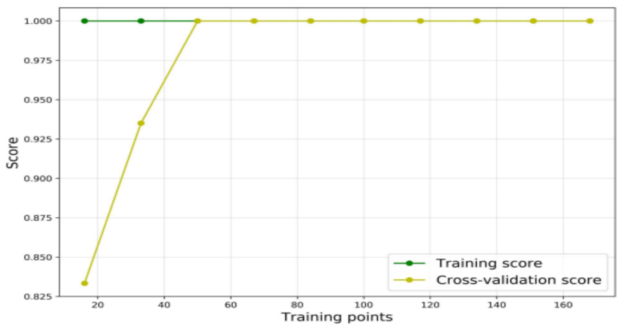

Fig 3 : Learning curves that are typical for the DTC model, which has not yet been optimize.

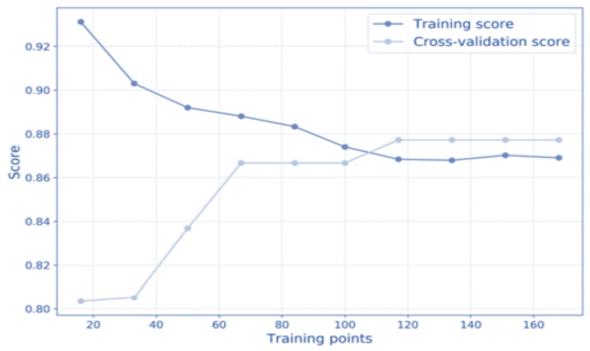

Fig 4 : The DTC model's learning curves, which have been optimized.

The enhanced learning curves are shown in Figure 4. When the number of training samples is equal to 110, the training score and cross-validation score are nearly identical. The DTC model will experience overfitting as the number of training points increases.

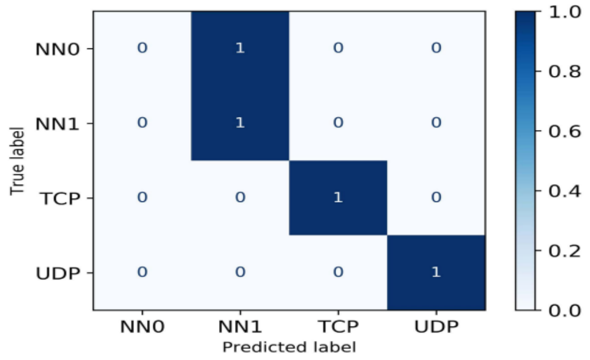

Fig 5 : Normalized confusion matrix for optimized DTC model.

Figure 5 shows DTC's normalized confusion matrix. This displays how successfully the model categorizes TCP, UDP, and NN1 packets. Since it expects all NN0 packets to be NN1, it doesn't detect any. The model can discriminate between conventional and nano-network traffic, but not nano domain packets.

4.4 Support Vector Machines

Figure 6 is a representation of the unoptimized model's learning curves that was created by fitting the support vector machine model. The training score and the cross-validation score both rise as more training points are accumulated, eventually reaching their respective maximums of (85%) and (90%) when a total of approximately 150 training points have been earned. The SVM model will become more susceptible to overfitting as the amount of training data continues to grow.

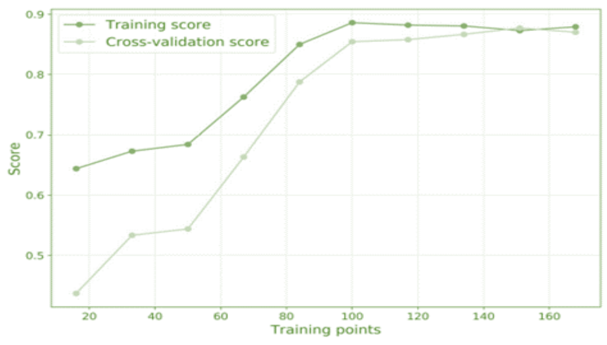

Fig 6 : Typical learning curves for an unoptimized support vector machine model.

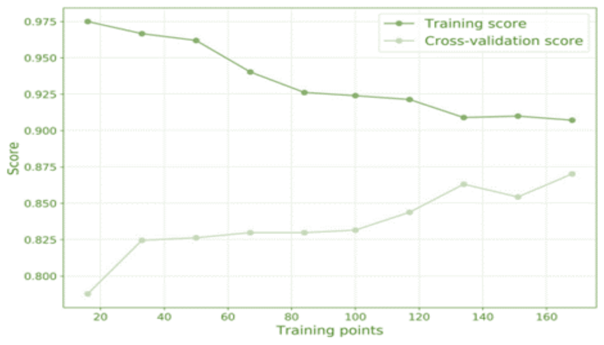

Fig 7: Optimal SVM model learning curves.

The improved model's learning curves are shown in Figure 7, which may be found here. Both the training score and the cross-validation score go up when the number of training points goes up, but the training score goes down, while the cross-validation score goes up. These two scores do not overlap for any of the training samples.

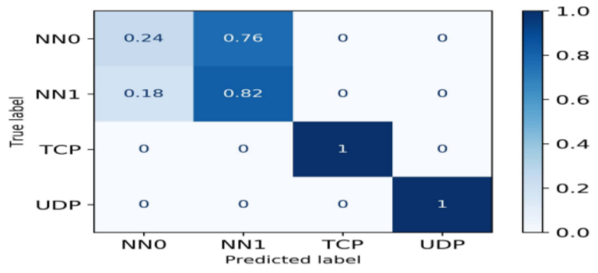

Fig 8 : For an improved SVM model, the normalized confusion matrix.

Figure 8 depicts the enhanced support vector machine's normalized confusion matrix. These results demonstrate that the model correctly detects TCP and UDP packets. Only 24% of NN0 packets can be expected effectively, whereas 82% of NN1 packets can. The model differentiates effectively between big and small network traffic, however it predicts erroneously for nano-domain packet transmission. Predictions from the model indicate that 76% of all NN0 packets will be NN1 ones.

4.5 KNN model

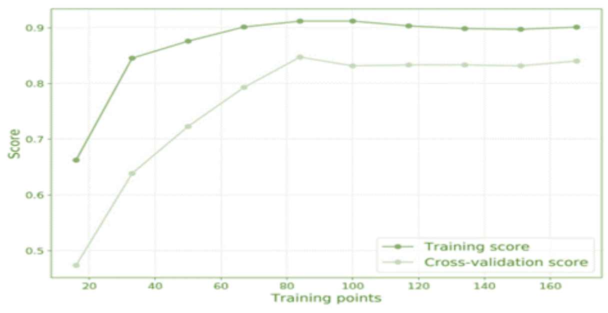

Fig 9: Curves of learning for an unoptimized KNN model

The model's learning curves before and after being fitted to the K-nearest neighbors model are shown in Figure 9. It demonstrates that, without converging, training and cross-validation scores climb together as training points increase.

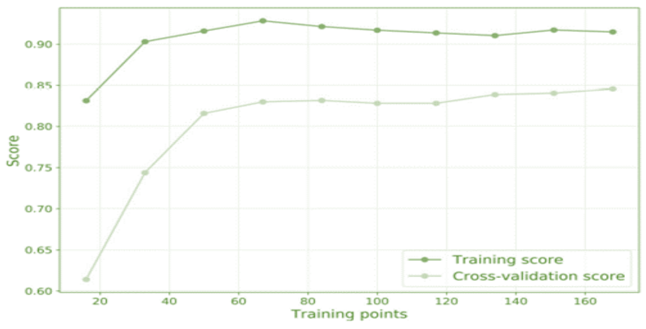

Fig 10: Curves of learning for the best KNN model.

Figure 10 depicts the improved KNN model's learning curves. This model avoids overfitting and underfitting for all training points, and it has an accuracy rate of 91.11% overall.

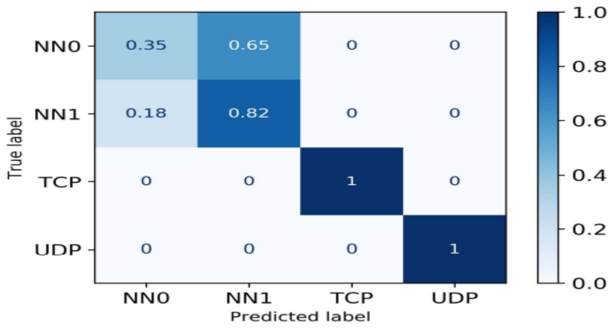

Fig 11 : The optimized KNN model features a normalized confusion matrix.

KNN confusion matrix with enhancements is shown in Figure 11. TCP and UDP packets are separately modeled in this framework. 97% of NN2 packets are accurately anticipated, compared to 82% of NN1 and 35% of NN0. The model can tell the difference between traffic on large and small networks, but its estimates for packet traffic in the nano-domain are off. Despite its preference for NN1 packets, the model outperforms its optimized SVM counterpart. Only 35% of NN0 packets are predicted properly by the KNN model; the rest are incorrectly labeled as NN1.

4.6 The comparison of the proposed system with existing systems

Table 4. Comparisons of various ML and DL precision metrics with the proposed algorithm.

Algorithm used	Accuracy (%)
KNN , RF	72.08 , 90.53
ANN	78.00
KNN	94
CNN	94
SVM	94.2
KNN, RF, NN, and NB	79.6,84.8,84.6,and 87.6
Proposed method	96.8

The table 4 provided data compares the accuracy of various algorithms used in an Intrusion Detection System (IDS). The accuracies are as follows: KNN and RF (72.08%, 90.53%), ANN (78.00%), KNN (94%), CNN (94%), SVM (94.2%), a combination of KNN, RF, NN, and NB (79.6%, 84.8%, 84.6%, and 87.6% respectively), and the proposed method (96.8%). This indicates that the proposed method outperforms the others in terms of accuracy.

5. Conclusion

Intrusion Detection System (IDS) significantly outperforms other methods in terms of accuracy. With an accuracy of 96.8%, it surpasses traditional approaches like KNN, RF, ANN, CNN, and SVM, which show varied accuracies with the highest being 94.2% for SVM. The data also highlights the effectiveness of combining multiple methods (KNN, RF, NN, and NB) but still shows that the proposed method holds a clear advantage in accurately detecting intrusions. This indicates a strong potential for the proposed method to be more effective in real-world applications of IDS.

Author contributions

Ankit Chakrawarti: Conceptualization, Methodology, Software, Field study, Data curation, Writing-Original draft preparation, Software, Validation., Field study. **Dr. Shiv Shakti Shrivastava:** Visualization, Investigation, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] Sharon A, Mohanraj P, Abraham TE, Sundan B, Thangasamy A. An intelligent intrusion detection system using hybrid deep learning approaches in cloud environment. InInternational Conference on Computer, Communication, and Signal Processing 2022 Feb 24 (pp. 281-298). Cham: Springer International Publishing.

[2] Friha O, Ferrag MA, Benbouzid M, Berghout T, Kantarci B, Choo KK. 2DF-IDS: Decentralized and differentially private federated learning-based intrusion detection system for industrial IoT. Computers & Security. 2023 Apr 1;127:103097.

[3] Sultana N, Chilamkurti N, Peng W, Alhadad R. Survey on SDN based network intrusion detection system using machine learning approaches. Peer-to-Peer Networking and Applications. 2019 Mar;12:493-501.

[4] Rao KN, Rao KV, PVGD PR. A hybrid intrusion detection system based on sparse autoencoder and deep neural network. Computer Communications. 2021 Dec 1;180:77-88.

[5] Amir MS, Bhatti G, Anwer M, Iftikhar Y. Efficient & Sustainable Intrusion Detection System Using Machine Learning & Deep Learning for IoT. In2023 4th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET) 2023 Mar 17 (pp. 1-6). IEEE.

[6] Elsayed R, Hamada R, Hammoudeh M, Abdalla M, Elsaid SA. A Hierarchical Deep Learning-Based Intrusion Detection Architecture for Clustered Internet of Things. Journal of Sensor and Actuator Networks. 2022 Dec 28;12(1):3.

[7] Rullo A, Midi D, Mudjerikar A, Bertino E. Kalis2. 0-a SECaaS-Based Context-Aware Self-Adaptive Intrusion Detection System for the IoT. IEEE Internet of Things Journal. 2023 Nov 20.

[8] Balamurugan E, Mehbodniya A, Kariri E, Yadav K, Kumar A, Haq MA. Network optimization using defender system in cloud computing security based intrusion detection system withgame theory deep neural network (IDSGT-DNN). Pattern Recognition Letters. 2022 Apr 1;156:142-51.

[9] Seifi S, Beaubrun R, Bellaiche M, Halabi T. A Study on the Efficiency of Intrusion Detection Systems in IoT Networks. In2023 International Conference on Computer, Information and Telecommunication Systems (CITS) 2023 Jul 10 (pp. 1-8). IEEE.

[10] Iwendi C, Srivastava G, Khan S, Maddikunta PK. Cyberbullying detection solutions based on deep learning architectures. Multimedia Systems. 2023 Jun;29(3):1839-52.

[11] Sethi M, Verma J, Snehi M, Baggan V, Chhabra G. Web Server Security Solution for Detecting Cross-site Scripting Attacks in Real-time Using Deep Learning. In2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1) 2023 Apr 21 (pp. 1-5). IEEE.

[12] Javeed D, Gao T, Jamil Z. Artificial Intelligence (AI)-based Intrusion Detection System for IoT-enabled Networks: A State-of-the-Art Survey. InProtecting User Privacy in Web Search Utilization 2023 (pp. 269-289). IGI Global.

[13] Raheema AQ. Threat Analysis in IOT Network Using Evolutionary Sparse Convolute Network Intrusion Detection System. International Journal of Online & Biomedical Engineering. 2023 Mar 1;19(3).

[14] Isaza G, Ramirez F, Duque N, Lopez JA, Montes J. DDoS Attacks Detection with Deep Learning Model Using a Cloud Architecture. InSustainable Smart Cities and Territories International Conference 2023 Jun 21 (pp. 87-96). Cham: Springer Nature Switzerland.

[15] Flak P, Czyba R. RF Drone Detection System Based on a Distributed Sensor Grid With Remote Hardware-Accelerated Signal Processing. IEEE Access. 2023 Dec 5.

[16] Satheesh N, Rathnamma MV, Rajeshkumar G, Sagar PV, Dadheech P, Dogiwal SR, Velayutham P, Sengan S. Flow-based anomaly intrusion detection using machine learning model with software defined networking for OpenFlow network. Microprocessors and Microsystems. 2020 Nov 1;79:103285.

[17] Hamidouche M, Popko E, Ouni B. Enhancing IoT Security via Automatic Network Traffic Analysis: The Transition from Machine Learning to Deep Learning. arXiv preprint arXiv:2312.00034. 2023 Nov 20.

[18] C. Hardegen, B. Pfülb, S. Rieger and A. Gepperth, "Predicting Network Flow Characteristics Using Deep Learning and Real-World Network Traffic," in IEEE Transactions on Network and Service Management, vol. 17, no. 4, pp. 2662-2676, Dec. 2020, doi: 10.1109/TNSM.2020.3025131.

[19] Y. Uhm and W. Pak, "Service-Aware Two-Level Partitioning for Machine Learning-Based Network Intrusion Detection With High Performance and High Scalability," in IEEE Access, vol. 9, pp. 6608-6622, 2021, doi: 10.1109/ACCESS.2020.3048900.

[20] S. Mo, X. Pei and C. Wu, "Safe Reinforcement Learning for Autonomous Vehicle Using Monte Carlo Tree Search," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 6766-6773, July 2022, doi: 10.1109/TITS.2021.3061627.

[21] X. Mo, S. Tan, B. Li and J. Huang, "MCTSteg: A Monte Carlo Tree Search-Based Reinforcement Learning Framework for Universal Non-Additive Steganography," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 4306-4320, 2021, doi: 10.1109/TIFS.2021.3104140.

[22] J. Lu, D. He and Z. Wang, "Secure Routing in Multihop Ad-Hoc Networks With SRR-Based Reinforcement Learning," in IEEE Wireless Communications Letters, vol. 11, no. 2, pp. 362-366, Feb. 2022, doi: 10.1109/LWC.2021.3128582.

[23] P. Ladosz et al., "Deep Reinforcement Learning With Modulated Hebbian Plus Q-Network Architecture," in IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 5, pp. 2045-2056, May 2022, doi: 10.1109/TNNLS.2021.3110281.

[24] P. Xu et al., "Active Power Correction Strategies Based on Deep Reinforcement Learning—Part I: A Simulation-driven Solution for Robustness," in CSEE Journal of Power and Energy Systems, vol. 8, no. 4, pp. 1122-1133, July 2022, doi: 10.17775/CSEEJPES.2020.07090.

[25] H. Shuai and H. He, "Online Scheduling of a Residential Microgrid via Monte-Carlo Tree Search and a Learned Model," in IEEE Transactions on Smart Grid, vol. 12, no. 2, pp. 1073-1087, March 2021, doi: 10.1109/TSG.2020.3035127.

[26] J. Kim, B. Kang and H. Cho, "SpecMCTS: Accelerating Monte Carlo Tree Search Using Speculative Tree Traversal," in IEEE Access, vol. 9, pp. 142195-142205, 2021, doi: 10.1109/ACCESS.2021.3120384.

[27] G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, ‘‘Nano-sim: Simulating electromagnetic-based nanonetworks in the network simulator 3,’’ in Proc. 6th Int. Conf. Simulation Tools Techn., Jul. 2013, pp. 203–210.