Smart Phishing Detection on Webpages Using Multi-Agent Deep Learning and Multi-Dimensional Features

Ziba Jafari, Seyyed Hamid Ghafouri*, and Mohammad Ahmadinia

Department of Computer Engineering, Kerman branch, Islamic Azad University, Kerman, Iran

Abstract

The increasing sophistication of phishing attacks has made their detection more challenging, as attackers use deceptive tactics to trick users into revealing sensitive information through fraudulent websites. Traditional detection methods struggle to keep up with evolving phishing techniques, necessitating more adaptive approaches. This study introduces a multi-agent deep learning framework that utilizes three specialized models to analyze different aspects of a webpage, including the URL, page content, and Document Object Model structure. The outputs of these models are combined using a highest confidence score mechanism to enhance accuracy. Experimental results demonstrate that this method outperforms existing techniques, achieving 99.21% accuracy with a false positive rate of only 0.22%. It effectively detects both known and new phishing sites, making it a robust solution against emerging threats. Furthermore, this approach highlights the potential of deep reinforcement learning in cybersecurity, paving the way for more automated and resilient security systems to combat phishing attacks.

Keywords: Deep Learning, Phishing, Representation Learning, Multi Agent Deep Reinforcement Learning (MADRL).
Article history: Received 2025/02/22; Revised 2025/04/15; Accepted 2025/06/04, Article Type: Research paper

1. Introduction

Phishing attacks have become a major cybersecurity threat, targeting millions of users annually. These attacks deceive individuals into revealing sensitive information through counterfeit websites[1],[2],[3]. Reports from the Anti-Phishing Working Group (APWG) indicate a significant rise in phishing activities, with a record 963,994 attacks observed in the third quarter of 2024 . Notably, during the COVID-19 pandemic, healthcare facilities with weaker security were heavily targeted. Despite 75% of phishing websites using SSL protection, it did not ensure their legitimacy[4],[5],[6].Machine learning techniques like Decision Trees, Random Forests, and Support Vector Machines have been used for phishing detection but struggle with unseen websites. Deep learning methods, such as CNNs and RNNs, offer improved performance but require extensive data and computational resources [7]. Existing approaches in phishing detection rely on URL-based, heuristic-based [8], visual similarity-based [9], and machine learning methods [10],[11]. However, many focus on a single input feature, limiting their ability to capture the full range of features on phishing webpages.

This study introduces a novel multi-agent deep learning approach for phishing detection, utilizing three distinct Deep Q-Networks (DQNs) to analyse features from the URL, HTML content, and DOM structure. By integrating data from multiple sources and applying an attention mechanism, the model selectively combines outputs based on their performance, significantly improving accuracy, precision, recall, and F1 score. This approach outperforms state-of-the-art methods and demonstrates robust detection capabilities even for previously unseen phishing websites[12],[13],[14].The proposed method leverages the adaptability of deep reinforcement learning to continuously improve its performance. The results show promise in enhancing phishing detection and contribute to better cybersecurity for individuals, businesses, and governments. This multi-faceted approach addresses the limitations of existing methods and offers a comprehensive solution for combating phishing attacks. This paper's main contributions are as follows:

The use of representation learning techniques to automatically learn webpage representations across all dimensions, based on the webpage's URL, content, and DOM structure, all treated as text.

The proposal of a multi-agent deep learning model that combines DDQN.

Conducting four experiments on our model from various perspectives, demonstrating strong classification performance.

The paper's organization is as follows: Related works on phishing webpage detection are presented in Section II. The framework and the detailed process of our model are outlined in Section III. In Section IV, the performance of our model is evaluated. Finally, the paper concludes with a discussion of future work.

2. Related work

Phishing website detection remains a significant challenge in cybersecurity. Various machine learning techniques, including logistic regression, decision trees, support vector machines, and neural networks, have been applied to this problem, with deep learning showing promising results in recent years. Studies in the literature [15] have explored diverse input features for detecting phishing websites, broadly categorized into URL-based, content-based, and DOM-based features. URL-based features, such as URL length, the number of dots, and specific keywords, are commonly used to distinguish legitimate websites from phishing ones. Content-based features, including keyword analysis, the presence of multimedia elements, and the text-to-HTML ratio, provide additional insights. Similarly, DOM-based features, like hidden fields, form tags, and suspicious script tags, have proven useful in detecting malicious webpages. Phishing detection methods can be broadly categorized into four types. Blacklist-based approaches compare URLs and webpage details against blacklists, achieving low false positive rates but missing unlisted websites, such as those used by Google Chrome [12]. Tools like the PhishBench framework [16] offer a benchmarking environment to evaluate advanced detection techniques. Heuristic-based methods extract rules from common phishing features; tools like PhishDetector [8] offer real-time detection but are prone to high false positive rates due to fuzzy matching. These methods depend on features such as URL statistics, Whois and DNS information, and webpage content ,[15], ,[17], [18], with advanced models extracting up to 2,130 features, as in Google Chrome's case.

Recent advancements[10], which integrates lexical analysis and URL HTML Encoding for real-time classification with unbiased voting mechanisms for improved accuracy. Despite these innovations, attackers continuously adapt by avoiding detectable features or algorithms, necessitating ongoing refinement to counter increasingly sophisticated phishing tactics.

Recently, deep learning has gained popularity as an alternative to traditional machine learning methods and has been successfully applied in phishing webpage detection [13],[14]. These studies are classified into three categories: 1) artificial feature engineering, where features are manually extracted for input to deep neural networks (DNNs), 2) automatic feature learning, where deep neural networks learn features directly from the data, and 3) hybrid methods that use both artificial and automatic features. In artificial feature engineering, DNNs are fed manually extracted features such as URL properties, domain details, webpage content, and encoding [13], [19]. Although these methods can detect phishing, they are still susceptible to human bias. Automatic feature learning allows DNNs to extract features on their own, enhancing model accuracy without human intervention. Recent studies also use NLP techniques to analyse webpage or email content for phishing detection [12, 20]. URL-based methods, which treat URLs as text and utilize models like LSTM, Denoising Auto encoder (DAE), and CNN, have been effective in detecting phishing websites [21], [22]. Page content-based methods, on the other hand, process webpage content as text and extract semantic features, often using models like Word2Vec [12],[23], which help avoid human biases and enhance generalization.

The study in [24] tackled class imbalance in phishing detection by creating a feature space using deep learning, while another study in [7] introduced a real-time browser plugin for phishing detection. Methods like those in [25] address dimensionality and sparsity in phishing detection, though they typically use a single feature input, such as URL or page content, limiting their comprehensiveness. Hybrid methods combine both artificial and automatic features. For instance, a hybrid CNN-LSTM model in [14] uses both extracted URL features and traditional page content features for classification. Other techniques, such as those in [26], use CNN with self-attention and GANs to balance datasets and improve detection accuracy, while [27] proposes a multi-agent system for network anomaly detection that integrates deep learning methods. PhishDet [28] combines Long-Term Recurrent Convolutional Networks and Graph Convolutional Networks, utilizing both URL and HTML features for phishing detection. A popular NLP-based approach using BERT [17] has also been applied to phishing URL detection, and methods like Web2Vec [12, 20] use hybrid CNN and BiLSTM models with attention mechanisms to extract both local and global features. In contrast, our model employs a multi-agent deep learning approach to

learn webpage representations from various perspectives. Our proposed method uses three DDQNs trained on different sets of features, including URL, content, and DOM, with additional features. These DDQNs' outputs are aggregated through voting to classify webpages. The results show our model outperforms existing phishing detection techniques in accuracy, precision, recall, and F1 score, offering significant improvements in phishing detection for internet security.

3. Proposed method

This section outlines the proposed method for phishing detection. Firstly, the formal definition of phishing detection is provided, followed by a detailed description of the proposed method framework and its key technologies.

A) Problem statement

This article presents the problem of detecting phishing webpages as a binary classification task, where the two possible classes are "phishing" or "benign". The input data comprises a collection of webpages N, with N = {WP_1, ..., WP_i, ..., WP_n}, where WPi refers to the i-th webpage with i ranging from 1 to n. Each webpage consists of three components, WP_i = {U_i, H_i, D_i}, where U_i denotes the webpage's URL, H_i is the page content, and D_i represents the DOM structure. The training dataset is represented by T webpages, each with corresponding data and class labels in the format (WP_1, X_1, Y_1), (WP_2, X_2, Y_2), ..., (WP_T, X_T, Y_T), where:For i = 1, 2, ..., T, WP denotes a webpage in the given training set T.

The features of webpage WPi are denoted by Xi, which are extracted from its URL, content, and DOM structure and are represented by a feature matrix. The label of the underlying webpage is denoted by Yi ∈ {0, 1}, where Yi = 0 represents a benign webpage and Yi = 1 represents a phishing webpage. The goal of proposed method is to learn a discriminant model f: X → Y using the training dataset, which can then be used to classify new webpages as phishing or benign

B) The overall framework

The proposed model detects phishing webpages through a multi-step process. First, it preprocesses webpages to extract data like URLs, content, and DOM structure, forming a dataset. Using NLP-based word embedding, the model learns data representations, which are then processed by a DDQN network to extract features. Finally, the combined features are classified as phishing or benign. proposed method employs multi-agent deep reinforcement learning for comprehensive feature extraction, automating classification through data preprocessing and feature learning. . Key technologies and processes are depicted in Figure 1.

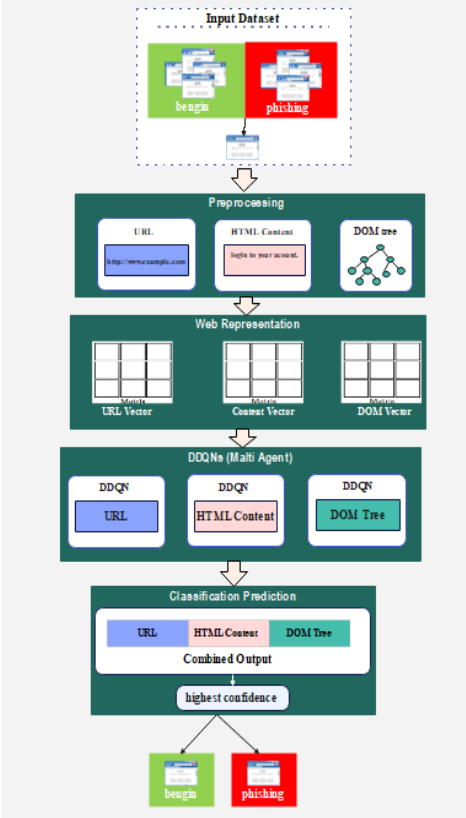

Fig. 1. Model Framwork

C) Web page reprocessing

Webpage reprocessing involves parsing webpage data and constructing various corpora to enhance feature representation. While prior studies primarily relied on URLs for phishing detection [12],[14], URLs alone lack the structural and semantic details of phishing webpages. The proposed method overcomes this limitation by learning features from the URL, page content, and DOM structure. URL corpora are processed at character and word levels: character-level analysis normalizes URLs to a uniform length, maps characters to a 96-character vocabulary, and encodes them into OneHot vectors; word-level analysis segments URLs into sequences of words based on structural elements such as protocols and paths, emphasizing sequential relationships. For webpage content, both word-level and sentence-level corpora are created, separating sentences by periods and removing multimedia, HTML tags, and non-textual elements for simplicity. The DOM structure is represented as a hierarchical sequence of HTML tags, parsed from the DOM tree using a breadth-first approach. Tags are treated as words, forming a word-level corpus that encapsulates the structural essence of the webpage.

D) Webpage representation

The previous section discussed the use of One-Hot encoding for constructing corpora, which is limited due to its lack of semantic relationships and sparse encoding. To overcome this, techniques like word embeddings, which map words to low-dimensional vectors capturing semantic relationships, were introduced [12]. The proposed method uses a simpler approach by embedding the One-Hot matrix as word vectors through a single-layer neural network, avoiding the complexity of pre-trained models like Word2Vec. This embedding layer, along with feature extraction and classification components, is optimized using backpropagation to enhance semantic representation. The embedding transforms the sparse One-Hot matrix into a dense, lower-dimensional representation matrix, reducing computational cost and memory usage.

Once the representation matrix S is acquired, it can be utilized as an input for the succeeding feature extraction and classification sections. These sections can be optimized together with the matrix using backpropagation to enhance the model's semantic representation capability. The same approach can be applied to the word-level and DOM structure-level corpora in the proposed method to acquire learned representations. The method used to learn the character representation for URLs in The proposed model can be illustrated using an example. Suppose we have a URL character-level corpus U, and we want to learn the representation for the i-th URL, denoted as "Ui"."Ui" is encoded using One-Hot encoding. Let's say the j-th character is represented by vector gj after One-Hot encoding, where gj = (gj1, gj2, · · ·, gjm) T . The matrix G has dimensions of m * n, where each column represents a character gi of the URL "Ui".To obtain the URL character embedding, each URL " Ui " is first represented as a One-Hot matrix G with dimensions Gm×n = (g1, g2, · · ·, gn). The One-Hot matrix G is then mapped to its representation matrix S using a single-layer neural network embedding. The weight matrix of the embedding layer is denoted as W, and it has dimensions Rp×m, where p is the embedding dimension and is set to 128 in this case. The calculation process for obtaining the URL character embedding is as eq.(1).

$$S^C = WG = \begin{bmatrix} w_{11} & \cdots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{p1} & \cdots & w_{pm} \end{bmatrix} * \begin{bmatrix} g_{11} & \cdots & g_{1n} \\ \vdots & \ddots & \vdots \\ g_{m1} & \cdots & g_{gm} \end{bmatrix} \quad (1)$$

Once the representation learning process is completed for webpage WP_i, its representation is referred to as X_i, which is a concatenation of five vectors with fixed-length as follows (2):

$$x_{i(URL)} = (U_i^c, U_i^w) \quad (2)$$

$$x_{i(CONTENT)} = (C_i^w, C_i^s)$$

$$x_{i(DOM)} = (D_i)$$

$$x_{i(URL)} = (U_i^c, U_i^w, U_i^f)$$

$$x_{i(DOM)} = (C_i^w, C_i^s, C_i^f) \quad (3)$$

$$x_{i(DOM)} = (D_i, D_i^f)$$

The representation **URL** vectors , ($\boldsymbol{U_i^c}$ and $\boldsymbol{U_i^w}$) are the representation vector learned from Ui. The CONTENT vectors ($\boldsymbol{C_i^w}$, $\boldsymbol{C_i^s}$) are vectors learned from the page content, and DOM vector($\boldsymbol{D_i}$) is learned from the DOM structure. To obtain the character-level representation$\boldsymbol{U_i^c}$, statistical features of each element are extracted, including the mean, variance, skewness, and kurtosis. On the other hand, a bag-of-words approach is used to represent the webpage content to obtain the word-level representation vector $\boldsymbol{U_i^w}$, where the frequency of each word in the webpage is counted and normalized by the total number of words in the webpage. Then, statistical features of the word frequency distribution are extracted, including the mean, variance, skewness, and kurtosis. For the character-level representation vector $\boldsymbol{C_i^w}$ and the sentence-level representation vector $\boldsymbol{C_i^s}$, statistical features are extracted from the page content. On the other hand, for the DOM structure vector $\boldsymbol{D_i}$, features are directly extracted from the DOM tree.

The five vectors, which have the same dimensions and contain information on the character-level, word-level, and sentence-level, enable mathematical calculations based on these features. The next step involves feature extraction, where important information is selected and extracted from the input data to create a new set of features that can more effectively represent the input data. In The proposed model, feature extraction is performed on the five fixed-length vectors obtained after representation learning. After feature extraction in this part, The results of the vectors obtained from this part are combined with 20 additional features below, including 20 features related to URL, 20 features related to HTML content, and 20 features related to DOM, for better learning of the learning network.

E) Additional feature extraction

The proposed approach extracts feature representations from the URL, content, and DOM, each incorporating an additional 20-dimensional vector space to enhance the distinction between benign and phishing webpages. URL-based features include 20 attributes, such as URL length and the presence of IP addresses, aiding in phishing detection [15] ,[29]. Similarly, 20 content-based features analyze multimedia elements, including images and JavaScript, ensuring high-accuracy

detection despite potential vulnerabilities. Additionally, 20 DOM tree features, derived from [15], provide structural insights for effective detection. Tables 1–3 outline these features.

F) Normalization

The normalization process converts all feature vectors and class labels for each URL into binary values (0 and 1). After feature extraction and normalization, the input matrix s(xi) is defined as shown in equation (3). The variables $\boldsymbol{U_i^f}$, $\boldsymbol{C_i^f}$ *and* $\boldsymbol{D_i^f}$ represent additinal features that are obtained through learning from the URL, content, and DOM structure.

Table.1.
URL Based Features

URL Feature	*Explanation for 0*	*Explanation for 1*
IP	URL doesn't use IP address	URL uses IP address, potentially phishing
U_Length	Short URL length	Long URL length, potential phishing
at_U	No "@" character in URL	"@" character present in URL, potential phishing
Dots_U	Few dots (".") in URL	Many dots in URL, potential phishing
TLD_P	No popular TLD in path	Popular TLDs in path, potential phishing
https	URL doesn't use https protocol	URL uses https protocol, potential phishing
hyphen_U	No "-" character in URL	"-" character present in URL, potential phishing
spe_chars_U	Few special characters in URL	Many special characters in URL, potential phishing
Sens_words	No sensitive words in URL	Sensitive words in URL, potential phishing
Length_S	Subdomain length is normal	Short subdomain length, potential phishing
Brand_Name	No brand name in URL	Brand name in URL, potential phishing
Det_Keywords	No specified keywords in URL	Specified keywords in URL, potential phishing
Slash_U	Few double slashes // in URL	Many double slashes in URL, potential phishing
Prefix	URL doesn't contain protocol	URL contains protocol, potential phishing
Port	URL uses reliable ports	URL uses non-reliable ports, potential phishing
Punctuation	Few punctuation characters in URL	Many punctuation characters in URL, potential phishing
Subdomain_U	Only one subdomain in URL	Multiple subdomains in URL, potential phishing
Length_DN	Domain name length is normal	Long domain name length, potential phishing
Hex_U	No hexadecimal characters in URL	Hexadecimal characters in URL, potential phishing
Paths	URL has no paths	URL has paths, potential phishing

Table.2.
Content HTML Based Features

Content Feature	*Explanation for 0*	*Explanation for 1*
Anchor	Links match the domain name	Links ratio different from the domain name, potentially phishing
U_request	Few requests from different domains	High number of requests from different domains, potentially phishing
Popup	No sensitive data requested via popups	Sensitive data requested via popups, potential phishing
Links_in_tags	No blacklisted addresses in <link>,<meta>,<script> tags	Blacklisted addresses in tags, potential phishing
Abn_req_URL	Request rate in object tag matches own domain	Abnormal request rate resembling another site, potential phishing
Cookies	Cookies used	No cookies used, potentially phishing
Iframe	Iframe tags don't link to external sites	Iframe used to deceive visitors, potential phishing
Submit	Submit button used sparingly	Frequent use of submit button, potential phishing
Form	Limited number of forms	Multiple forms used, potential phishing
Favicon	Favicon domain matches website URL	Favicon loaded from different domain, potential phishing
Mailto	Safe form management	Sending user info via mailto, potential phishing
IMG_Hyperlink	Pictures link to same domain	Pictures link to different domains, potential phishing
Susp_links	Text link domain matches text	Text link domain differs, potential phishing
Right_click	Right-click common on websites	Rare right-click on website, potential phishing
Security	Secure forms used	Insecure forms used, potential phishing
Action	URL action relative or absolute	Relative URL action, potential phishing
Dest_port	Destination port not anomalous	Anomalous destination port, potential phishing
Links	Few URLs linking to the page	Many URLs linking to the page, potential phishing
Message	Social media posts not anomalous	Anomalous repetitive message, potential anomaly
Diff_message	No repeated messages from different users	Repeated messages from different users, potential anomaly

Table.3.
DOM Tree Based Features

DOM Feature	*Explanation for 0*	*Explanation for 1*
Number of DOM Nodes	Fewer nodes, simpler structure	Many nodes, complex structure
Depth of the DOM Tree	Shallower tree	Deeper tree
Tag Count	Normal tag distribution	Unusually high counts of specific tags
CSS Class Count	Few CSS classes used	Many CSS classes used, indicating complexity
DOM Element Types	Absence of certain DOM elements	Presence of certain DOM elements
Text Length	Low total text length	High total text length, overwhelming content
Number of Child Elements	Normal number of child elements	Excessive number of child elements, potentially obfuscating
Hidden Form Elements	No hidden form elements	Hidden form elements, potentially capturing data secretly
Pop-up Windows	No pop-up windows	Presence of pop-up windows, potentially deceptive
Iframe Elements	No iframe elements	Presence of iframe elements, potentially loading malicious content
External Resources	Minimal external content	Multiple external resources, indicating potential data collection
Script Tags	No suspicious scripts	Presence of suspicious or obfuscated scripts
Meta Refresh Tags	No meta refresh tags	Presence of meta refresh tags, potentially redirecting
Interactive Elements	Minimal interactive elements	Presence of interactive elements, potentially deceptive
Mismatched Domains	URL matches actual domain	URL doesn't match actual domain, potentially deceptive
Hover URLs	Legitimate hover URLs	Fake or misleading hover URLs, potentially deceptive
Hidden Content	No hidden content	Presence of hidden content, potentially deceptive
Download Links	Normal download links	Suspicious download links, potentially leading to malware
Mixed Content	Consistent secure content	Mixture of secure and non-secure content, potentially exploited
URL Parameters	Normal URL parameter count	Unusual or excessive URL parameters, potentially malicious

G) Background of Multi-Agent Deep Reinforcement Learning

This section provides a foundational overview of key concepts and methodologies relevant to our approach. We first discuss Multi-Agent Deep Reinforcement Learning (MADRL) and its role in collaborative environments. Then, we introduce the Double Deep Q-Network (DDQN) algorithm, which mitigates overestimation bias and instability in traditional Q-learning. These concepts lay the groundwork for understanding how our model integrates deep reinforcement learning and multi-agent architectures to enhance phishing detection accuracy and robustness [30], [31].

Multi-Agent Deep Reinforcement Learning (MADRL) enables multiple agents to interact within a shared environment to achieve a common objective. These agents either act independently to maximize individual rewards (competitive model) or collaborate to optimize a shared reward (cooperative model) [32] [33].

To classify webpages as legitimate or phishing, our model employs a multi-agent deep reinforcement learning algorithm. This algorithm

processes webpage features—including URL, content, and DOM—by mapping them into a high-dimensional feature space. The function f:Rd→R optimizes these representations to enhance phishing detection. Our approach integrates this function into a deep learning framework designed to refine classification accuracy through iterative training.

H) Background of Double Deep Q-Network (DDQN)

The Double Deep Q-Network (DDQN) improves upon the traditional Deep Q-Network (DQN) by addressing overestimation bias in Q-learning. Standard Q-learning estimates action values using a single Q-network, often leading to overoptimistic value predictions and unstable learning. DDQN overcomes this issue by employing two separate networks: one for action selection and another for evaluation, ensuring more accurate Q-value estimation [34] [35],[36].

Key Steps in DDQN Algorithm:

- Initialize two Q-networks: an online network (Q_online) and a target network (Q_target).
- Collect experience tuples (state, action, reward, next state) and store them in a replay buffer.
- Sample mini-batches from the buffer for training.
- Compute target Q-values using the target network to reduce overestimation bias.
- Update the online network using a loss function to minimize the error between predicted and target Q-values.
- Periodically update the target network by copying weights from the online network.
- Advantages of DDQN:
- Reduces overestimation bias by separating action selection from evaluation.
- Enhances stability through the use of a target network with delayed updates.
- Improves learning efficiency, leading to faster and more accurate convergence in complex environments.

Due to its stability and effectiveness, DDQN is widely used in reinforcement learning applications, particularly those involving high-dimensional data and intricate decision-making tasks.

I) DDQN

In our research, the Double Deep Q-Network (DDQN) is a crucial element of the proposed approach, which plays a central role in training a multi-agent model to classify webpages as either phishing or legitimate.

Double Q-Learning: DDQN mitigates overestimation bias by using two separate Q-networks. One network selects actions, and the other estimates the Q-values for those actions, resulting in more accurate and stable Q-value estimates.

Agent: The agent uses webpage features (URL, content, DOM structure) to understand the state current. It performs actions, receives rewards, and updates the Q-table, initially set to zero and refined over time to improve the policy.

Action (WP). Actions in the model affect the environment and trigger updates to the Q-values. The available actions depend on factors such as feature vectors, the dataset, and neural network architecture.

State (S). It represents the webpage features and influences agent actions based on changes in the environment.

Policy: The policy (π) guides the agent in choosing actions that maximize rewards based on the current state. As shown in Equation (4), $\pi(s)$ selects the action that maximizes the Q-value, Q(s,a), for the states.

Reward and Learning: The reward (r) is the feedback given to the agent after an action in a specific state, indicating whether if the action was beneficial or detrimental. In the phishing detection task, correct classifications result in positive rewards, while incorrect ones result in negative rewards. Cumulative rewards at time t are calculated by taking the sum of discounted future rewards, as shown in Equation (5), where the discount factor γ determines the importance of future rewards.

Q-Value (Action-Value Function): The Q-value Q(s, a), also known as the action-value function, is crucial in reinforcement learning and the DDQN algorithm. It estimates the expected cumulative reward for taking action in state s and by following the optimal policy thereafter. DDQN learns to estimate Q-values for different actions given the current state, guiding the agent's decisions. Mathematically, the Q-value is defined in Equation (6) where $\boldsymbol{R_C}$ represents the cumulative reward.:

The Goal of DDQN Agents: The primary goal of DDQN agents is to maximize cumulative rewards $\boldsymbol{R_C}$ by determining the optimal Q* function by using an ε-greedy policy. The Q* function represents the maximum achievable rewards for an optimal policy $\pi*$. In the ε-greedy policy, actions are selected randomly from the available options to facilitate learning and maximize reward acquisition based on policy $\pi*$. By applying the optimal policy $\pi*$, the Q* function can be transformed into the optimal classifier model for the experiment, as represented by Equation (7).

This variable "y" is a representation of the expected Q-value, which is calculated using the Bellman equation. It is used to update the Q-values throughout the training process. The Bellman equation computes the expected future reward from a given state-action pair, by taking into account the discounted future rewards and the Q-values of the next state. The equation is as follows (8).

This formula calculates the maximum Q-value for a given state and action. The DQN estimates the optimal action-value function by computing the expected Q-value through the Bellman equation, which helps it perform better in predicting class labels.r_j is the immediate reward obtained by the DDQN after taking action (a) in state (s) at time t.

Loss Function: The loss function (L) measures the difference between the DDQN's estimated Q-values and the target Q-values, calculated using the Mean Squared Error (MSE). The target Q-values are provided by a target network, a stable copy of the Q-Network. The loss function is (9). The target Q-values are computed as (10) where r is the immediate reward, **γ** is the discount factor, s' is the next state, and a' is the action in s''. The DDQN minimizes this loss by updating its parameters, improving its ability to classify webpages as phishing or legitimate.

Exploration vs. Exploitation: The method uses an ε-greedy policy to balance exploration of new actions and exploitation of learned knowledge. During training, the agent selects actions based on Q-values with a probability (ε) for exploration. This probability decreases over time to shifting favor exploitation.

Training and Fine-tuning: The DDQN interacts with the environment (webpage features) to update Q-value estimates and fine-tune its parameters. The process uses backpropagation and gradient descent to minimize the loss function and improve Q-value accuracy. By leveraging DDQN in deep reinforcement learning, this method effectively learns complex patterns from webpage data, enhancing phishing detection accuracy.

$$\pi\,(s)\; = argmax(\,Q(s,a)) \tag{4}$$

$$R_C\; = \sum_{k=1}^{\infty} \gamma^k\,.\,r_{t+k} \tag{5}$$

$$Q^{\pi}\,(\,s\,,a\,) = E_{\pi}[\textstyle\sum_{k=1}^{\infty}\gamma^k\,.\,r_{t+k}\,|(s_t = s, a_t = a)] \tag{6}$$

$$\pi^* = \begin{cases} 1 & a = \; argmax_a Q^*\,(s,a) \\ 0 & otherwise \end{cases} \tag{7}$$

$$y = \begin{cases} r_j & terminal_j = T \\ r_j + \gamma max_{a_{t_1}} Q(s_{t+1}\,, a_{t+1}, \theta_{k-1}\,)), & terminal_j = F \end{cases} \tag{8}$$

$$L(\theta) = \sum_{(s_1\,,a_t,r_t,s_{t+1})\in Bm} ((Q(s,a,\theta_k) - Q_{target}(s,a))^2 \tag{9}$$

$$Q_{target}(s,a,\theta) = r\; +\; \gamma\; *\; max(Q(s_1,a_t,\theta)) \tag{10}$$

J) Classification based on DDQN

In the proposed phishing webpage classification problem, the Double Deep Q-Network (DDQN) is used as a reinforcement learning algorithm for sequential decision-making. Agents representing phishing and legitimate classes are given input vectors of URL, content, and DOM to interact with the environment and make decisions. During the learning process, agents use these input vectors to perform actions and obtain rewards. The agents for the phishing class aim to maximize rewards, whereas those for the legitimate class aim to minimize them. This iterative process helps agents improve their decision-making over time. The state at each time step (t) is defined by a vector space representation of the training dataset (T), comprising three matrices for URL, content, and DOM features. Each matrix has dimensions of ((96+96+20)* WPT), where 212 is the number of feature vectors and WPT is the number of webpages in the dataset. Training samples are updated each episode to expose agents to different scenarios.

The action (a) is binary: 0 or 1, representing the classification of a webpage (ut) as phishing or legitimate. function estimates the likelihood that (ut) is a phishing webpage. If the Q-function value exceeds 0.5, it is normalized to 1 (phishing); if between 0 and <0.5, it is normalized to 0 (legitimate). The reward (R) is feedback received by the agent based on the correctness of its classification. For each action (at), the reward (rt) is defined as follows, based on whether the agent correctly or incorrectly classified (ut) according to its true label (lt) as (11):

$$R = \begin{cases} 1\,, & a_t = l_t \\ -1\,, & a_t \neq l_t \end{cases} \tag{11}$$

Using rewards, agents iteratively update their Q-values through the DDQN algorithm, gradually improving their classification ability for phishing or legitimate webpages based on cumulative rewards. In conclusion, DDQN allows agents to interact effectively with the environment, receive feedback, and learn to make better classification decisions, resulting in an effective and accurate phishing detection system.

K) Training the network

In our proposed phishing detection model using the Double Deep Q-Network (DDQN) algorithm we utilized three separate DDQNs for URL, content, and DOM features. Each DDQN comprises two fully connected neural networks, each with ReLU activation and a softmax output layer for Q-values. a batch size of 64, a learning rate of 0.001, and the Adam optimizer across 1000 epochs. The loss was calculated by comparing predicted Q-values with target Q-values. Random data batches were used to train the DDQNs, with weights updated simultaneously. The reward function was based on prediction accuracy. Cross-validation was performed to enhance accuracy and generalization. By combining DDQNs with CNNs, our method effectively detects phishing webpages, achieving high accuracy and robustness. The DDQN

agents learn optimal Q-values through exploration (ε-greedy strategy) and exploitation, guiding accurate classification decisions. The Figure 2 shows the structure of the three DQNs, and their combined outputs determine the final classification as phishing or legitimate based on the highest confidence.

L) Highest confidence score

After training and extracting the feature vectors outputs from each DDQN(URL, Content, DOM), the highest confidence score as the final output. For example, if the confidence scores are (1, 1, 0), the highest confidence score is 1.

4. Experimental results and analysis

This section summarizes four experiments designed to evaluate the performance of the proposed model in detecting phishing websites. The experiments examine various aspects of the model, including feature extraction, multi-agent approaches, and comparisons with other models.

A) Experimental environment

The development environment used for the experiments consisted of Python 3.5, with PyCharm as the integrated development environment (IDE). The system had 16GB of memory and an Intel Core i7-6700 CPU, providing sufficient computational power for the tasks. Windows 10 was the operating system used, compatible with the necessary machine learning frameworks and libraries.

Table.4.
Evaluation Indicator

Evaluation indicator	***Calculation formula***
Accuracy	(TP+TR)/(TP+FP+TN+FN)
Precision	TP/(TP+FP)
TPR(Recall)	TP/(TP+FN)
F1	(2*Precision*Recall)/(Precision+Recall)
FPR	FP/(TN+FP)

Table.5.
Data Source

Data Source	***Legitimated Urls***	***Phishing Urls***
Phish Storm[37]	48009	47902
Phish Tank[38]	0	178495
Iscx-Url2016 [39]	35378	9965
Kaggle[40]	345738	0

B) Parameter setting

The proposed model utilizes several key parameters to optimize performance. It uses 128 deep neural network cells to capture complex features and a batch size of 64 for efficient sample processing. A dropout rate of 0.5 helps prevent overfitting by deactivating 50% of neurons during training. The model is trained for 100 epochs and 1000 episodes, with a learning rate of 0.001, balancing adaptation speed and stability. The RELU activation function adds nonlinearity, improving feature learning.

Input sizes for URLs, HTML content, and DOM structures are set at 200, 1000, and 2000, respectively. These lengths are chosen to reduce unnecessary padding while preserving relevant information, enhancing the model's ability to detect phishing websites and optimizing computational efficiency.

C) Evaluation metrics

Evaluation metrics commonly used in research include Accuracy, Precision, Recall (True Positive Rate), False Positive Rate (FPR), and F1-measure, as detailed in Table 4. True Positive (TP) and True Negative (TN) indicate correct classifications, while False Positive (FP) and False Negative (FN) represent misclassifications. The F1 score combines Precision and Recall to assess overall performance effectively.

D) Dataset

Data plays a crucial role in the performance of machine learning models, with both the quality and quantity being critical factors [41]. In our study, data collection was organized into two main stages: gathering data from various sources and processing and storing it. We used several open datasets, including the Phish Storm dataset [37], which contains 96,018 URLs (48,009 legitimate and 48,009 phishing URLs), and the ISCX-URL2016 dataset [39], which includes 35,378 legitimate and 9,965 phishing URLs. Additionally, approximately 490,000 legitimate URLs were obtained from an open Kaggle project [40], supplemented by daily updates from the Phish Tank platform[38].Data processing included cleansing, removing duplicates, and standardizing URL formats to ensure a balanced representation of legitimate and phishing URLs. This approach helped prevent bias and ensured the model could adapt to evolving phishing tactics. The extensive and diverse dataset contributed to better generalization, improving performance in phishing detection by reducing false positives and negatives. The data assessment in Table 5 reflects critical aspects related to data sourcing, processing, and integrity, which directly impacted the model's performance.

E) Compared methods

To validate the efficacy of the proposed model, a comparison study was conducted between The proposed model and traditional phishing webpage detection methods. Several deep learning-based approaches have been developed for phishing detection. PhishDet[28] employs a Long-term Recurrent Convolutional Network and Graph

Convolutional Network to analyse URL and HTML features. The RNN-GRU model [7] utilizes a CNN to extract features from website screenshots and applies a Long Short-Term Memory (LSTM) network for classification. WEB2VEC [12]integrates Convolutional Neural Network (CNN) for local feature extraction and Bidirectional Long Short-Term Memory (BiLSTM) for global semantic feature representation. Hybrid DLM [42] combines deep LSTM and CNN networks, with the LSTM analysing URL data and a separate CNN processing HTML features. URLNet [43] uses CNNs with character-level and word-level embedding's to automatically extract features from URLs, while MPURNN [44] employs a CNN for character-level embedding and an LSTM for additional feature extraction, demonstrating the diversity and effectiveness of deep learning techniques in phishing detection. The comparison demonstrates the strengths and approaches of each model, highlighting the advancements and unique contributions in phishing webpage detection techniques.

In evaluating the feature extraction methods in the proposed model, we considered several deep learning networks including CNN, RNN, and LSTM, as well as hybrid networks such as CNN-RNN, CNN-LSTM, and CNN-BiLSTM. However, it's worth noting that we did not compare traditional supervised machine learning methods like Sequential Minimal Optimization (SMO), Bayesian Network (BN), Support Vector Machine (SVM), and AdaBoost in our experiments.

Table.6.
Detection effects of different feature combinations

CNN-LSTM	***0.9695***	***0.9698***	***0.9628***	***0.9651***	***0.0032***
CNN- RNN	0.9918	0.9869	0.0059	0.9915	0.0104
LSTM	0.9654	0.9924	0.9325	0.9616	0.0061
RNN	0.9665	0.9922	0.9351	0.9628	0.0063
CNN	0.9786	0.9844	0.9756	0.9800	0.0085

Table.7.
Performance of various feature extraction models in detecting phishing webpages is evaluated

Aalgorithm	***Accuracy***	***Precision***	***TPR(Recall)***	***F1***	***FPR***
DDQN	0.9921	0.9905	09825	0.9930	0.0022
CNN-BLSTM	0.9905	0.9869	0.9826	0.9908	0.0025
CNN-LSTM	0.9695	0.9698	0.9628	0.9651	0.0032
CNN- RNN	0.9918	0.9869	0.0059	0.9915	0.0104
LSTM	0.9654	0.9924	0.9325	0.9616	0.0061
RNN	0.9665	0.9922	0.9351	0.9628	0.0063
CNN	0.9786	0.9844	0.9756	0.9800	0.0085

Table.8.
Detection effects of different feature combinations

FPR	***F1***	***TPR (Recall)***	***Precision***	***Accuracy***	***Feature combination***
0.0022	0.9930	09825	0.9905	0.9921	URL+HTML+DOM
0.0127	0.9914	0.9913	0.9919	0.9915	URL+HTML
0.0080	0.9887	0.9834	0.9812	0.9870	HTML+ DOM
0.0026	0.9710	0.9758	0.9798	0.9790	URL+ DOM
0.0030	0.9514	0.9700	0.9767	0.9712	HTML
0.0221	0.9363	0.9733	0.9978	0.9372	DOM
0.0259	0.9416	0.8800	0.8670	0.9010	URL

Table.9.
Detection effect of multiagent

Agent	***Accuracy***	***Precision***	***TPR (Recall)***	***F1***	***FPR***
Multi Agent	0.9921	0.9905	09825	0.9930	0.0022
Single Agent	0.9110	0.8670	0.8811	0.8731	0.0030

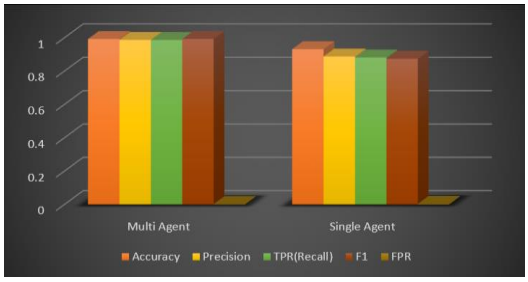

Fig. 2. Detection effect of multi-agent

F) Experiment 2: The effectiveness of DDQN with additional features

Experiment 2 investigates the impact of the feature extraction process on the proposed model by using a Double Deep Q-Network (DDQN). In this experiment, various models, including CNN-BILSTM, CNN-LSTM, CNN-RNN, LSTM, RNN, and CNN, are used in place of DQN in the proposed model. The results of this comparison are summarized in Table 7.the findings show that the DDQN network outperforms CNN-BILSTM in terms of classification detection, suggesting that incorporating additional feature extraction improves performance in multi-agent scenarios. Specifically, the DDQN model achieves a False Positive Rate (FPR) of 0.0022, an F1 score of 0.9930, a True Positive Rate (Recall/TPR) of 0.9825, Precision of 0.9905, and an overall Accuracy of 0. 9921.In comparison, CNN-BILSTM performs slightly worse with an FPR of 0.0025, an F1 score of 0.9908, a recall of 0.9826, precision of 0.9869, and an accuracy of 0.9905. Other models, such as CNN-LSTM, CNN-RNN, LSTM, RNN, and CNN, show varying performance. CNN-LSTM has a higher FPR (0.0104) and lower recall (0.0059), while CNN-RNN and LSTM models perform similarly with lower F1 scores and accuracy compared to DDQN. these results indicate that DDQN with additional features is more effective in phishing webpage detection than the other models tested, highlighting the importance of feature extraction in improving model performance.

G) Experiment 3: Effects of multi-faceted feature learning

Experiment 3 assessed the impact of learning features from different webpage components, including URL, page content, and DOM structure. The results, shown in Table 8, reveal that the best detection performance is achieved by combining all three features: URL, page content, and DOM

structure. This combination yields an FPR of 0.0022, an F1 score of 0.9930, a recall of 0.9825, precision of 0.9905, and accuracy of 0. 9921.when combining two features, such as URL + HTML, the model still performs well, with an FPR of 0.0127 and an F1 score of 0.9914. However, using only a single feature, like HTML or DOM, results in lower performance, indicating that combining multiple features enhances detection accuracy. Overall, the experiment shows that incorporating multiple sources of information significantly improves phishing webpage detection.

H) Experiment 4: Evaluating the effectiveness of a multiagent approach

Experiment 4 evaluated the proposed model's multi-agent (3 DDQN) approach compared to a single-agent model. Results in Table 9 show that the multi-agent model achieved superior performance, with faster, more stable training and testing. The multi-agent model recorded an FPR of 0.0022, F1 score of 0.9930, recall of 0.9825, precision of 0.9905, and overall accuracy of 0.9921, outperforming the single-agent model (FPR: 0.0030, F1: 0.8731, recall: 0.8811, precision: 0.8670, accuracy: 0.9110). These results confirm that the proposed model effectively represents webpages and improves classification by leveraging multi-agent deep learning and additional feature extraction, showcasing excellent prediction performance. Figure 2 illustrated that the training and testing processes in the multi-agent model were faster and more stable

Figure 3 highlights the training and testing accuracy and loss trends for single-agent and multi-agent models over multiple epochs. The accuracy graphs show a steady increase, reaching high stability, while the loss graphs display rapid decreases in early epochs, followed by stabilization. These trends demonstrate the multi-agent model's effective learning, rapid convergence, and strong generalization from training to unseen data.

5. Conclusion

The study introduces the proposed model, an automated framework for detecting phishing webpages using a multi-agent deep reinforcement learning approach with additional features. This method leverages NLP-based representation learning techniques to extract a comprehensive webpage representation from various aspects, such as URL, page content, and DOM structure. A multi-channel, multi-agent deep learning network is utilized to identify and extract deep hidden features, with influential features weighted more heavily in classification predictions. Results from four experiments demonstrate that the proposed model outperforms existing advanced phishing detection techniques, achieving an impressive accuracy of 99.21% and a false positive rate as low as 0.22%. Overall, this approach shows great potential to enhance web security, empowering users and organizations to effectively identify and prevent phishing attacks. Future work could involve optimizing the model by adding new features or integrating more data to improve performance in dynamic environments. Additionally, extending this approach to address more complex threats and enable phishing detection across multi-lingual settings could further strengthen the system's capabilities.

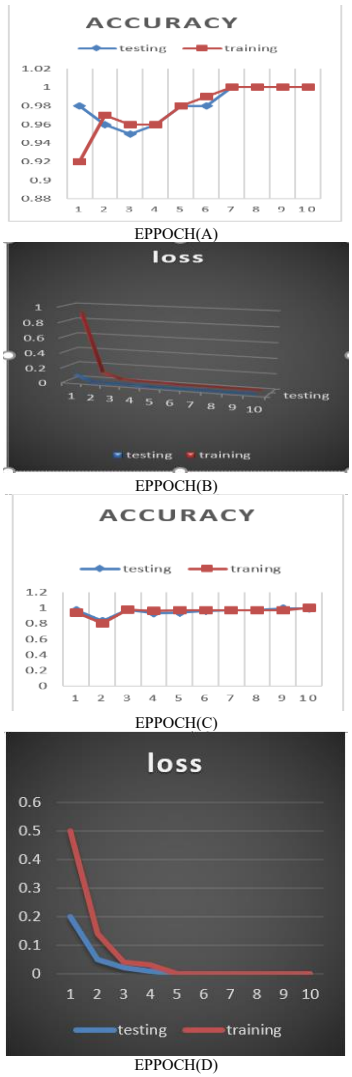

Fig. 3. The impact of multi-agent deep learning

References

[1] Opara, C., Y. Chen, and B. Wei, Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics. Expert Systems with Applications, 2024. 236: p. 121183.

[2] Wang, M., et al., Phishing webpage detection based on global and local visual similarity. Expert Systems with Applications, 2024. 252: p. 124120.

[3] Sánchez-Paniagua, M., et al., Phishing URL detection: A real-case scenario through login URLs. IEEE Access, 2022. 10: p. 42949-42960.

[4] Roy, S.S., et al., Multimodel Phishing URL Detection Using LSTM, Bidirectional LSTM, and GRU Models. Future Internet, 2022. 14(11): p. 340.

[5] Purwanto, R.W., et al., PhishSim: Aiding Phishing Website Detection with a Feature-Free Tool. IEEE Transactions on Information Forensics and Security, 2022. 17: p. 1497-1512.

[6] Ozcan, A., et al., A hybrid DNN–LSTM model for detecting phishing URLs. Neural Computing and Applications, 2021: p. 1-17.

[7] Tang, L. and Q.H. Mahmoud, A Deep Learning-Based Framework for Phishing Website Detection. IEEE Access, 2021. 10: p. 1509-1521.

[8] Moghimi, M. and A.Y. Varjani, New rule-based phishing detection method. Expert systems with applications, 2016. 53: p. 231-242.

[9] Cao, J., et al., A phishing web pages detection algorithm based on nested structure of earth mover's distance (Nested-EMD). Chinese Journal of Computers, 2009. 32(5): p. 922-929.

[10] Rao, R.S. and A.R. Pais, Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach. Journal of Ambient Intelligence and Humanized Computing, 2020. 11(9): p. 3853-3872.

[11] Liang, B., et al. Cracking classifiers for evasion: A case study on the google's phishing pages filter. in Proceedings of the 25th International Conference on World Wide Web. 2016.

[12] 1, J.F., et al., Web2Vec: Phishing Webpage Detection Method Based on Multidimensional Features Driven by Deep Learning. 2020.

[13] Vrbančič, G., I. Fister Jr, and V. Podgorelec. Swarm intelligence approaches for parameter setting of deep learning neural network: case study on phishing websites classification. in Proceedings of the 8th international conference on web intelligence, mining and semantics. 2018.

[14] Yang, P., G. Zhao, and P. Zeng, Phishing website detection based on multidimensional features driven by deep learning. IEEE access, 2019. 7: p. 15196-15209.

[15] Korkmaz, M., O.K. Sahingoz, and B. Diri. Feature selections for the classification of webpages to detect phishing attacks: a survey. in 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). 2020. IEEE.

[16] El Aassal, A., et al., An in-depth benchmarking and evaluation of phishing detection research for security needs. IEEE Access, 2020. 8: p. 22170-22192.

[17] Elsadig, M., et al., Intelligent Deep Machine Learning Cyber Phishing URL Detection Based on BERT Features Extraction. Electronics, 2022. 11(22): p. 3647.

[18] Chatterjee, M. and A.-S. Namin. Detecting phishing websites through deep reinforcement learning. in 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC). 2019. IEEE.

[19] Feng, J., L. Zou, and T. Nan, A phishing webpage detection method based on stacked autoencoder and correlation coefficients. Journal of computing and information technology, 2019. 27(2): p. 41-54.

[20] Basit, A., et al., A comprehensive survey of AI-enabled phishing attacks detection techniques. Telecommunication Systems, 2021. 76(1): p. 139-154.

[21] Chen, W., W. Zhang, and Y. Su. Phishing detection research based on LSTM recurrent neural network. in International Conference of Pioneering Computer Scientists, Engineers and Educators. 2018. Springer.

[22] Douzi, S., M. Amar, and B. El Ouahidi. Advanced phishing filter using autoencoder and denoising autoencoder. in Proceedings of the International Conference on Big Data and Internet of Thing. 2017.

[23] Zhang, X., et al. Boosting the phishing detection performance by semantic analysis. in 2017 ieee international conference on big data (big data). 2017. IEEE.

[24] Bu, S.-J. and H.-J. Kim. Learning Disentangled Representation of Web Address via Convolutional-Recurrent Triplet Network for Classifying Phishing URLs. in 2021 International Conference on Electronics, Information, and Communication (ICEIC). 2021. IEEE.

[25] Gualberto, E.S., et al., The answer is in the text: multi-stage methods for phishing detection based on feature engineering. IEEE Access, 2020. 8: p. 223529-223547.

[26] Xiao, X., et al., Phishing websites detection via CNN and multi-head self-attention on imbalanced datasets. Computers & Security, 2021. 108: p. 102372.

[27] Louati, F. and F.B. Ktata, A deep learning-based multi-agent system for intrusion detection. SN Applied Sciences, 2020. 2(4): p. 1-13.

[28] Ariyadasa, S., S. Fernando, and S. Fernando, Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML. IEEE Access, 2022. 10: p. 82355-82375.

[29] Mohammad, R.M., F. Thabtah, and L. McCluskey. An assessment of features related to phishing websites using an automated technique. in 2012 international conference for internet technology and secured transactions. 2012. IEEE.

[30] Louati, F. and F.B. Ktata, A deep learning-based multi-agent system for intrusion detection. SN Applied Sciences, 2020. 2(4): p. 675.

[31] Nguyen, T.T., et al., A multi-objective deep reinforcement learning framework. Engineering Applications of Artificial Intelligence, 2020. 96: p. 103915.

[32] Nguyen, T.T., N.D. Nguyen, and S. Nahavandi, Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications, in IEEE transactions on cybernetics. 2020. p. 3826-3839.

[33] Sartoli, S. and A.S. Namin. A semantic model for action-based adaptive security. in Proceedings of the Symposium on Applied Computing. 2017.

[34] Jiang, F., et al., A reinforcement learning-based computing offloading and resource allocation scheme in F-RAN. EURASIP Journal on Advances in Signal Processing, 2021. 2021: p. 1-25.

[35] Sutton, R.S. and A.G. Barto, Reinforcement learning: An introduction. 2018: MIT press.

[36] Du, W. and S. Ding, A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. Artificial Intelligence Review, 2021. 54(5): p. 3215-3238.

[37] Wang, W., et al., PDRCNN: Precise phishing detection with recurrent convolutional neural networks. Security and Communication Networks, 2019. 2019(1): p. 2595794.

[38] PhishTank > See All Suspected Phish Submissions. Accessed: Oct. 20, 2021. www.phishtank.com.[Online]. Available: https://www.phishtank.com/phish_archive.php.

[39] URL 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Accessed: Oct. 20, 2021. www.unb.ca. [Online]. Available: https://www.unb.ca/cic/datasets/url-2016.html.

[40] Kumar., S., Malicious and Benign URLs.kaggle.com. 2019.

[41] Gupta, N., et al. Data quality for machine learning tasks. in Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining. 2021.

[42] Ariyadasa, S., S. Fernando, and S. Fernando, Detecting phishing attacks using a combined model of LSTM and

CNN. International Journal of Advanced And Applied Sciences, 2020. 7(7): p. 56-67.

[43] Le, H., et al., URLNet: Learning a URL representation with deep learning for malicious URL detection. arXiv preprint arXiv:1802.03162, 2018.

[44] Bahnsen, A.C., et al. Classifying phishing URLs using recurrent neural networks. in 2017 APWG symposium on electronic crime research (eCrime). 2017. IEEE.