Penetration Testing Framework using the Q Learning Ensemble Deep CNN Discriminator Framework

Dipali Nilesh Railkar, Dr. Shubhalaxmi Joshi
Dr. Vishwanath Karad, MIT World Peace University, Pune, Maharashtra, India

***Abstract*—Penetration testing (PT) serves as an effective tool for examining networks and identifying vulnerabilities by simulating a hacker's attack to uncover valuable information, such as details about the host's operating and database systems. Strong penetration testing is crucial for assessing system vulnerabilities in the constantly changing world of cyber security. Existing methods often struggle with adapting to dynamic threats, providing limited automation, and lacking the ability to discern subtle security weaknesses. In comparison to manual PT, intelligent PT has gained widespread popularity due to its efficiency, resulting in reduced time consumption and lower labor costs. Considering this, the effective penetration testing framework is developed using prairie natural swarm (PNS) optimized Q-learning ensemble deep CNN. Initially, the penetration testing environment (Shodan search engine) is simulated, and along with that expert knowledge base is also generated. Subsequently, the Nmap script engine and Metasploit are deployed, providing robust tools for network investigation and vulnerability assessment. The system state is then relayed to the Q-learning ensemble deep convolutional neural network (Q-learning ensemble deep CNN) classifier. This unique ensemble combines the strengths of Q-learning and deep CNNs, enabling optimal policy learning for decision-making. The prairie natural swarm optimization algorithm is developed through the hybridization of coyote and particle swarm characteristics to fine-tune classifier parameters, enhancing performance. Additionally, the discriminator is trained to maximize standard action rewards while minimizing discounted action rewards, distinguishing valuable from less valuable information. By evaluating the advantage function, successful penetration likelihood is determined, informing situational decision-making through the Q-learning ensemble deep CNN classifier. Accuracy, sensitivity, and specificity as well as the proposed PNS-optimized Q-learning ensemble deep model are used to evaluate the output. In comparison to other approaches currently in use, CNN achieves values of 94.54%, 94.40%, 94.90% for TP, 94.64%, 94.69%, and 94.52% for k-fold.**

Keywords*—Penetration testing; Q-learning; ensemble deep CNN; prairie natural swarm optimization; Nmap script engine

I. Introduction

In today's interconnected world, the significance of robust cybersecurity measures cannot be overstated. With the relentless advancement of technology, the frequency and sophistication of cyber threats have risen to unprecedented levels. This digital age has ushered in a landscape where individuals, businesses, and governments are all interconnected through complex networks, presenting both unparalleled opportunities and considerable risks [1]. The escalating frequency of cyber threats underscores the gravity of the situation. Malicious actors are exploiting vulnerabilities in software, networks, and infrastructure at an alarming rate, leading to data breaches, financial losses, and significant disruptions. These threats are not confined to specific sectors; they affect organizations of all sizes across industries, as well as individuals who rely on technology for daily tasks. The sophistication of modern cyber threats adds another layer of complexity. Hackers are using advanced tactics, techniques, and procedures that can bypass traditional security measures [2]. From advanced malware to social engineering and zero-day vulnerabilities, cyber threats have become multifaceted and difficult to predict. This calls for cyber security measures that can adapt and respond to evolving attack vectors [3]. As our lives become increasingly digitized, from critical infrastructure to personal devices, the potential consequences of a successful cyber attack become more severe. Breaches can compromise sensitive data, disrupt essential services, and even pose risks to public safety. This underscores the urgent need for robust cyber security measures that can effectively counter these threats.

The threat landscape in the realm of cyber security is dynamic and ever-evolving. It is a landscape characterized by constant change, as cybercriminals consistently innovate and develop new techniques to exploit vulnerabilities and breach systems [4]. This dynamic nature of the threat landscape presents a significant challenge to individuals, organizations, and institutions responsible for safeguarding digital assets and sensitive information. Cybercriminals exhibit remarkable adaptability, constantly refining their tactics, techniques, and procedures (TTPs) to stay one step ahead of security measures [5]. Just as security professionals identify and mitigate one vulnerability cybercriminals quickly shift their focus to discover new avenues of attack [6][7]. This agility allows them to circumvent traditional security mechanisms and exploit unforeseen weaknesses [8]. ML and AI have transformed cyber security by automating complex tasks, analyzing vast data to detect anomalies, and adapting to evolving threats. In penetration testing, ML and AI automate threat detection, identifying vulnerabilities, and simulating attacks. They enhance response by swiftly isolating threats, and minimizing damage. This automation accelerates testing, enabling security professionals to focus on strategic analysis [9]. By enhancing threat detection accuracy, optimizing resource allocation, and reducing false positives, ML and AI elevate penetration testing's efficiency and effectiveness, fortifying cyber security in an increasingly intricate threat landscape.

The fusion of Q Learning, a reinforcement learning technique, with Deep CNNs forms a powerful strategy to

tackle intricate, ever-changing decision-making tasks across diverse domains, spanning from robotics to gaming [10]. This combination leverages the strengths of both techniques to maximize decision-making accuracy in high-dimensional state spaces [11]. Q Learning, as a model-free reinforcement learning method, operates through trial-and-error, learning optimal actions by maximizing long-term rewards. It's particularly well-suited for sequential decision-making tasks in dynamic environments. However, it often faces challenges when dealing with high-dimensional or intricate state representations, which are common in applications such as image-based gaming or robotic perception [12]. This is where CNNs come into play. By using convolutional layers to recognize hierarchical patterns, these neural networks excel in processing complicated data, such as videos or images. They can extract meaningful features from raw sensory input, reducing the dimensionality of the state space and enabling more effective decision-making. CNNs also enable end-to-end learning, allowing the agent to autonomously discover relevant features for its task [13]. CNNs can analyze complex data and close the gap between perception and action when used in conjunction with Q Learning. The combination empowers reinforcement learning agents to operate efficiently in scenarios with high-dimensional state spaces [14]. For instance, in gaming, an agent can learn to play complex video games directly from pixel inputs, making it more versatile and adaptive. In robotics, it enables intelligent machines to navigate and interact with their environment, making them suitable for real-world applications [15]. The integration of Q Learning with CNNs represents a promising approach for enhancing decision-making accuracy in dynamic, high-dimensional environments. This combination of deep learning and reinforcement learning methods has the potential to completely transform a variety of applications by enhancing their intelligence, adaptability, and capacity for processing complex data, bringing in a new era of machine learning-driven solutions.

The main aim of the research is to develop a prairie natural swarm-optimized Q-learning ensemble deep CNN for penetration testing. The initial step involves simulating a penetration testing environment using the Shodan search engine, alongside the generation of an expert knowledge base. Subsequently, the deployment of powerful tools, namely the Nmap script engine and Metasploit, facilitates the comprehensive investigation and assessment of network vulnerabilities. The state of the system is then conveyed to the Q-learning ensemble deep CNN classifier, which uniquely amalgamates the capabilities of Q-learning and deep CNNs to enable the acquisition of optimal decision-making policies. The optimization process involves the development of a prairie natural swarm optimization algorithm, achieved through the fusion of coyote and particle swarm characteristics, resulting in the refinement of classifier parameters for enhanced performance. Additionally, the discriminator is trained to maximize standard action rewards while minimizing discounted action rewards, discerning between valuable and less valuable data. The evaluation of the advantage function aids in determining the likelihood of successful penetrations, subsequently guiding situation-based decisions through the Q-learning ensemble deep CNN classifier. The contributions of the research are as follows.

Prairie natural swarm optimization: The prairie natural swarm optimization (PNS) is developed through the hybridization of coyote and particle swarm algorithms. In the coyote algorithm, the velocity and position are not interpreted so it faces limited capability to explore the search space effectively and slower convergence to optimal solutions. Considering this, the particle swarm algorithm velocity is merged with a coyote for faster convergence and balanced exploration and exploitation.

PNS-optimized Q-learning ensemble deep CNN: The PNS-optimized Q-learning ensemble deep CNN classifier is a combination of two powerful techniques in artificial intelligence such as Q-learning and deep CNNs. The advantage of an ensemble Q-learning and deep CNN classifier in penetration testing is its capacity to enhance the accuracy of identifying vulnerabilities and security weaknesses. The PNS algorithm helps in the fine-tuning of the parameters inside the classifier, which helps in enhancing the performance of the classifier.

The manuscript follows a structured organization, commencing with Section II, which delves into the comprehensive reviews of penetration testing. Moving on to Section III, this section elaborates on the proposed methodology for conducting penetration testing and introduces the mathematical equation underpinning the PNS algorithm. Section IV is dedicated to a detailed examination of the empirical results and overarching conclusions drawn from the research findings. Finally, in Section V, the manuscript wraps up by presenting the ultimate thoughts and conclusions that emerge from the research work.

II. Literature Review

The vulnerability scanning and penetration testing with respect to network security reviews are as follows: A black-box Reinforcement Learning-based framework was presented by Wei Song et al. [13] to provide Adversarial Examples (AEs) for PE threat classifiers and AV engines. Although this approach achieved notably higher evasion rates and a more effective search for successful adversarial patterns, it may necessitate substantial computational resources and time to optimize the generation process. Soheil Malekshah et al. [4] introduced a deep reinforcement learning approach for identifying optimal strategies to adjust power flow when network reliability diminishes. While this method considered uncertainties and variability associated with distributed generation, providing a more precise representation of network performance, it introduced some complex challenges. A digital twin-powered IIoT architecture was introduced by Wei Yang et al. [15], in which the characteristics of industrial devices are captured for real-time processing and intelligent choice-making. This method facilitated smoother and more effective collaborative learning, contributing to enhanced overall training accuracy. However, it may require specialized expertise in both industrial processes and advanced machine learning techniques. Mohsen Ahmadi et al. [1] presented DQRE-SCnet, a Deep-Q-Reinforcement Learning Ensemble integrated with Spectral Clustering, aimed at selectively

sharing data among nodes in Federated Learning. This approach improved privacy protection and efficiency, yet it grappled with overfitting challenges. LSTM-EVI, a deep learning-based penetration testing system specially created for scanning assaults within a smart airport-based test bed, was introduced by Nickolaos Koroniotis et al. [12]. It outperformed its peer techniques and effectively-identified vulnerabilities in systems. Nonetheless, it exhibited computational complexity. A smart penetration testing framework that included expert demonstration data was introduced by Yongjie Wang et al. [8]. This approach successfully mitigated overfitting concerns and improved the efficiency of penetration testing. However, it demanded significant computational resources and expertise in machine learning. Yang Li et al. [9] introduced an enhanced network graph model for penetration testing, which seamlessly integrated pertinent security attributes into the process. This intelligent penetration testing method leveraged reinforcement learning and social engineering factors. Yet, it entailed complexity and resource-intensiveness, necessitating expertise in both penetration testing and machine learning. An automated penetration testing methodology designed to find the most common weaknesses in IoT devices used in smart homes was presented by Rohit Akhilesh et al. [2]. This method reduced the time and effort required for penetration testing compared to manual approaches, enhancing efficiency. However, it confronted overfitting issues.

The review on penetration testing in network security highlights various approaches, each with strengths and limitations. Common challenges include the need for significant computational resources and time, complexity in dealing with uncertainties and overfitting, resource intensiveness demanding specialized expertise, and struggling with overfitting issues. To address these limitations, a novel PSN-optimized Q-learning ensemble deep CNN framework is developed in this research that integrates multiple techniques like reinforcement learning, deep learning, and expert demonstration data while optimizing efficiency, enhancing robustness, and improving usability. This approach aims to advance the field of vulnerability scanning and penetration testing by mitigating drawbacks, improving the effectiveness of cyber security measures in network environments, and facilitating practical implementation in real-world scenarios.

A. *Challenges*

- Combining Q-learning, deep CNNs, and a discriminator framework presents integration challenges, requiring the harmonization of these diverse components for effective operation.
- Handling intricate data sources in penetration testing, like network traffic, may lead to data preprocessing and feature extraction challenges for deep CNNs.
- The discriminator requires a robust dataset of real-world attacks, which may be limited and pose challenges in creating a representative knowledge base.
- Optimizing parameters for Q-learning, deep CNNs, and the discriminator, including learning rates and network architectures, presents challenges to achieving optimal performance.
- Ensuring the framework's scalability to accommodate various network sizes and complexities while maintaining efficient decision-making can be challenging.

III. Proposed Methodology for Penetration Testing

Penetration testing also referred to as pen testing, is a cyber security procedure that involves simulating actual assaults on computer networks, applications, or systems in order to find security flaws and vulnerabilities. The objective of penetration testing is to proactively assess the security measures of an organization's IT infrastructure and applications, with the goal of uncovering potential weaknesses before malicious attackers can exploit them. Initially, the penetration testing environment (Shodan search engine) is simulated and along with that expert knowledge base is generated. A CVE dataset is utilized in this research for penetration testing. After simulating the penetration testing environment, the Nmap script engine and Metasploit are employed. The Nmap script engine serves as a penetration scanning framework within Nmap, a robust tool for investigating and evaluating networks. NSE empowers users to develop and deploy scripts that automate a range of *tasks* pivotal to penetration testing. Similarly, Metasploit is a widely-used penetration testing framework that helps cyber security professionals and ethical hackers identify vulnerabilities in computer systems, networks, and applications. It provides a range of tools and resources for assessing and exploiting security weaknesses, as well as testing the effectiveness of defense mechanisms and security controls. A deep CNN known as the Q learning ensemble is formed by combining the power of deep learning with reinforcement learning. Deep CNNs receive the state (current circumstance) as input and provide predictions for each possible action. To enable an agent to acquire the best policy for making decisions, Q-learning must be enabled. A model-free reinforcement learning algorithm called Q-learning is used by the agent to learn the optimal policy for making decisions in a given state. It helps the agent to determine the best course of action by maximizing cumulative rewards over time. aids the agent in determining the optimal course of action to pursue in a particular state in order to optimize cumulative rewards over time. Here, Q learning is used to guide the agent in choosing actions that lead to successful penetration by mapping the states to action and optimizing the Q values which represent the predicted cumulative rewards. The Deep CNNs are efficient in handling complex data and extracting relevant features, making them suitable for analyzing the diverse aspects of penetration testing environments and decisions. The optimized Q-learning ensemble deep CNN classifier is a combination of two powerful techniques in artificial intelligence, Q-learning and deep CNNs. In reinforcement learning problems, this hybrid technique is applied when the agent has to learn an optimal policy for making decisions. An agent engages with its surroundings in reinforcement learning, and it is rewarded for its behaviors. The agent aims to learn a policy that maximizes the predicted cumulative reward, or Q-value, by mapping states to actions. The predicted cumulative reward if the agent begins in a state, performs a certain action, and then proceeds with further decisions in accordance with its policy is represented by the Q-value of a state-action pair. Here, the standard

hybridization of the coyote and particle swarm characteristics leads to the development of the prairie natural swarm optimization. These hybridized characteristics help in the fine-tuning of the parameters inside the classifier, which helps in enhancing the performance of the classifier. Simultaneously, the discriminator receives the expert knowledge base and the data from the penetration testing environment. The discriminator plays a crucial role in training the agent by providing feedback on the quality of actions taken. A discounted reward is obtained by maximizing the typical action reward and reducing the action reward output, hence providing training for the discriminator. This feedback loop enables the agent to refine its decision-making process and improve its performance over time. The discounted reward provides the less valuable information and the q value provides the efficient information. Here the discounted reward is subtracted from the q value. In comparison to alternative possible actions, the advantage function calculates the benefit or advantage of performing a specific action in a given situation. Using this advantage function the possibility of successful penetration is determined and the decision according to the situation is made using the Q learning ensemble deep CNN classifier. The collaborative approach enhances operational efficiency by leveraging advanced machine learning techniques to navigate and adapt to the dynamic and complex nature of penetration testing environments. Overall, the model can facilitate adaptive and intelligent decision-making, leading to more effective penetration testing outcomes. The systematic representation of the proposed penetration testing framework is depicted in Fig. 1.

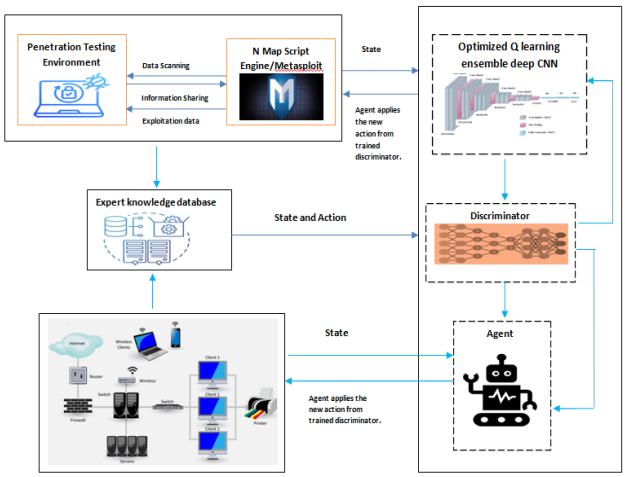


Fig. 1. The proposed block diagram of the penetration testing framework.

A. Penetration Testing Environment

A penetration testing environment, often referred to as a pen-testing lab, is a controlled and isolated system or network setup specifically designed for simulating real-world cyber attacks safely. It duplicates the company's actual IT architecture, including its systems, networks, and apps, enabling cybersecurity experts to find security flaws without endangering sensitive data or operational systems. This environment is equipped with various security tools and resources to assist in the testing process. Data sanitization is crucial to protect privacy and comply with regulations. Comprehensive documentation is essential for tracking and reporting findings. Penetration testing environments facilitate proactive security assessments and help organizations strengthen their cyber security defenses. Creating a penetration testing environment that simulates the Shodan search engine and incorporates an expert knowledge base is a valuable approach for cyber security testing. Shodan is a specialized search engine for finding and analyzing internet-connected devices. It is used by cybersecurity professionals to identify and assess potential security vulnerabilities and misconfigurations in these devices. Its advantage lies in helping experts proactively secure networks by providing insights into exposed assets and potential risks, enhancing cyber security posture. Segmenting the penetration testing environment into two distinct components, one for the Nmap Script Engine (NSE) and the other for the Metasploit framework, can provide an organized and efficient approach to penetration testing.

1) Nmap Script Engine (NSE): The Penetration Scanning Framework for the Nmap Script Engine (NSE) is a critical component of a penetration testing environment. It serves as the initial reconnaissance and vulnerability scanning phase, aiming to identify weaknesses and potential entry points in target systems and networks. NSE leverages the Nmap tool, a versatile and widely used network scanner. Within this framework, NSE scripts are employed to automate specific scanning tasks. These scripts are highly customizable, allowing penetration testers to tailor them to the testing objectives. NSE is used to discover live hosts and open ports within the target environment. It helps testers map out the network's topology and identify reachable systems. By utilizing NSE scripts, the framework gathers information about services running on open ports. This includes identifying service versions, banners, and configurations. NSE scripts capable of detecting vulnerabilities are executed against target systems. These scripts may check for known vulnerabilities in services, applications, or system configurations. Information obtained during scanning, such as service banners and version details, is collected and analyzed to identify potential weaknesses or misconfigurations. Comprehensive reports are generated based on the findings of NSE-based scans. These reports provide organizations with insights into their network's security posture, highlighting vulnerabilities that require remediation.

2) Metasploit: The Penetration Testing Framework for Metasploit, often simply referred to as Metasploit, is a powerful and widely used penetration testing and exploitation framework. It provides security professionals, ethical hackers, and penetration testers with a comprehensive set of tools and resources for identifying vulnerabilities, exploiting them, and assessing the security of computer systems, networks, and applications. Metasploit includes a vast collection of exploit modules that allow testers to exploit known vulnerabilities in target systems. These modules are organized by the target's operating system, service, and application, making it easier to find and execute the right exploit. Metasploit supports various

payloads, which are pieces of code that are delivered to the target system after a successful exploit. Payloads can be used for tasks like gaining remote access, executing commands, or performing post-exploitation activities. The framework provides post-exploitation modules and functionalities to maintain control over compromised systems. This includes activities like privilege escalation, data exfiltration, and lateral movement within a network. Metasploit includes auxiliary modules for various tasks, such as scanning, reconnaissance, and vulnerability detection. These modules can be used to gather information about target systems or to perform non-exploitative actions. Metasploit maintains a database of known vulnerabilities, exploits, payloads, and compromised hosts. This database helps testers keep track of their findings and simplifies the exploitation process. Metasploit can be integrated with other security tools and frameworks, making it a versatile tool for comprehensive security assessments. Integration with tools like Nmap, Wireshark, and Burp Suite enhances its capabilities. Metasploit is available in both open-source community and commercial versions. The community version is free and open-source, while the commercial version offers additional features, support, and updates. Users can create custom scripts and automate tasks within Metasploit, allowing for more efficient and tailored penetration testing processes. Exploit Development: Metasploit provides a platform for developing custom exploits and modules for zero-day vulnerabilities. The framework offers reporting capabilities to document findings, vulnerabilities, and the overall security assessment process.

B. Optimized Q learning Ensemble Deep CNN

The optimized Q-learning ensemble deep CNN classifier is a combination of two powerful techniques in artificial intelligence such as Q-learning and deep CNNs. The advantage of an ensemble Q-learning and deep CNN classifier in penetration testing is its capacity to enhance the accuracy of identifying vulnerabilities and security weaknesses. The PNS algorithm helps in the fine-tuning of the parameters inside the classifier, which helps in enhancing the performance of the classifier.

A popular model-free reinforcement learning algorithm is Q-learning. Assessing the effectiveness of taking particular actions in distinct states, aids an agent in decision-making. By making updates to a Q-table or function that gives each state-action pair a Q-value, the agent gradually learns to optimize its cumulative rewards. The predicted cumulative benefit of performing a certain action in a particular condition is represented by the Q-value. The Q-learning ensemble deep CNN combines the strengths of both Q-learning and deep CNNs. In this approach, the deep CNN is used as a function approximation to estimate Q-values. The agent uses the neural network to anticipate Q-values for state-action pairs rather than keeping a Q-table. This neural network is trained using Q-learning principles, such as temporal difference updates, to learn an optimal policy. With the help of the optimized Q-learning ensemble deep CNN classifier, the agent is able to decide depending on the Q-values that have been learned. The agent can utilize the neural network to evaluate the Q-values of potential actions given a current state and choose the action with the greatest estimated Q-value. This decision-making process is guided by the goal of maximizing cumulative rewards over time.

1) Deep CNN classifier: A deep CNN is a specialized neural network created for processing organized grid-like data, with images being a common application. It has gained significant popularity in the realm of malware detection, where the primary objective is to categorize input data as either benign (safe) or potentially harmful (malicious). The input data typically takes the form of an image or a structured grid-like representation. In the context of malware detection, this representation could be a visual rendering of binary code, a heat map detailing system behavior, or some other organized format. The CNN's architecture typically begins with one or more convolutional layers. These layers employ a set of learnable filters, also known as kernels, which are applied to the input data. Each filter traverses the input data, extracting features by performing convolutions. These convolution operations are meticulously designed to identify distinct patterns or characteristics within the input data. In the context of malware detection, these patterns might correspond to specific code structures or behaviors typically associated with malicious software. An element-wise application of a non-linear activation function, such as ReLU, follows each convolution operation. This introduces essential non-linearity into the model, enabling the network to grasp intricate relationships within the data. In order to reduce the spatial dimensions of the feature maps produced by the convolutional layers, pooling layers, which can be either MaxPooling or AveragePooling, are extremely important. This downsampling serves to simplify computational complexity while capturing the most significant features. The generated feature maps flatten into a 1D vector following many convolutional and pooling layers. This vector encapsulates the high-level features extracted from the input data. This flattened vector then passes through one or more completely connected layers. These layers are akin to conventional neural network layers, with each neuron establishing connections to every neuron in the preceding and following layers. The fully connected layers master intricate combinations of features and ultimately map these features to the output classes, namely benign or malicious. The final fully connected layer commonly employs a softmax activation function to yield probability scores for each class. The output layer, generally featuring two neurons representing benign and malicious classes, produces the ultimate classification results. The softmax function is instrumental in converting the network’s outputs into class probabilities, with the class exhibiting the highest probability serving as the ultimate prediction. For training, the network relies on labeled data where the ground truth (benign or malicious) is known. During this training process, the network adjusts its internal parameters, encompassing weights and biases, employing optimization algorithms. The goal is to

reduce the size of a loss function that measures the discrepancy between expected and real labels. The network is assisted in learning to differentiate between benign and malicious data by this repeated training procedure. Fig. 2 illustrates the architecture of the deep CNN classifier.

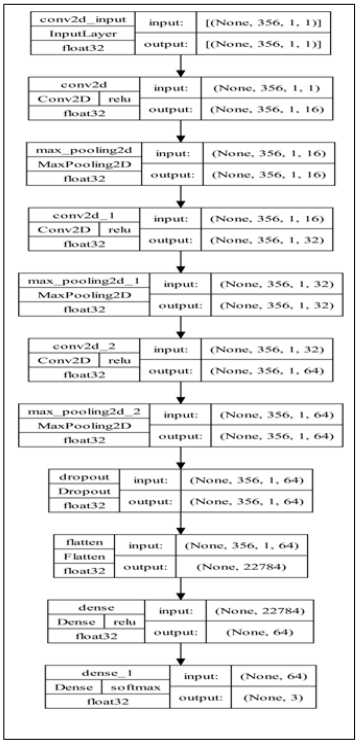


Fig. 2. Architecture of CNN.

2) Q-learning algorithm: A method for reinforcement learning called Q-learning is essential in assisting agents in understanding the best course of action to adopt in contexts where their goal is to maximize their cumulative rewards. This algorithm proves to be particularly valuable when the agent starts with limited knowledge of the environment and needs to gather insights and refine its strategy through interactions over time. Its fundamental concept is rooted in Markov decision processes. The core process involves the agent perceiving the current state of the environment, deciding on the action to take through a specific strategy, and then receiving immediate feedback in the form of a reward. The information regarding the future state of the environment is also included in this comment. Essentially, Q-learning functions by creating a map that connects the present environmental condition to the most beneficial course of action. The primary steps of the algorithm can be summarized as follows.

Step 1: To begin, define the state set as $Se=\{se_1, se_2, ...se_n\}$ and the actions set as $Ae=\{ae_1, ae_2, ...ae_n\}$ and also initiate the state-action function, denoted as $Q(se, ae)$, and create the reward matrix, represented as Re. Additionally, set crucial parameters, including the maximum number of iterations, denoted as M.

Step 2: The process commences by randomly selecting an initial state from the state set S. The iteration ends and a new initial state is selected if, by chance, the initial state is already the goal state. On the other hand, the algorithm moves on to step 3 if the starting state is not the desired state. This mechanism ensures that the algorithm begins with a suitable starting point and repeats until it reaches the target state.

Step 3: The algorithm chooses an action from the pool of all feasible actions available in the current state, adhering to the ε-greedy strategy. This chosen action then guides the agent to transition to the next state within the environment. This approach effectively balances exploration and exploitation, allowing the agent to make decisions that prioritize known, rewarding actions while occasionally exploring new possibilities.

Step 4: Eq. (1) serves as the means to update the Q-matrix.

$$Q(se_t, ae_t) = Q(se_t, ae_t) + \beta * \left[R(se_t, ae_t) + \gamma \max_{ae' \in A} Q\left(\left(se_{t+1}, ae' \right) - Q(se_t, ae_t) \right) \right] \quad (1)$$

Where se_t is the environment's state at the time t, at is the agent's action at time t, $Q(se_t, ae_t)$ is the state-action operates at time t, se_{t+1} is the environment's state at time t + 1, $R(se_t, ae_t)$ is the immediate reward of the environment's feedback from time t to time t + 1, and ae' is the action that maximizes value. Q When the agent arrives se_{t+1}, the learning rate is β varies from $\beta \in (0,1)$ the discount factor is $\gamma \in [0,1]$, and γ is the discount factor.

Step 5: Proceed by updating the state for the next moment, setting it as the current state, which is expressed as $se_t = se_{t+1}$. If the current state (se_t) is not the target state, the algorithm loops back to step 3. This iterative process continues until the target state is reached, ensuring that the agent refines its decision-making strategy over multiple cycles.

Step 6: The method ends when the maximum number of iterations is reached, indicating that the training phase is complete. At this point, the converged Q-matrix is acquired, and the optimal action strategy is determined using Eq. (2). However, if the maximum iterations have not been reached, the process returns to step 2, initiating the next iteration. This iterative approach continues until the training process reaches its defined limit, thereby ensuring the refinement of the optimal action strategy.

$$\pi^*(se) = \arg\max_{a \in A} \left(Q*(se, ae) \right) \quad (2)$$

3) Prairie natural swarm optimization (PNS): The prairie natural swarm optimization (PNS) is developed through the hybridization of coyote and particle swarm algorithms. In the coyote algorithm, the velocity and position is not interpreted so it faces limited capability to explore the search space effectively and slower convergence to optimal solutions. Considering this, the particle swarm algorithm velocity is merged with coyote for faster convergence and balanced the exploration and exploitation.

Motivation Coyote (prairie) Optimization is a novel meta-heuristic algorithm that draws inspiration from the remarkable problem-solving and adaptability exhibited by coyotes, an exceptionally resourceful species. The development of this optimization technique is driven by several factors. Firstly, coyotes showcase remarkable problem-solving skills and adaptability in diverse environments, making them an intriguing source of inspiration for optimizing complex and dynamic scenarios. Their intelligence, social behavior, and efficient foraging strategies offer valuable insights for algorithm design. Secondly, in addressing complex and dynamic problems, existing optimization algorithms may not always be well-suited. Prairie Optimization aims to bridge this gap by providing a nature-inspired approach capable of effectively handling real-world complexities. Additionally, as researchers continue to explore nature-inspired optimization techniques, drawing inspiration from a wide array of species, the creation of new algorithms like Prairie Optimization adds to the expanding toolbox of computational intelligence methods for solving intricate problems in various domains, including engineering, logistics, and finance.

Particle Swarm Optimization (Natural swarm) is an algorithm driven by the emulation of the collective behavior of birds and fish, drawing inspiration from the flocking patterns of birds and schooling behaviors of fish in the natural world. Natural Swarm aims to enhance the optimization of solutions within complex search spaces. Its motivation lies in harnessing the potential of swarm intelligence for problem-solving, with each particle in the swarm representing a potential solution. These particles interact with one another based on the principles of exploration and exploitation. They adjust their positions by learning from both their individual experiences and those of their neighbors, all with the aim of converging toward optimal solutions. Natural swarm is particularly well-suited for tackling optimization problems that challenge traditional methods, such as those in high-dimensional spaces, non-convex landscapes, and scenarios with numerous local optima. Its foundation in nature highlights the strength of collective decision-making, adaptability, and the synergy among individual agents. In essence, the motivation behind natural swarms is to develop a versatile optimization technique that leverages the collective intelligence of swarms to discover high-quality solutions across a broad spectrum of applications in fields like engineering, economics, science, and more.

C. Mathematical Equation of Prairie Natural Swarm Optimization

This section delves into the mathematical equation that underpins Prairie Natural Swarm Optimization, which is presented in the following passage.

In the COA algorithm, the coyote population is partitioned into $M_q \in M^*$ packs, with each pack comprising $M_a \in M^*$ coyotes. This initial suggestion assumes that there are the same number of coyotes in each pack, everywhere. Thus, the algorithm's total population is determined by multiplying $M_q \in M^*$ and $M_a \in M^*$. To simplify matters, this initial version of the algorithm does not take into account solitary coyotes. In this formulation, each coyote symbolizes a potential solution to the optimization issue, and its social condition is expressed in the cost associated with the objective function. This is crucial to note for the convenience of the reader.

Inspired by the social dynamics of coyotes, which are equivalent to the choice variables $\vec{y}$ in a global optimization problem, the COA mechanism was devised. Therefore, the social condition, denoted as V (comprising the set of decision variables), for the a^{th} coyote in the q^{th} pack during the s^{th} time instance is expressed as follows,

$$V_a^{q,s} = \vec{y} = (y_1, y_2, ...y_E) \tag{3}$$

The first step in the COA is to establish the coyote population worldwide. Due to the stochastic nature of the COA, randomization is used to set the initial social conditions for every single coyote. To do this, random values are assigned for the a^{th} coyote in the q^{th} pack and along the i^{th} dimension within the defined search space, as follows.

$$V_{a,i}^{q,s} = L_i + k_i \cdot (U_i - L_i) \tag{4}$$

Where E denotes the search space's dimension and L_i and U_i represent the i^{th} decision variable's lower and upper bounds, respectively. Furthermore, inside the range [0,1], k_i represents a true random number that is produced from a uniform probability distribution. Following this randomization process, the adaptation of the coyotes within their current social conditions is assessed.

$$f_a^{q,s} = f\left(V_{a,i}^{q,s}\right) \tag{5}$$

Coyotes are randomly assigned to packs at the beginning. However, there are times when coyotes decide to leave their existing packs and live alone or decide to join a new pack. A coyote's eviction from a pack is contingent upon the size of the pack at that moment and occurs with a probability represented by the symbol qr. The following is a description of this process,

$$Q_r = 0.005 \cdot M_a^2 \tag{6}$$

Considering the parameter Q_r can take values exceeding 1 for $M_a \le \sqrt{200}$, causing a maximum quantity of coyotes per pack to be limited to 14. The goal of this mechanism is to promote contact and diversity among all of the coyotes in the population. In essence, it encourages cross-cultural communication among people everywhere, leading to a more extensive and dynamic process of information sharing.

In the natural behavior of this species, packs typically consist of two alpha individuals. However, in the COA, only one alpha is considered, specifically the one that demonstrates the highest level of adaptation to the environment. When dealing with a minimization problem, the following definition applies to the alpha of the q^{th} pack at the s^{th} time instance,

$$\alpha^{q,s} = \left\{ V_{a,i}^{q,s} \mid h_{a=\{1,2,\ldots M_a\}} \min f\left(V_{a,i}^{q,s}\right) \right\} \tag{7}$$

The COA operates under the assumption that coyotes possess a level of organization that allows them to share their social conditions and aid in the upkeep of their packs, given the observable signs of swarm intelligence within this species. As a result, the COA compiles all of the data that came from the coyotes and treats it as the pack's cultural inclination.

$$T_i^{q,s} = \begin{cases} G^{q,s}_{\frac{(M_a+1)}{2}}, & M_a \text{ is odd} \\ \dfrac{G^{q,s}_{\frac{M_a}{2},i} + G^{q,s}_{\left(\frac{M_a}{2}+1\right),i}}{2}, & \text{otherwise} \end{cases} \tag{8}$$

For each i in the range, $G^{q,s}$ represents the social conditions that are ranked for all coyotes in the q^{th} pack during the s^{th} time occurrence [1, E]. To put it simply, the median social conditions of all the coyotes in that specific pack are computed to identify the pack's cultural inclination.

With birth and death as basic biological events in mind, the COA determines the coyote's age $b_a^{q,s} \in \mathrm{N}$ in years. A new coyote's social circumstances are said to be a combination of its two randomly chosen parents' social circumstances as well as external factors. This can be stated in the manner shown below:

$$g_i^{q,s} = \begin{cases} V_{k1,i}^{q,s}, & kn_i < Q_t \text{ or } i = i_1 \\ V_{k2,i}^{q,s}, & kn_i \geq Q_t + Q_d \text{ or } i = i_2 \\ B_i, & Otherwise \end{cases} \tag{9}$$

In this case, i_1 and i_2 stand for two randomly selected problem dimensions, and $k1$ and $k2$ stand for two randomly chosen coyotes from the q^{th} pack. Furthermore, B_i is a random number inside the boundaries of the i^{th} dimension's decision variable, kn_i is a uniformly produced random number within the range [0,1], and Q_t , Q_d , and kn_i represent the scatter and association probabilities, respectively. The scatter and association probability, Q_t and Q_d , are important factors that influence how diverse the coyotes' cultures are within the pack. Here are the definitions of Q_t and Q_d in this first edition of the COA.

$$Q_t = 1/E \tag{10}$$

$$Q_d = (1 - Q_t)/2 \tag{11}$$

Where, Q_d exerts an equivalent influence and impact on both parents. To capture the cultural dynamics within the packs, the COA introduces the concepts of the alpha influence (γ_1) and the pack influence (γ_2). The alpha influence is the difference in culture between a randomly picked coyote in the pack (ak_1) and the alpha coyote, while the pack influence is the difference in culture between another randomly selected coyote (ak_2) and the group's cultural tendency. The uniform probability distribution is used to select these random coyotes, and γ_1 and γ_2 are expressed as follows,

$$\gamma_1 = \alpha^{q,s} - V_{ak_1}^{q,s} \tag{12}$$

$$\gamma_2 = T^{q,s} - V_{ak_2}^{q,s} \tag{13}$$

In the coyote algorithm, the velocity and position is not interpreted so it faces limited capability to explore the search space effectively and slower convergence to optimal solutions. Considering this, the particle swarm algorithm velocity is merged with a coyote for faster convergence and balanced exploration and exploitation. Then the new mathematical equation becomes,

$$X^{t+1} = X^t + k_1\gamma_1 + k_2\gamma_2 + v^{t+1} \tag{14}$$

$$X^{t+1} = X^t + k_1\gamma_1 + k_2\gamma_2 + \left[v(t) + g_1u_1\left(D_{best}(t) - y(t)\right) + g_2u_2\left(H_{best}(t) - y(t)\right) \right] \tag{15}$$

Where, the weighing factors for the pack influence and the alpha influence are represented by the variables k_1 and k_2 , respectively. First, a uniform probability distribution is used to produce random numbers within the range [0,1] for both k_1 and k_2 . Furthermore, the particle swarm optimization's social and cognitive acceleration coefficients are represented by the parameters g_1 and g_2 . In the meantime, two uniformly distributed random numbers produced within the interval [0, 1] are u_1 and u_2 .

Algorithm 1: Pseudo code for the proposed Prairie Natural Swarm Optimization

S.No	Pseudo code for the proposed Prairie Natural Swarm Optimization
1.	Initialize M_q packs with M_a coyotes (eqn 4)
2.	Coyotes adaptation verification (eqn 5)
3.	***While do***
4.	***For*** each q pack ***do***
5.	Define alpha coyote (eqn 7)
6.	Compute social tendency of the pack (eqn 8)
7.	***For*** each a coyote of the pack q ***do***
8.	Update the social condition (eqn 12 and 13)
9.	Determine best solution (eqn 15)
10.	***End for***
11.	Birth and death (eqn 9)
12.	***End for***
13.	Transition between packs (eqn 6)
14.	Update age of coyotes
15.	***End while***
16.	Choose best adapted coyote

In the optimization process, the strategies employed draw from the adaptability and strategic decision-making observed in coyote behavior. This entails dynamically adjusting parameters in response to changes in the penetration testing environment. Just as coyotes adapt their hunting strategies based on factors like prey behavior and environmental conditions, the PNS optimization enables the system to flexibly modify parameters to optimize the performance as new threats emerge. Moreover, the PNS optimization leverages the collaborative optimization capabilities inspired by particle swarm behavior. Similar to how swarms of particles collectively explore and converge toward optimal solutions. This collaborative aspect ensures that the optimization process explores a diverse range of parameter configurations, allowing for the discovery of superior settings that enhance the system's overall performance.

The PNS optimization is used to fine-tune the hyperparameters in Q learning ensemble deep CNN. This fine-tuning process ensures that the models are configured optimally for the specific task of penetration testing. By tuning the parameters such as weight and bias, the optimization contributes to improved convergence rates, higher accuracy, and enhanced generalization capability of the models involved in penetration testing. This leads to more effective identification of security flaws and vulnerabilities within the IT applications. Additionally, the dynamic adjustment of parameters enables the system to adapt rapidly to new threats or changes in the environment, thereby enhancing operational efficiency and ensuring robust cyber security measures. Overall, by combining adaptability, strategic decision-making, and collaborative optimization, PNS optimization enables the system to achieve superior performance, effectively mitigating security risks and safeguarding the organization against cyber threats.

IV. RESULT

The subsequent section provides a comprehensive account of the outcomes achieved through the application of Q-learning ensemble deep CNN with Prairie Natural Swarm Optimization for the purposes of penetration testing.

A. Experimental Setup

The experiment, which centers on penetration testing and employs the optimization of Q-learning ensemble deep CNN, is conducted using Python. The experiment is conducted on a Windows 10 computer that has 8GB of internal memory.

B. Dataset

CVE dataset [22]: The research utilizes a dataset sourced from the National Institute of Standards and Technology (NIST) called Common Vulnerabilities and Exposures (CVE). The CVE dataset contains information about cyber security threats, vulnerabilities, and exposures making it a valuable source for penetration testing. It includes various software systems, networks, and applications, ensuring the dataset's diversity. Furthermore, since CVE entries are meticulously documented and categorized, the dataset's representativeness is enhanced, allowing for a wide range of cyber security threats to be captured and analyzed.

C. Parameter Metrics

1) Accuracy: Accuracy in penetration testing refers to the overall correctness of the testing results. It is a measure of how well the test findings and identified vulnerabilities align with the actual security weaknesses present in the target system. High accuracy means that the test results are reliable and reflect the true security status of the system, while low accuracy indicates a higher likelihood of false positives or false negatives.

$$acc = \frac{R_{tn} + R_{tp}}{R_{tn} + R_{tp} + R_{fn} + R_{fp}} \tag{16}$$

2) Sensitivity: Sensitivity, also known as the true positive rate or recall, represents the ability of the penetration test to correctly identify and report actual vulnerabilities or security issues present in the system. A high sensitivity means that the test is effective at finding true vulnerabilities and minimizing the risk of overlooking them.

$$sen = \frac{R_{tp}}{R_{tp} + R_{fn}} \tag{17}$$

3) Specificity: Specificity, on the other hand, measures the ability of the penetration test to avoid false alarms or false positives. A high specificity indicates that the test is less likely to report security issues that do not exist. This is important for minimizing the time and resources required for investigating and remediating issues, as well as preventing unnecessary disruption to the target system.

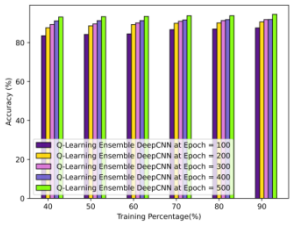
$$spec = \frac{R_{tn}}{R_{tn} + R_{fp}} \tag{18}$$

D. Performance Analysis

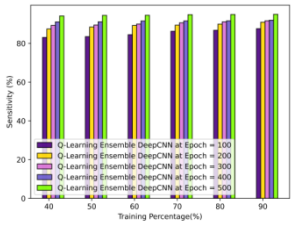
Two important performance indicators are used to demonstrate the efficacy of Q-learning ensemble deep CNN optimization via Prairie Natural Swarm such as training percentage (TP) and k-fold. To fully evaluate its performance, this evaluation is carried out throughout several epochs, namely at intervals of 100, 200, 300, 400, and 500.

1) Performance analysis with TP: Fig. 3 vividly illustrates the effectiveness of Prairie Natural Swarm (PSN) optimized Q-learning ensemble deep CNN when applied to penetration testing within the context of the TP. Fig. 3(a) shows that the PSN-optimized Q-learning ensemble deep CNN performs admirably when it comes to evaluating accuracy at TP 90, with results of 87.55%, 90.65%, 91.84%, 91.86%, and 94.54. Similarly, when evaluating sensitivity at TP 90 through the PSN-optimized Q-learning ensemble deep CNN, the results are notably robust, registering figures of 87.57%, 90.90%, 91.63%, 91.96%, and 94.98 [as illustrated in Fig. 3(b)]. The PSN-optimized Q-learning ensemble deep CNN consistently yields high results in the evaluation of specificity for the 90% training, with values of 87.44%, 90.98%, 91.19%, 91.93%, and 94.99% [as shown in Fig. 3(c)]. These outcomes highlight

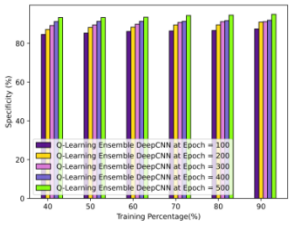
the method's effectiveness over many epochs and show how proficient it is becoming in penetration testing situations.



a) Accuracy



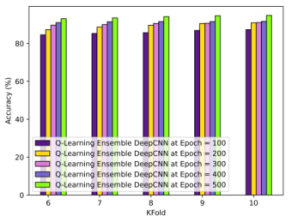
b) Sensitivity



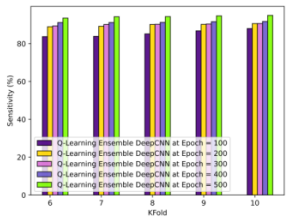
c) Specificity

Fig. 3. Performance with TP.

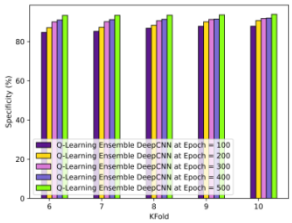
2) Performance analysis with k-fold: Fig. 4 shows the effectiveness of the Q-learning ensemble deep CNN optimized for Prairie Natural Swarm (PSN) in penetration testing, specifically using the k-fold evaluation framework. In the context of assessing accuracy at TP 90, the PSN-optimized Q-learning ensemble deep CNN demonstrates performance results of 87.29%, 90.86%, 91.00%, 91.67%, and 94.82% ([as presented in Fig. 4(a)]. Similarly, when considering sensitivity at TP 90 through the PSN-optimized Q-learning ensemble deep CNN, the results remain robust, recording figures of 87.97%, 90.58%, 90.60%, 91.68%, and 94.90% [as depicted in Fig. 4(b)]. The PSN-optimized Q-learning ensemble deep CNN regularly produces good results in the evaluation of specificity for the 90% training, with values of 87.83%, 90.70%, 91.76%, 91.99%, and 93.89% [as shown in Fig. 4(c)]. These results illustrate the approach's efficacy in the k-fold evaluation and point to its possible applications in penetration testing scenarios.



a) Accuracy



b) Sensitivity

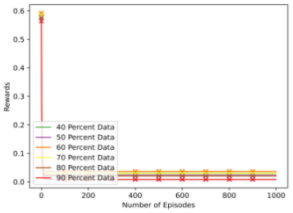


c) Specificity

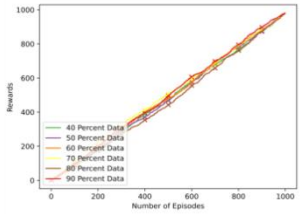
Fig. 4. Performance with k-fold.

E. Analysis based on Q-learning

Fig. 5 provides a visual representation of the effectiveness of PSN-optimized Q-learning ensemble deep CNN in the context of penetration testing, specifically within the framework of loss and rewards evaluation. In the context of assessing loss at 90% data demonstrates results of 0.007982595, 0.007981617, 0.007982108, 0.007980786, and 0.00798194 for 995, 996, 997, 998, 999 episodes [as presented in Fig. 5(a)]. Similarly, when considering rewards at 90% data demonstrates results of 978, 979, 980, 980, and 981 for 995, 996,997, 998, 999 episodes [in Fig. 5(b)].



a)



b)

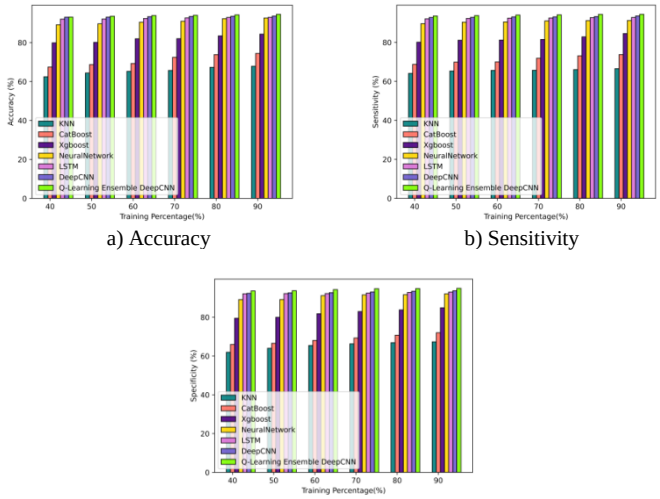
Fig. 5. Analysis based on Q-learning.

F. Comparative Methods

KNN [H1] [16], CatBoost [H2] [17], Xgboost [H3] [18], Neural Network [H4] [19], LSTM [H5] [20], Deep CNN [H6] [21] is compared with PSN optimized Q-learning ensemble deep CNN [H7].

1) Comparative analysis with TP: Fig. 6 provides a visual representation of the penetration testing methodology evaluation. Surprisingly, Fig. 6(a) depicts that the H7 outperforms the H6, outperforming by a gradually increasing margin of 0.93 when it comes to accuracy evaluation inside the 90% training. This significant improvement is also seen when sensitivity is assessed in the same training setting which is presented in Fig. 6(b), where H7 once more demonstrates a notable increase of 0.89 relative to H6. In addition, when looking at specificity for the 90% training, the H7 shows a 1.34 gain over the H6, continuing its remarkable performance trend, and the analysis of specificity is illustrated in Fig. 6(c). These results highlight the H7's obvious benefits in the field of penetration testing.

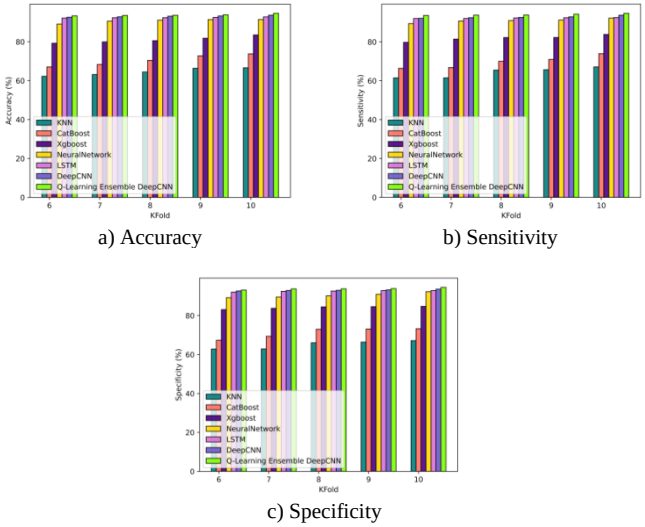
2) Comparative analysis with k-fold: Fig. 7 provides a visual representation of the penetration testing methodology assessment. Surprisingly, Fig. 7(a) depicts that the H7 outperforms the H6 by a steadily increasing margin of 1.00 when it comes to accuracy assessment inside the nine-fold framework. This significant improvement is also shown in the sensitivity analysis in the same training situation which is presented in Fig. 7(b), where the H7 again demonstrates a noteworthy rise of 1.01 in comparison to the H6. In addition, the H7 exhibits a 1.01 improvement over the H6 in terms of specificity inside the 9-fold, thereby sustaining its remarkable performance trend, and the analysis of specificity is illustrated in Fig. 7(c). These outcomes highlight the H7's noteworthy benefits when it comes to penetration testing.



a) Accuracy b) Sensitivity

c) Specificity

Fig. 6. Comparative with TP.



a) Accuracy b) Sensitivity

c) Specificity

Fig. 7. Comparative with k-fold.

G. Comparative Discussion

Comparisons with other current methods are used to determine the efficacy of the proposed PSN-optimized Q-learning ensemble deep CNN method. The PSN-optimized Q-learning ensemble deep CNN outcomes are 94.54%, 94.40%, and 94.90% respectively for TP. For the k-fold, the PSN-optimized Q-learning ensemble deep CNN obtained values are 94.64%, 94.69%, and 94.52% respectively. Table I depicts the obtained values of the PSN-optimized Q-learning ensemble deep CNN method with existing methods.

TABLE I. COMPARATIVE DISCUSSION OF PROPOSED METHOD WITH EXISTING METHODS

Methods	TP(90)			k-fold (9)		
	Accuracy (%)	Sensitivity (%)	Specificity (%)	Accuracy (%)	Sensitivity (%)	Specificity (%)
KNN	67.87	66.53	67.29	66.62	67.13	67.14
Cat Boost	74.45	73.74	72.00	73.80	73.92	73.26
Xgboost	84.36	84.53	84.86	83.52	83.83	84.73
Neural Network	92.64	91.22	92.05	91.47	92.20	92.30
LSTM	92.96	92.77	92.99	92.81	92.54	92.86
Deep CNN	93.66	93.56	93.63	93.69	93.74	93.56
Proposed	94.54	94.40	94.90	94.64	94.69	94.52

V. CONCLUSION

In this research the effective penetration testing framework is developed using prairie natural swarm (PNS) optimized Q-learning ensemble deep CNN. Initially, the penetration testing environment (Shodan search engine) is simulated and along with that expert knowledge base is also be generated. Subsequently, the Nmap script engine and Metasploit are deployed, providing robust tools for network investigation and vulnerability assessment. The system state is then relayed to the Q-learning ensemble deep CNN classifier. This unique ensemble combines the strengths of Q-learning and deep CNNs, enabling optimal policy learning for decision-making. The prairie natural swarm optimization algorithm is developed through the hybridization of coyote and particle swarm characteristics to fine-tune classifier parameters, enhancing performance. Additionally, the discriminator is trained to maximize standard action rewards while minimizing discounted action rewards, distinguishing valuable from less valuable information. By evaluating the advantage function, successful penetration likelihood is determined, informing situational decision-making through the Q-learning ensemble deep CNN classifier. PNS-optimized Q-learning ensemble deep learning is used to measure the output along with accuracy, sensitivity, and specificity. In comparison to other current approaches, it achieves higher efficiency, achieving 94.54%, 94.40%, 94.90% for TP and 94.64%, 94.69%, 94.52% for k-fold. In the future, advanced deep learning techniques, dynamic environment adaption, integration with security

operations, privacy–preserving techniques will be involved to address the robustness and resilience challenges.

REFERENCES

[1] A.Mohsen, A. Taghavirashidizadeh, D. Javaheri, A. Masoumian, S.J. Ghoushchi, and Y. Pourasad. "DQRE-SCnet: a novel hybrid approach for selecting users in federated learning with deep-Q-reinforcement learning based on spectral clustering." Journal of King Saud University-Computer and Information Sciences 34, no. 9 (2022): 7445-7458.

[2] A. Rohit, O. Bills, N. Chilamkurti, and M.J.M. Chowdhury. "Automated Penetration Testing Framework for Smart-Home-Based IoT Devices." Future Internet 14, no. 10 (2022): 276.

[3] C. Jihua, Z. Wang, S. Tian, J. Zhao, and S. Wang. "Incorporating Clustering Modification Directions into Reinforcement Learning Based Cost Learning Framework." (2022).

[4] M., Soheil, A. Rasouli, Y. Malekshah, A. Ramezani, and A.Malekshah. "Reliability-driven distribution power network dynamic reconfiguration in presence of distributed generation by the deep reinforcement learning method." Alexandria Engineering Journal 61, no. 8 (2022): 6541-6556.

[5] Jinyin Chen, Shulong Hu, Haibin Zheng, Changyou Xing, Guomin Zhang, "GAIL-PT: An intelligent penetration testing framework with generative adversarial imitation learning" Computers & Security,Volume 126,2023,103055,ISSN 0167-4048.

[6] Zhenguo Hu, Razvan Beuran, Yasuo Tan Japan Advanced Institute of Science and Technology. "Automated Penetration Testing Using Deep Reinforcement Learning." © 2020, Zhenguo Hu. Under license to IEEE. DOI 10.1109/EuroS&PW51379.2020.00009.

[7] B.Hafsa, M. Jouhari, K. Ibrahimi, J. B. Othman, and E. M.Amhoud. "Anomaly Detection in Industrial IoT Using Distributional Reinforcement Learning and Generative Adversarial Networks." Sensors 22, no. 21 (2022): 8085.

[8] W.Yongjie, Y. Li, X. Xiong, J. Zhang, Q. Yao, and C. Shen. "DQfD-AIPT: An Intelligent Penetration Testing Framework Incorporating Expert Demonstration Data." Security and Communication Networks 2023 (2023).

[9] L. Yang, Y. Wang, X. Xiong, J. Zhang, and Q. Yao. "An Intelligent Penetration Test Simulation Environment Construction Method Incorporating Social Engineering Factors." Applied Sciences 12, no. 12 (2022): 6186.

[10] K. S. Hussain, T. J. Alahmadi, W. Ullah, J. Iqbal, A. Rahim, H.K.Alkahtani, W.Alghamdi, and A.O. Almagrabi. "A new deep boosted CNN and ensemble learning based IoT malware detection." Computers & Security 133 (2023): 103385.

[11] N.Thanh Thi, and V. J. Reddi. "Deep reinforcement learning for cyber security." IEEE Transactions on Neural Networks and Learning Systems (2021).

[12] K.Nickolaos, N. Moustafa, B. Turnbull, F. Schiliro, P.Gauravaram, and H.Janicke. "A deep learning-based penetration testing framework for vulnerability identification in internet of things environments." In 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 887-894. IEEE, 2021.

[13] S.Wei, X. Li, S. Afroz, D. Garg, D. Kuznetsov, and H. Yin. "Mab-malware: A reinforcement learning framework for blackbox generation of adversarial malware." In Proceedings of the 2022 ACM on Asia conference on computer and communications security, pp. 990-1003. 2022.

[14] M. Xianbo, S. Tan, B. Li, and J. Huang. "MCTSteg: A Monte Carlo tree search-based reinforcement learning framework for universal non-additive steganography." IEEE Transactions on Information Forensics and Security 16 (2021): 4306-4320.

[15] Y.Wei, W.Xiang, Y. Yang, and P. Cheng. "Optimizing federated learning with deep reinforcement learning for digital twin empowered industrial IoT." IEEE Transactions on Industrial Informatics 19, no. 2 (2022): 1884-1893.

[16] J. Galupino, and J. Dungca. "Development of a k-Nearest Neighbor (kNN) Machine Learning Model to Estimate the SPT N-Values of Valenzuela City, Philippines." In IOP Conference Series: Earth and Environmental Science, vol. 1091, no. 1, p. 012021. IOP Publishing, 2022.

[17] H. Jiazhi, X. Feng, and M. Lu. "Accurate and Generalizable Soil Liquefaction Prediction Model Based on the CatBoost Algorithm." (2023).

[18] A. R. T. E. M., V. Y. A. C. H. E. S. L. A. V. A.Maaz, I. Ahmad, M. Ahmad, P.Wróblewski, P. Kamiński, and U. Amjad. "Prediction of pile bearing capacity using XGBoost algorithm: modeling and performance evaluation." Applied Sciences 12, no. 4 (2022): 2126.

[19] T. Kharchenko, D. M. Y. T. R. O. Uzun, and A. R. T. E. M. Nechausov. "Architecture and model of neural network based service for choice of the penetration testing tools." International Journal of Computing 20, no. 4 (2021): 513-518.

[20] K.Nickolaos, N. Moustafa, B. Turnbull, F. Schiliro, P. Gauravaram, and H. Janicke. "A deep learning-based penetration testing framework for vulnerability identification in internet of things environments." In 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 887-894. IEEE, 2021.

[21] H. Mingming, Z. Zhang, J. Ren, J. Huan, G. Li, Y. Chen, and N. Li. "Deep convolutional neural network for fast determination of the rock strength parameters using drilling data." International Journal of Rock Mechanics and Mining Sciences 123 (2019): 104084.

[22] CVE dataset,https://www.kaggle.com/datasets/andrewkronser/cve-common-vulnerabilities-and-exposures, on december 2023.