BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **24**, No 3
Sofia • 2024 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2024-0032

AHT-QCN: Adaptive Hunt Tuner Algorithm Optimized Q-learning Based Deep Convolutional Neural Network for the Penetration Testing

Dipali Railkar, Dr. Shubhalaxmi Joshi

School of Computer Science, Dr. Vishwanath Karad MIT World Peace University, Pune, India

E-mails: railkar.dipali@gmail.com shubhalaxmi.joshi@mitwpu.edu.in

***Abstract*: *Penetration Testing (PT), which mimics actual cyber attacks, has become an essential procedure for assessing the security posture of network infrastructures in recent years. Automated PT reduces human labor, increases scalability, and allows for more frequent evaluations. Real-world exploitation still challenges RL-based penetration testing because the agent's many possible actions make it hard for the algorithm to converge. To resolve these shortcomings, a deep learning- model named Adaptive Hunt Tuner algorithm optimized Q-learning based deep Convolutional neural Network (AHT-QCN) is developed for efficient PT. Specifically, the Q-learning employed in this model improves its efficiency by enabling optimal policy learning for decision-making. In addition, the Adaptive Hunt Tuner (AHT) algorithm enhances the model's performance by tuning its parameters with reduced computational time. The experimental outcomes demonstrate that the developed model attains 95.25% accuracy, 97.66% precision, and 93.81% F1 score.*

***Keywords*: *Deep learning, Penetration testing, Q-learning, Adaptive hunt tuner algorithm, Deep convolutional neural network.*

1. Introduction

The process of conducting approved attacks on networks and computer systems to find whichever security flaws can be exploited is known as Penetration Testing (PT) [6]. Network systems, including database systems [8], network equipment systems [9], host operating systems [7], and so on, have utilized numerous PT methodologies for identifying vulnerability and assessing overall security. Based on the many testing procedures, PT methods can be broadly classified as manual, automatic, or intelligent [5]. The majority of PT procedures were initially carried out manually, and the efficacy of manual PT was based on studies being conducted against a small number of hosts. Subsequently, as computer networks grew in number, PT tools had to become more automated to cover ground faster [10, 1]. Automated PT seeks to accommodate the intricacies of contemporary networks while easing the time and financial constraints connected with manual, traditional approaches [11, 3]. However, in certain highly sophisticated enterprises with hundreds or even thousands of IP addresses, increasingly intricate applications, and virtualization, automation was inadequate; PT professionals struggled to evaluate each component's security promptly [1] and there are other significant obstacles to this move toward automation, particularly in complex PT scenarios [3].

PT is considered one of the significant cyber security exercises [2], therefore in recent years, various data-driven learning techniques have evolved for cyber security management. Artificial Intelligence (AI) techniques with the machine and deep learning approaches ensemble various hybrid approaches for solving various cyber security issues [26]. Recent developments in AI also have given researchers studying automated and intelligent PT a fresh perspective. Among these, Reinforcement Learning (RL) techniques are shown to be a versatile and successful strategy [2]. In RL, the Deep Q-Network (DQN) method has been frequently employed in recent years to enhance PT efficiency [12]. Numerous trials have shown that the DQN algorithm performs better than the prior technique [13]. Further, utilizing the traditional Q-learning algorithm and a dueling network mechanism to train the conventional Q-function, enhanced the DQN algorithm and thus there was an improvement in training efficiency [4]. The study analyzes computer networks' responses to attacks by examining susceptibility, exposure, infection, and recoverability using nonhomogeneous differential equations [28]. To categorize PT problems based on situations [15], RL and imitation learning techniques are utilized. In addition, these techniques give the agent presumptive knowledge about a particular network structure and help the agent to come up with a better solution by exploring their problem area in a better way. However, its specialist knowledge is less interpretable and generalizable due to the created penetration test scenarios [2]. Though various approaches have been Put forth to address Partially Observed Markov Decision Process (POMDP), like combining model-free reinforcement learning with recurrent models [15] or approximate approaches using neural networks and conventional POMDP models, the successful application of these approaches to address the complex environments in PT requires additional research [3]. To solve those above challenges, a deep learning-based technique is developed in this research for PT.

The research aims to create an AHT-QCN model for efficient PT, where the implementation of Q-learning with deep CNN leads to adaptive and intelligent decision-making with effective PT results. The AHT algorithm improves the model's performance and reduces the need for more computational resources. The main contributions of the research are,

Adaptive Hunt Tuner (AHT) algorithm: The AHT is created by combining the hunting traits of the coati and swarm behavior of the particle swarm, where the issue of falling in local optima and slow convergence of coati is resolved by utilizing the high velocity of particle swarms with improved search ability and fast convergence.

AHT Q-learning based deep CNN (AHT-QCN): The incorporation of the Q-learning with the deep CNN for PT improves its accuracy in identifying the vulnerabilities and the q values from Q-learning help to develop an efficient model. The AHT optimization tunes the deep CNN classifier to improve performance.

2. Literature review

Ghanem, Chen and Nepomuceno [1] introduced an Intelligent Automated PT network (IAPTF) with RL, where the framework served as an efficient approach for PT with high generalization ability and the approach resolved the PODP issues. However, complex attack vectors were missed in the framework because of the decrease in covered attack vectors.

Wang et al. [2] employed a DQfD-AIPT, a PT intelligent framework in which deep Q learning was utilized to plan the PT path, which reduced the overfitting issue. The algorithm showed improved efficiency in PT and attained a high cumulative reward. Nevertheless, the covered expert knowledge was very low, which led to limited performance.

Li, Zhang and Yang [3] utilized an efficient POMDP-driven PT agent named EPPTA. The employed framework provided reduced convergence time and high performance with improved scalability to enhance network security. However, the approach achieved limited performance in dynamic network environments.

Yi and Liu [4] introduced a deep double Q-network with a multistate vulnerability analysis language (MDDQN) for PT to improve network security. The findings of the experiments demonstrated that as the complexity of the experimental scenarios increased, so did the advantages of the MDDQN Algorithm. Other drawbacks of the MDDQN Algorithm include a certain amount of overestimation and its incapacity to independently scan the structures and get network data.

Chen et al. [5] presented a generative adversarial imitation learning method for PT testing named GAIL-PT, which provides efficient penetration performance with less time and cost. The approach also obtains maximum cumulative rewards. However, the complexity of the GAIL-PT approach was high, which impacted the PT performance.

2.1. Challenges

- Most expert knowledge now gathered is acquired through training in networks within a specific size range [1].
- The minor decline in the covered attack vectors, which would lead to the omission of some complex attack routes that human hackers would use, is detectable by IAPTF. Because of this, the representational structure of the transition data limits the coverage of expert knowledge [2].
- In dynamic cyber security settings, the assumption of a largely fixed environment may not always apply, which could restrict EPPTA's adaptability [3].
- The MDDQN approach has several drawbacks, including a certain amount of overestimation and the inability to autonomously scan the constructions and get network information [4].
- Although the network simulator NASim appears to offer accurate vulnerability assessments and targeted penetration operations in network scenarios, there are still differences between NASim and the real network, and the GAIL approach hasn't been verified in real complex networks. The GAIL-based PT procedure is also extremely complex [5].

2.2. Problem statement

The goal of Penetration Testing (PT) is to simulate cyberattacks on a computer system to find security flaws and evaluate overall protection. Emerging technologies in AI provide various approaches to PT to improve network security. However, there are still some limitations seen in those approaches, like the requirement of more computational time and cost, the inability to handle complex environments, and scalability issues. The PT also involves choosing an action, where some approaches struggle with large spaces because of their need to explore the space efficiently. To resolve these issues, the research introduced an optimized deep CNN-based model with a deep Q learning approach for PT.

3. Adaptive Hunt tuner algorithm – Q-learning-based deep CNN model for Penetration testing

The research aims to design a PT framework to improve the security infrastructure of the environment by learning the different attacks on the network using a deep learning approach named AHT-QCN. At first, a PT environment is simulated using the Shodan search engine and generates the expert knowledge base. The Metasploit and Nmap script engines are employed in this environment. The NSE scripts check the vulnerabilities and associate data with common vulnerabilities and exposures. Similarly, Metasploit analyzes the target network vulnerabilities before hackers can exploit them. For PT, a CVE dataset [16] is used for training the model, where the CVE dataset contains both numerical and text data. Dimensionality reduction techniques, such as Feature Selection (FS), are essential for handling the increasing volume and complexity of data in the digital era [27]. The data are then preprocessed using Text cleaning, stop word removal, and Lemmatization after stimulating the PT environment. Here, the text cleaning removes unwanted characters in the text; the stop word removal process removes the words that occur frequently in the text and the lemmatization process breaks down the words into their root form. Tokenizer is also applied to the data, which breaks down the text into smaller units called tokens. After preprocessing, the data are fed as input to the AHT-QCN model for learning the attacks. In addition, the Q-learning approach is employed in the deep CNN model for training, which aids an agent in determining the optimal action in a given state by improving the cumulative rewards or Q-values over time and leads to successful penetration. The utilization of CNN extracts relevant features and helps to improve the security of the PT environment with optimal decisions. Here, an AHT optimization algorithm is created with the hunting and swarming characteristics of particle swarms and coati respectively, which tunes the classifier's parameters and aids in the model's performance improvement. In this research, the model is tested using the randomly simulated IPs from the PT environment and the developed trained model classifies the attack's defect as normal, partial, and complete. The illustration of the developed PT framework is presented in Fig. 1.

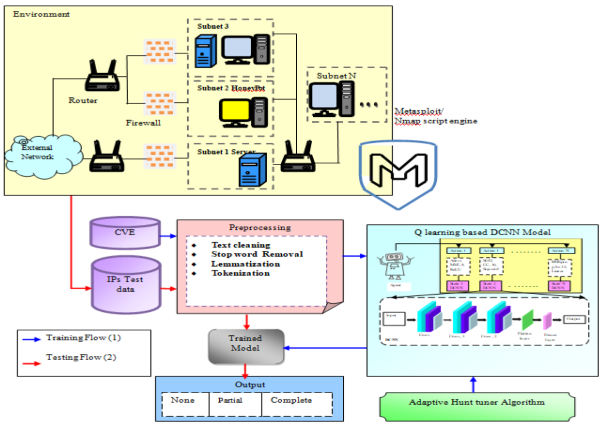


Fig. 1. Illustration of the developed Penetration testing framework using AHT-QCN

3.1. Simulation of Penetration testing environment

A PT environment is a network setup or a controlled system, which is created for simulating cyber attacks. Servers, firewalls, switches, and subnetworks make up the network configuration. Shodan is a search engine that helps penetration testers find devices connected to the internet. It shows information like open ports, operating systems, and services running on each device. This makes it easier to find weak spots in a network and identify security risks. It's a valuable tool for spotting potential problems in online devices. Nmap Script Engine (NSE) and Metasploit framework are the two components employed in the PT environment for efficient PT testing.

- **Nmap Script Engine (NSE).** The NSE enables the users of Nmap to create scripts, carry out network operations, and explore the network. Nmap scripts are used to automate tasks, gather more detailed information about a target, and perform specific actions during a penetration test, such as brute-forcing passwords and exploiting vulnerabilities. It finds all the information about the operating system and software, the IP addresses of distant hosts, the open ports, and vulnerabilities on local and remote computers.
- **Metasploit.** With one of the top penetration test tools, Metasploit is as easy to use as a directory containing a list of vulnerabilities. An open-source program called the Metasploit Framework gives users the tools and infrastructure they need to conduct security audits and penetration tests. The Metasploit framework facilitates the process of discovering and creating new exploits for previously unidentified or undiscovered vulnerabilities in operating systems, applications, and networks.

3.2. Input

The CVE database [20] is utilized in this research for training the developed model and is represented as

(1) $$P=\{P_1,P_2,P_3,\dots,P_n\},$$

where P represents the input CVE database, $\{P_1,P_2,P_3,\dots,P_n\}$ are the data in the data, and n is the total number of data in the dataset.

For testing, random IPs are simulated from the PT simulation environment and the IPs are represented as

(2) $$I = \left\{ I_{p1}, I_{p2}, I_{p3}, ..., I_{pn} \right\},$$

where I contains $\left\{ I_{p1}, I_{p2}, I_{p3}, ..., I_{pn} \right\}$ that are simulated IPs from the PT environment.

3.3. Pre-processing

The input data P is preprocessed using various techniques such as Text cleaning, stop word removal, Lemmatization, and Tokenizer.

- **Text cleaning.** The text cleaning removes irrelevant characters, misspellings, and inconsistent formatting. This reduces the noise in the text data and improves the performance.
- **Stop word removal.** This process removes frequently occurring words from the text to improve the accuracy and efficiency of the approach. Stop words, also known as noise words, are typically articles and pronouns that don't add much value to understanding the text.
- **Lemmatization.** Lemmatization is the process of breaking down words into their most basic form, or lemma. Normalizing various inflected versions of a word is intended to facilitate easier analysis and comparison.
- **Tokenizer.** Tokenizer divides a text into smaller parts, known as tokens which can be engrams, words, integers, or symbols. This procedure helps to extract individual words from the raw text, making it easier to read, which makes it a crucial component of text data analysis.

3.4. Adaptive Hunt Tuner Algorithm – Q-learning based deep CNN

The Q-learning-based deep CNN model is employed for efficient PT, where the Q-learning utilizes the RL concept. It begins by defining the Q-values. Further, the model makes use of this Q-value to maximize rewards and enhance the learning agent's performance. Unlike standard RL, Q-learning emphasizes the quality of an action instead of solely relying on the reward of the subsequent state [17]. Q-values are provided in the form of a set of states and actions $q(\delta, \alpha)$, which shows how to choose an action α that is optimal for a particular state δ. The following equation is utilized to calculate $q(\delta, \alpha)$,

(3) $$q_{\tau}(\delta, \alpha) = q_{\tau-1}(\delta, \alpha) + \alpha\left(r(\delta, \alpha) + \lambda \max_{\alpha'} q(\delta', \alpha') - q_{\tau-1}(\delta, \alpha)\right),$$

with δ representing the agent's current state, α current action taken following a specific policy, δ' representing the next state the agent must transit to, α' representing the best course of action based on the current estimation of Q-value, $r(\delta, \alpha)$ representing the current reward calculated from the provided environment in response to the current action, λ representing the discount factor, and α controlling the updating of $q(\delta, \alpha)$. Every time step, the Q-value is updated using the equation above. Throughout its existence, the agent begins at a location known as the start state. It then transitions between states. The first is the selected action. The other one is the specific setting in which the agent works. Every time it moves, the agent acts based on a state, determines its reward from the surroundings, and then shifts to a new state. There will not be any further movements possible if the agent ever enters one of the ending states.

The Q-learning stores the state-action pair in a Q-table during the training process. Here Q-learning finds the best sequence of action to improve the deep CNN's performance, where the hyperparameters of the CNN like default optimizers, losses, batch size, and activation functions are selected based on the actions from the Q-learning algorithm, where the best state is selected. The Q-learning is applied on the first layer of the deep CNN's convolutional layer, and the developed AHT-QCN method consists of a convolutional layer, max pooling, flattened, and dense layer. Here the convolutional layer extracts the efficient features and the pooling layers reduce the dimensions of the data with the size $\left[N \times 356 \times 1 \times 16\right]$. A dropout is employed to process the max pooling output that prevents overfitting and the output of the max-pooling layer is then flattened with a size of $\left[N \times 22,784\right]$ and connected to the dense layer and classifies the attacks efficiently as none, partial, and complete. The training of the model is performed by Q-learning, which then improves the model's performance by determining the optimal state for the deep CNN model. The developed AHT optimization algorithm is employed in the first layer of the deep CNN through which the classifier's parameters are tuned for enhanced performance with less computational time. The systematic representation of the AHT-QCN model is presented in Fig. 2.

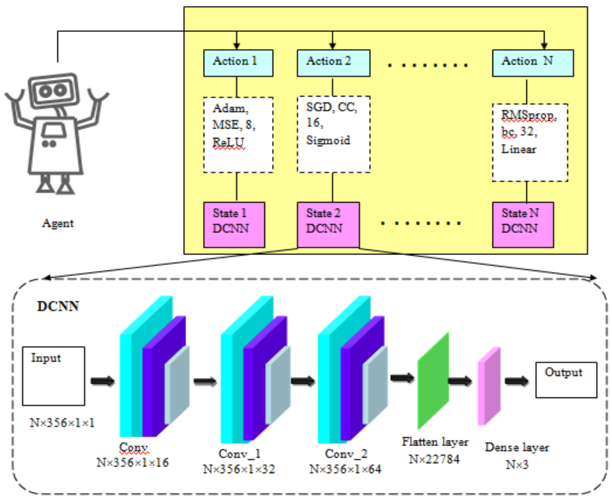


Fig. 2. The systematic representation of the AHT-QCN model

3.4.1. Adaptive Hunt Tuner Algorithm (AHT Algorithm)

The AHT algorithm is designed from the hunting traits of the coati [18] and bird's swarm [19] characteristics to tune the developed model for PT with improved performance. The AHT algorithms reduce the risk of local optima and slow convergence speed of the coati algorithm by adapting the high velocity of the birds with the coati's hunting traits, where the high velocity aids in the exploration of the search space and improves the global search capabilities.

Inspiration. The hunting and attacking traits are inspired by the coatis, which belong to the Procyonidae family. In addition, the coatis exhibit cunning behavior when hunting and avoiding predators as well as when hunting and killing their prey. The simulation of those coati's real behaviors served as a major creative inspiration for the coati optimization approach. The coati's distinct adaptability, which is based on cognitive concepts and makes it perfect for dynamic optimization problems, is what motivates the decision to use it. The particle swarm optimization is made up of particles that are propelled by natural swarms and communicate through evolutionary algorithms. PSO incorporates social and self-experiences and its potential solution is shown as a particle. It reaches a global optimum by moving toward a promising area and gathering flying particles (varying solutions) in a search area (current and potential solutions). The phases involved in determining the best solution using the AHT algorithm are detailed below,

Initialization. The solutions of the AHT initialized as *C*, where the weight and bias are the tunable parameters of the deep CNN and are represented as

$$C \in [w, \beta], \tag{4}$$

$$C = \{C_1, C_2, ..., C_j, ..., C_m\}, \tag{5}$$

where *m* are the total solutions in the search space and C_j is the *j*-th solution.

Fitness evaluation. The fitness of the algorithm is determined based on the accuracy and the solution with high accuracy is determined as the best solution. The fitness is expressed as,

$$f_{\text{fit}}(C) = \max(\text{accuracy}). \tag{6}$$

Solution update. The solutions of the algorithm are updated with iterations based on the following phases.

Case 1: $A \le 1$, Exploration phase. If the search probability (*A*) is less than or equal to one, some of the solutions scare and attack the prey for hunting, and the remaining solutions wait for the prey to fall. Here the best solution is updated based on the position of the prey and the leading solutions using the following equation,

$$C^{\tau+1} = C^{\tau} + \gamma\left(C_{\text{best}} - KC^{\tau}\right), \tag{7}$$

where $C^{\tau+1}$ is the solution's new position, C_{best} denotes the prey's best position, the C^{τ} is the solution's current position, K is the integer and is randomly selection from the interval [0, 2], and γ is the random number in the interval [1, 2].

Case 2: $A > 1$, Exploitation Phase. If the search probability is greater than one, the solution's position is updated based on equation (6). Here, when a solution attacks the prey, the prey escapes from the location and then the solution moves to the new position closer to the current position and hunts the prey. When the solution moves towards the new position, the solution may fall into local optima and suffer from slow convergence. Therefore, a high velocity is applied to the solution

(8) $$C^{\tau+1}=\frac{1}{2}\left[2C^{\tau}+\left(1-2\gamma_1\right)\left[C_{\mathrm{p}}^{\tau}+\gamma_2\left(C_{\mathrm{g}}^{\tau}-C_{\mathrm{p}}^{\tau}\right)\right]+v^{\tau}+2\gamma_3\left(v_{\mathrm{p}}^{\tau}-C^{\tau}\right)+2\gamma_4\left(v_{\mathrm{g}}^{\tau}-C^{\tau}\right)\right],$$

where C_{p}^{τ} and v_{p}^{τ} are the personal best solution and the velocity of that solution at the time τ. Likewise, C_{g}^{τ} and v_{g}^{τ} are the global best position and its velocity at the time τ.

Here,

(9) $$\gamma_1=\frac{\mu-i}{\mu},$$

where μ and i are the upper bound and lower bound at the time τ.

(10) $$\gamma_2=1-\frac{\tau}{\tau_{\max}},$$

(11) $$\gamma_3=\frac{F\left(C_{\mathrm{best}}\right)-F\left(C\tau\right)}{F\left(C_{\mathrm{best}}\right)},$$

(12) $$\gamma_4=\frac{\gamma_1+\gamma_2+\gamma_3}{3},$$

where, $\gamma, \gamma_1, \gamma_2, \gamma_3$, and γ_4 are the random parameters.

Re-evaluation of the fitness measure. To declare the best solution, the fitness values are re-evaluated after updating the solutions.

Termination. When the condition $\tau<\tau_{\max}$ is met, the iterations end, and the best solution is declared. The flowchart of the AHT Algorithm is presented in Fig. 3 and the pseudo code of the algorithm is depicted in Algorithm 1.

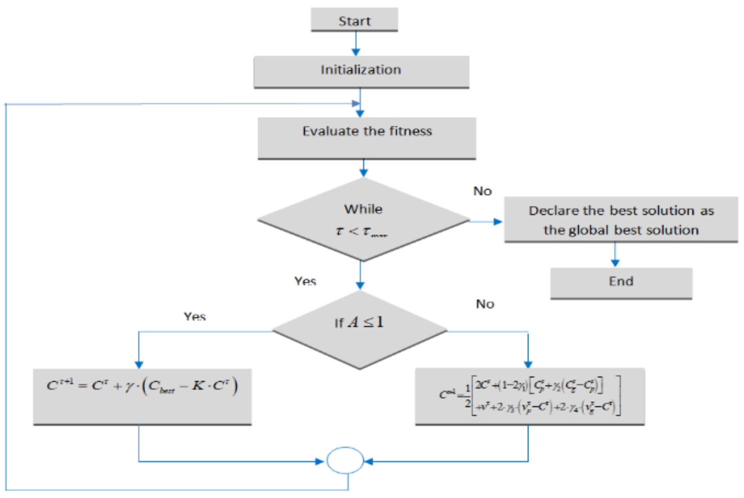

Start
Initialization
Evaluate the fitness
While $\tau<\tau_{\max}$
No
Declare the best solution as the global best solution
End
Yes
If $A\le 1$
Yes
No
$C^{\tau+1}=C^{\tau}+\gamma\cdot\left(C_{best}-K\cdot C^{\tau}\right)$
$C^{\tau+1}=\frac{1}{2}\left[\begin{array}{l}2C^{\tau}+(1-2\gamma_1)\left[C_p^{\tau}+\gamma_2(C_g^{\tau}-C_p^{\tau})\right]\\+v^{\tau}+2\cdot\gamma_3\cdot(v_p^{\tau}-C^{\tau})+2\cdot\gamma_4\cdot(v_g^{\tau}-C^{\tau})\end{array}\right]$

Fig. 3. The flowchart of the AHT Algorithm

Algorithm 1. Pseudo code of AHT Algorithm

```
Step 1.   Begin
Step 2.   Initialization
Step 3.   Evaluate the fitness
Step 4.   While $\tau < \tau_{\max}$
Step 5.   do
Step 6.       if $A \le 1$
Step 7.           $C^{\tau+1} = C^{\tau} + \gamma\left(C_{\text{best}} - KC^{\tau}\right)$
Step 8.       else
Step 9.           $C^{\tau+1} = \frac{1}{2}\left[\begin{array}{l} 2C^{\tau} + (1-2\gamma_1)\left[C_{\text{p}}^{\tau} + \gamma_2\left(C_{\text{g}}^{\tau} - C_{\text{p}}^{\tau}\right)\right] + \\ + v^{\tau} + 2\gamma_3\left(v_{\text{p}}^{\tau} - C^{\tau}\right) + 2\gamma_4\left(v_{\text{g}}^{\tau} - C^{\tau}\right). \end{array}\right]$
Step 10.  Return to fitness evaluation
Step 11.  Declare the best solution
Step 12.  Terminate the process
Step 13.  End
```

4. Result and discussion

In this section, the experimental results of the proposed AHT-QCN model in PT are detailed and discussed.

4.1. Experimental setup

The AHT-QCN model is developed in this research for PT and the experiment is implemented on Python in Windows 10 with 16 GB RAM.

4.2. Dataset description

CVE dataset [16]: The proposed research makes use of a common vulnerabilities and exposure dataset and is obtained from the National Institute of Standard Technology (NIST). The CVE dataset is a useful resource for PT since it provides details on cyber security threats, vulnerabilities, and exposures. The CVE dataset contains 89,660 unique values.

4.3. Confusion matrix

The confusion matrix of the proposed AHT-QCN model is presented in Fig. 4, where the actual values of the dataset are compared with the prediction values to determine the performance of the model. It identifies if a model is confusing two classes. Here 0 represents the normal, 1 denotes the partially affected, and 2 represents the completely affected.

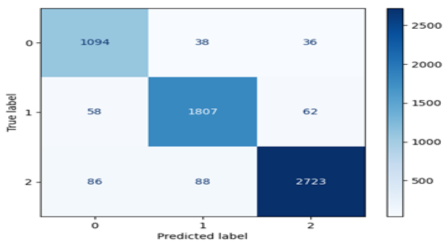


Fig. 4. Confusion matrix of the AHT-QCN model

4.4. Analysis of success rate and rewards

Fig. 5 depicts the success rate and reward rate analysis of the AHT-QCN model based on the number of episodes on various epochs. The AHT-QCN model shows a success rate of 0.87 at epoch 100 and 0.89 for 100 episodes at epoch 500. Similarly, the accumulated rewards of the AHT-QCN model are 4193 at epoch 100 and 4293 at epoch 500 for 100 episodes. These results indicate that the success rate and the accumulated rewards increase with the increase in epochs and no of episodes.

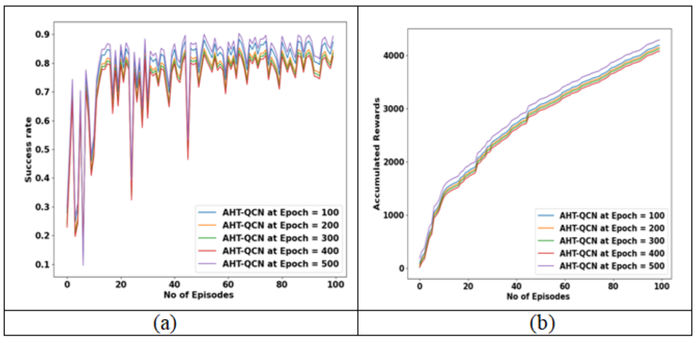


Fig. 5. Analysis of success rate (a), and reward rate based on number of episodes (b)

4.5. Convergence analysis

Fig. 6 depicts the convergence analysis of the developed model based on the no of episodes and epochs, where the accuracy of the developed model is 0.89, 0.85, 0.84, 0.83, and 0.90 for epochs 100, 200, 300, 400, and 500 respectively for 100 episodes.

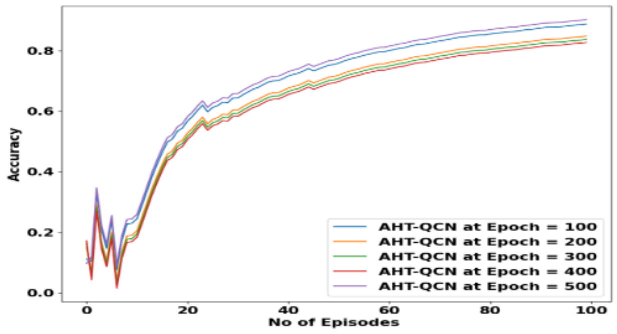
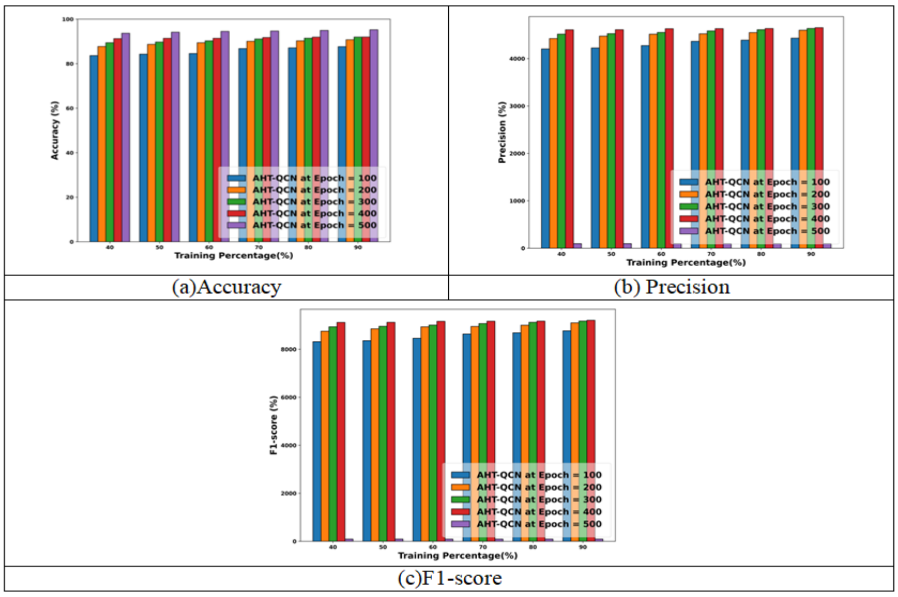


Fig. 6. Convergence of the AHT-QCN model

4.6. Performance analysis

The performance of the developed AHT-QCN model is analyzed based on Training Percentage (TP) and in terms of accuracy, precision, and F1-score. The AHT-QCN model's performance evaluation is presented in Fig. 7 based on TP for epochs 100, 200, 300, 400, and 500. The accuracy of the AHT-QCN is 87.66%, 90.76%, 91.945, 91.97%, and 95.24% at TP 90 for the above-mentioned epochs respectively, indicating that the accuracy improves with an increase in epochs. The AHT-QCN model's precision is 97.65% at TP 90 for 500 epochs. Similarly for the F1-score, the AHT-QCN achieves 93.80% for TP 90 with epoch 500.



(a)Accuracy

(b) Precision

(c)F1-score

Fig. 7. Performance evaluation of AHT-QCN model

4.7. Comparative analysis

KNN [20], CatBoost [21], Xgboost [22], Neural Network (NN) [23], LSTM [24], deep CNN [25], Q-learning ensemble deep CNN, Particle Swarm Optimization-QCN (PSO-QCN), and Coati Optimization Algorithm-QCN (COA-QCN) are compared with the developed AHT-QCN based on TP. The comparative evaluation of the developed model is depicted in Fig. 8. The AHT-QCN model achieved an accuracy of 95.24% for TP 90, which shows an improvement of 1.66% over deep CNN, 239% over LSTM, 2.73% over NN, 11.43% over Xgboost, 21.83% over CatBoost, 28.74% over KNN. Similarly, the AHT-QCN model shows a precision of 97.65%, which is improved by 0.30% over Q-learning ensemble deep CNN, and 0.104% over COA-QCN. Similarly for F1-score, the developed approach attains 93.80%, which is improved by 1.25% over PSO-QCN. The results indicate that the developed AHT-QCN shows superior performance than other methods in PT.

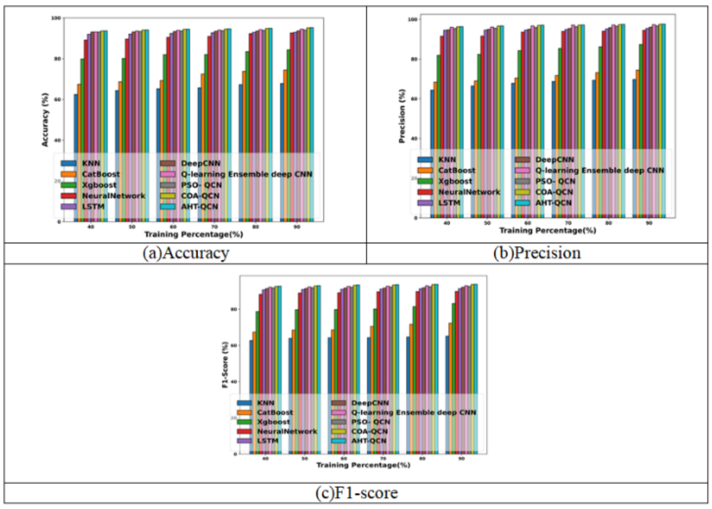


Fig. 8. Comparative evaluation of AHT-QCN model

4.8. Comparative discussion

Table 1 presents the results of the developed AHT-QCN model compared with the existing methods. Even though, though the existing approaches show considerable performance, the approaches suffer from various limitations, where the CatBoost is sensitive to hyper parameters and the Neural networks are computationally expensive. The LSTM suffers from over fitting issues and the deep CNN requires large labeled datasets. To resolve these issues, the AHT-QCN is developed in this research, where the incorporation of AHT improves the performance of the model by tuning its parameters and the Q-learning shows efficient decision-making performance even in complex environments.

Table 1. Comparative discussion

No	Methods	Accuracy (%)	Precision (%)	F1-score %)
1	KNN	67.87	69.75	65.19
2	CatBoost	74.45	74.45	72.39
3	xgboost	84.36	87.32	83.18
4	Neural Network	92.64	94.51	89.87
5	LSTM	92.96	95.45	91.42
6	DeepCNN	93.66	96.09	92.22
7	Q-learning Ensemble deep CNN	94.54	97.36	93.05
8	PSO-QCN	94.10	96.72	92.63
9	COA-QCN	95.16	97.55	93.69
10	Proposed AHT-QCN	95.25	97.66	93.81

5. Conclusion

One essential way to assess a network system’s level of security is through PT. Further, PT attack route planning is crucial because it mimics the actions of an attacker to find vulnerabilities, lower possible losses, and continuously enhance security measures. In this research, an AHT-QCN model is developed for PT with improved performance, with the utilization of deep CNN and Q-learning. The AHT algorithm fine-tunes the deep CNN's parameters and improves its performance in PT. AHT-QCN model's efficiency is increased by the Q-learning that it uses to provide optimal policy learning for decision-making. The AHT algorithm also reduces the computational time of the model due to its fast convergence. The developed model shows superior results compared to other existing approaches, attaining accuracy of 95.25%, 97.66% precision, and 93.81% F1-score. In the future, the large dataset can be used in the model with additional network topologies to enhance the model's stability, and efficient optimization techniques with ensemble techniques can also be implemented in the future for more accurate results.

References

1. G h a n e m, M. C., T. M. C h e n, E. G. N e p o m u c e n o. Hierarchical Reinforcement Learning for Efficient and Effective Automated Penetration Testing of Large Networks. – Journal of Intelligent Information Systems, Vol. **60**, 2023, No 2, pp. 281-303.
2. W a n g, Y., Y. L i, X. X i o n g, J. Z h a n g, Q. Y a o, C. S h e n. DQfD-AIPT: An Intelligent Penetration Testing Framework Incorporating Expert Demonstration Data. – Security and Communication Networks, 2023, No 1, 5834434.
3. L i, Z., Q. Z h a n g, G. Y a n g. EPPTA: Efficient Partially Observable Reinforcement Learning Agent for Penetration Testing Applications. – Engineering Reports, 2023, e12818.
4. Y i, J., X. L i u. Deep Reinforcement Learning for Intelligent Penetration Testing Path Design. – Applied Sciences, Vol. **13**, 2023, No 16, 9467.
5. C h e n, J., S. H u, H. Z h e n g, C. X i n g, G. Z h a n g. GAIL-PT: An Intelligent Penetration Testing Framework with Generative Adversarial Imitation Learning. – Computers & Security, Vol. **126**, 2023, 103055.
6. A r k i n, B., S. S t e n d e r, G. M c G r a w. Software Penetration Testing. – IEEE Security & Privacy, Vol. **3**, 2005, No 1, pp. 84-87.
7. K a u r, G., N. K a u r. Penetration Testing – Reconnaissance with NMAP Tool. – International Journal of Advanced Research in Computer Science, Vol. **8**, 2017, No 3, pp. 844-846.
8. K a u s h i k, M., G. O j h a. Attack Penetration System for SQL Injection. – International Journal of Advanced Computer Research, Vol. **4**, 2014, No 2, p. 724.
9. H a e n i, R. E. Firewall Penetration Testing. – In: Technical Report. The George Washington University Cyberspace Policy Institute, 2033 K St, Suite. Vol. **340**. 1997.
10. P h o n g, C. T., W. Q. Y a n. An Overview of Penetration Testing. – International Journal of Digital Crime and Forensics (IJDCF), Vol. **6**, 2014, No 4, pp. 50-74.
11. H e n r y, K. Penetration Testing: Protecting Networks and Systems. – IT Governance Publishing, 2012.
12. H a f i z, A. M. A Survey of Deep q-Networks Used for Reinforcement Learning: State of the Art. – In: Proc. of Intelligent Communication Technologies and Virtual Mobile Networks: (ICICV'22), 2022, pp. 393-402.
13. C h a u d h a r y, S., A. O'B r i e n, S. X u. Automated Post-Breach Penetration Testing through Reinforcement Learning. – In: Proc. of IEEE Conference on Communications and Network Security (CNS'20), 2020, pp. 1-2.
14. Z e n n a r o, F. M., L. E r d ő d i. Modelling Penetration Testing with Reinforcement Learning Using Capture-the-Flag Challenges: Trade-Offs between Model-Free Learning and a Priori Knowledge. – IET Information Security, Vol. **17**, 2023, No 3, pp. 441-457.
15. C h e n, X., Y. M. M u, P. L u o, S. L i, J. C h e n. Flow-Based Recurrent Belief State Learning for Pomdps. – In: Proc. of International Conference on Machine Learning (PMLR'22), June 2022, pp. 3444-3468.

16. CVEdataset, on July 2024.
https://www.kaggle.com/datasets/andrewkronser/cve-common-vulnerabilities-and-exposures
17. H u, Z., R. B e u r a n, Y. T a n. Automated Penetration Testing Using Deep Reinforcement Learning. – In: Proc. of IEEE European Symposium on Security and Privacy Workshops (EuroS&PW'20), September 2020, pp. 2-10.
18. D e h g h a n i, M., Z. M o n t a z e r i, E. T r o j o v s k á, P. T r o j o v s k ý. Coati Optimization Algorithm: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems. – Knowledge-Based Systems, Vol. **259**, 2023, 110011.
19. W a n g, D., D. T a n, L. L i u. Particle Swarm Optimization Algorithm: An Overview. – Soft Computing, Vol. **22**, 2018, No 2, pp. 387-408.
20. L i, J., S. W a n g, H. Z h a n g, A. Z h o u. A Multi-Objective Evolutionary Algorithm Based on KNN-Graph for Traffic Network Attack. – Electronics, Vol. **9**, 2020, No 10, p. 1589.
21. N a k h o d c h i, S. A Framework Based on Bag of Feature and CatBoost for Attack Detection and Attribution in Industrial Control Systems. – Doctoral Dissertation, University of Guelph, 2021.
22. X u e, W., T. W u. Active Learning-Based XGBoost for Cyber Physical System against Generic AC False Data Injection Attacks. – IEEE Access, Vol. **8**, 2020, pp. 144575-144584.
23. H u a n g, S., N. P a p e r n o t, I. G o o d f e l l o w, Y. D u a n, P. A b b e e l. Adversarial Attacks on Neural Network Policies. – arXiv preprint arXiv:1702.02284, 2017.
24. H o s s a i n, M. D., H. I n o u e, H. O c h i a i, D. F a l l, Y. K a d o b a y a s h i. LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications. – IEEE Access, Vol. **8**, 2020, pp. 185489-185502.
25. M a r r a, F., D. G r a g n a n i e l l o, L. V e r d o l i v a. On the Vulnerability of Deep Learning to Adversarial Attacks for Camera Model Identification. – Signal Processing: Image Communication, Vol. **65**, 2018, pp. 240-248.
26. S a r k e r, I. HDeep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective. – SN Computer Science, Vol. **2**, 2021, No 3, p. 154.
27. V e n k a t e s h, B., J. A n u r a d h a. A Review of Feature Selection and Its Methods in Cybernetics and Information Technologies. – Cybernetics and Information Technologies, Vol. **19**, 2019, No 1, pp. 3-26.
28. L a z a r o v, A. D. Mathematical Modelling of Malware Intrusion in Computer Networks. – Cybernetics and Information Technologies, Vol. **22**, 2022, No 3, pp. 29-47.

Received: 05.08.2024; Accepted: 21.08.2024 (fast track)