Adaptive Phishing Detection on Webpages via Multi-Agent Deep Learning and Multi-Dimensional Features

Ziba Jafari[a], Seyyed Hamid Ghafouri [a,*], Mohammad Ahmadinia[a]

[a]*Department of Computer Engineering, Kerman branch, Islamic Azad University, Kerman, Iran*

**e-mail: sh,ghafouri@iau.ac.ir*

Abstract

The challenge of detecting and preventing phishing attacks is becoming increasingly difficult due to their growing sophistication. Phishing is a social engineering tactic where users are tricked into revealing sensitive information, such as passwords, credit card details, and usernames, through fake websites that appear legitimate. Traditional detection techniques often fail to keep up with the dynamic and evolving nature of phishing tactics, necessitating more adaptive and complex approaches. This study introduces a novel method for identifying phishing websites through a multi-agent deep learning approach. The method uses three deep learning networks, each trained to analyze specific features of websites, including the URL, page content, and the Document Object Model structure. A highest confidence score mechanism is then applied to combine the results of these networks and produce a final prediction. The results demonstrate that this approach outperforms current advanced phishing detection methods, achieving an accuracy rate of 99.20% and a false positive rate of just 0.20%. This proposed method not only excels at identifying known phishing sites but also shows strong performance in detecting previously unseen sites, making it highly adaptable to emerging phishing techniques. The approach has significant potential to improve web security by assisting both users and organizations in detecting and preventing phishing attacks. Furthermore, this framework highlights the potential of deep reinforcement learning in cybersecurity, paving the way for more resilient and automated security systems to combat the growing threat of cyberattacks.

Keywords: Deep Learning; Phishing; Representation Learning; Multi Agent Deep Reinforcement Learning (MADRL)

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],[4]. 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[5],[6],[7].Machine learning techniques like Decision Trees, Random Forests, and Support Vector Machines have been used for phishing detection but struggle with unseen websites[8],[9]. Deep learning methods, such as CNNs and RNNs, offer improved performance but require extensive data and computational resources [10]. Existing approaches in phishing detection rely on URL-based, heuristic-based, visual similarity-based, and machine learning methods [11],[12],[13]. 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 multi-agent deep learning method for phishing detection using three Deep Q-Networks (DQNs) to analyze URL, HTML, and DOM features. An attention mechanism selectively combines outputs based on performance, enhancing accuracy, precision, recall, and F1 score. The approach outperforms existing methods and effectively detects previously unseen phishing websites [12]. By leveraging deep reinforcement learning, the model continuously adapts and improves, offering a robust and comprehensive solution to phishing threats across individuals, businesses, and governments. 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 DQN.
- Conducting four MADRL- Phish experiments 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 MADRL- Phish are

outlined in Section III. In Section IV, the performance of MADRL-Phish is evaluated. Finally, the paper concludes with a discussion of future work.

2. RELATED WORKS

Phishing website detection remains a significant challenge in cybersecurity [14]. 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 [15],[16].

Recent studies [17] have explored various features for phishing detection, generally classified into URL-based, content-based, and DOM-based features. URL features, like length, number of dots, and keywords, help distinguish phishing from legitimate sites. Content features, including keyword analysis, multimedia presence, and text-to-HTML ratio, provide additional insights, while DOM features, such as hidden fields, form tags, and suspicious scripts, aid in identifying malicious webpages. Combining traditional and deep learning methods has further improved adaptability and detection accuracy, highlighting opportunities to enhance phishing detection systems.

2.1. Traditional Phishing Webpage Detection Methods

Phishing detection methods are commonly divided into four categories. Blacklist-based approaches compare URLs with blacklists, yielding low false positives but missing new sites, such as those detected in Google Chrome [12]. Benchmarking frameworks like PhishBench provide controlled testing [18]. Heuristic methods apply predefined rules but suffer from high false positives [12],[13], while visual similarity-based techniques transform webpages into images for analysis [19]. Hybrid methods, e.g., cosine similarity with neural networks [20]and SPWalk with network embedding [21], further improve detection. The fourth category, machine learning–based methods, leverages URL, Whois/DNS, and content features [17], [22],[23],[24],[25]; advanced systems use thousands of features, with Chrome applying over 2,130. Recent advances include PhishHaven [26], PSO-based feature weighting [27], HinPhish [28], and meta-learners[29]. Despite progress [11], evolving phishing tactics demand continuous refinement [15],[16].

2.2. Phishing webpage detection methods based on deep learning techniques

Deep learning has surpassed traditional methods in phishing detection [10], [15]. Approaches are generally divided into hand-crafted feature engineering (manual features such as URL, domain, content fed into DNNs [30]), automatic feature learning (directly extracting features from raw data [12],[13]), and hybrid methods combining both [24]. URL-based models apply LSTM, CNN, and autoencoders [30], [31], while content-based methods leverage embeddings like Word2Vec [12] and transformer models such as BERT [32]. Recent studies [33],[34], [35],[36],[37] explored CNNs, GANs, and CNN–LSTM hybrids, and frameworks like PhishDet integrated recurrent and graph convolutional models [30]. However, most models depend on single features, reducing robustness [15]. To address this, ensemble and multi-agent frameworks enhance generalization [38], [30]. Building on this, MADRL-Phish employs three DQNs (URL, content, DOM) aggregated via majority voting [24], [38], outperforming both traditional [10],[15] and deep learning methods [30] against zero-day attacks (Table 1).

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 MADRL-Phish framework and its key technologies.

3.1. 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 $N = (WP_1, ...WP_i, \ldots, WP_n)$, where WP_i refers to the i-th webpage with i ranging from 1 to n. Each webpage consists of three components, $WP_i = (U_i, H_i, \ldots, D_i)$, 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 $(WP_1, X_1, Y_1), (WP_2, X_2, Y_2) \ldots (WP_T, X_T, Y_T)$, where: For $i = 1, 2, \ldots, T$, WP denotes a webpage in the given training set T.

• The features of webpage WP_i are denoted by X_i, 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 $Yi \in \{0, 1\}$, where $Y_i = 0$ represents a benign webpage and $Y_i = 1$ represents a phishing webpage. The goal of MADRL- Phish is to learn a discriminant model $f: X \rightarrow Y$ using the training dataset, which can then be used to classify new webpages as phishing or benign

3.2. The Overall Framework

The MADRL-Phish 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 DQN network to extract features. Finally, the combined features are classified as phishing or benign. MADRL-Phish 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.

3.3. Web Page Reprocessing

Webpage reprocessing enhances feature representation by constructing multiple corpora. While prior studies mainly relied on URLs for phishing detection [12], URLs alone lack structural and semantic details. To address this, MADRL-Phish learns from three sources: URL, page content, and DOM structure. URLs are analyzed at both character and word levels, with character-level encoding using OneHot vectors and word-level segmentation based on structural elements (e.g., protocol, path). Webpage content is processed into word- and sentence-level corpora after removing non-textual elements, while the DOM is represented as a hierarchical sequence of HTML tags parsed via a breadth-first approach.

3.4. Webpage Representation

One-Hot encoding is limited by sparsity and lack of semantic relationships, leading to the use of word embeddings [12]. MADRL-Phish adopts a lightweight approach, embedding the One-Hot matrix into dense word vectors through a single-layer neural network instead of relying on pre-trained models like Word2Vec. This process generates a representation matrix S, which is

jointly optimized with feature extraction and classification components via backpropagation. Matrix S not only reduces computational cost and memory usage but also improves semantic representation. The same strategy is applied to URL (character- and word-level) and DOM corpora for learned low-dimensional representations. Suppose we have a URL character-level corpus U, and we want to learn the representation for the i-th URL, denoted as $"U_i"$. $"U_i"$ is encoded using One-Hot encoding. Let's say the j-th character is represented by vector g_j after One-Hot encoding, where $g_j = \left(g_{j1}, g_{j2}, \ldots, g_{jm} \right)^T$. The matrix G has dimensions of $m * n$, where each column represents a character g_j of the URL $"U_i"$. To obtain the URL character embedding, each URL $"U_i"$ is first represented as a One-Hot matrix G with dimensions $G_{m*n} = \left(g_1, g_2, \ldots, g_n \right)$. 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 $R^{p \times m}$, 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:

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

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):

$$\boldsymbol{x}_{i(\mathbf{URL})} = \left(\boldsymbol{U}_i^c, \boldsymbol{U}_i^w \right), \boldsymbol{x}_{i(\mathbf{CONTENT})} = \left(\boldsymbol{C}_i^w, \boldsymbol{C}_i^s \right), \ \boldsymbol{x}_{i(\mathbf{DOM})} = \left(\ \boldsymbol{D}_i \right) \quad \textbf{(2)}$$

The representation URL vectors , (U_i^c and U_i^w) are the representation vector learned from U_i. The CONTENT vectors (C_i^w , C_i^s) are vectors learned from the page content, and DOM vector (D_i) is learned from the DOM structure. To obtain the character-level representation 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 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 C_i^w and the sentence-level representation vector C_i^s, statistical features are extracted from the page content.

On the other hand, for the DOM structure vector 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 MADRL-Phish, 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 48 additional features below, including 16 features related to URL, 16 features related to HTML content, and 16 features related to DOM, for better learning of the learning network.

3.5. Additional Feature Extraction

The proposed approach extracts feature representations from the URL, content, and DOM, each with an additional 16-dimensional vector space to enhance the distinction between benign and phishing webpages. URL-based features include 16 attributes, such as lengthy URLs and the presence of IP addresses, aiding in phishing detection [1],[9],[17],[39]. Content-based features analyze multimedia elements like images and JavaScript for high-accuracy detection despite potential vulnerabilities. Similarly, 16 DOM tree features, derived from [17], provide structural insights for effective detection. Tables 2-4 outline these features.

3.6. 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 $S(x_i)$ is defined as shown in Eq. (3).The variables U_i^f , C_i^f *and* D_i^f represent additinal features that are obtained through learning from the URL, content, and DOM structure.

$$\boldsymbol{x}_{i(\mathrm{URL})} = \left(\boldsymbol{U}_i^c, \boldsymbol{U}_i^w, \boldsymbol{U}_i^f\right), \quad \boldsymbol{x}_{i(\mathrm{DOM})} = \left(\boldsymbol{C}_i^w, \boldsymbol{C}_i^s, \boldsymbol{C}_i^f\right) \ , \ \boldsymbol{x}_{i(\mathrm{DOM})} = \left(\boldsymbol{D}_i, \boldsymbol{D}_i^f\right) \tag{3}$$

3.7. Background Of Deep Reinforcement Learning: Multi Agent

A multi-agent system enables agents to interact with each other and the environment, with deep reinforcement learning (DRL) widely used for solving complex cooperative tasks [38], [40]. In this context, MADRL extends the Markov Decision Process (MDP) to a stochastic game where agents' actions affect system states and rewards [41]. For phishing detection, MADRL leverages features from URL, content, and DOM encoded as vectors to classify webpages as benign or phishing, optimizing detection accuracy through deep learning.

3.8. The Paradigm Of Deep Reinforcement Learning

Reinforcement Learning (RL) enables an agent to maximize cumulative rewards through interaction with an environment and feedback [25],[42], with applications across diverse domains [43]. In deep reinforcement learning, multiple Deep Q-Networks (DQNs) can act as agents using different features (URL, content, DOM) to detect phishing. Each agent's action space reflects classification decisions, with rewards defined by prediction accuracy. In the proposed method, webpage classification is determined by the highest confidence score among three agents. Using the DQN framework—states, actions, policy, rewards, discount factor (γ), and transition probabilities—agents iteratively optimize Q-values to improve detection accuracy.

3.9. The classifier based on Deep Reinforcement Learning

In our study, a deep reinforcement learning–based classifier is designed with three inputs: URL, content, and DOM. It employs three deep neural networks as agents, each interacting with the environment by receiving a state (s) and selecting an action (a) according to the policy (π), which defines the probability of taking an action given a state (Eq. 4). Agents aim to maximize cumulative rewards (Rc), computed as the discounted sum of future rewards (Eq. 5), where the discount factor $\gamma \in [0, 1]$ balances short- and long-term gains. The Q-value for a state-action pair (s, a) is updated using the Q-function, which estimates the expected reward based on the current state and action (Eq. 6)

$$\pi(a \mid s) = \Pr(a_t = a \mid s_t = s) \quad (4)$$

$$R_C = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^k . r_{t+k} = \sum_{k=1}^{\infty} \gamma^k . r_{t+k} \quad (5)$$

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

In reinforcement learning (RL), agents aim to maximize cumulative rewards (*Rc*) by finding the optimal *Q** function through an ε-greedy policy, where actions are randomly selected to support learning (Eq. 7). The *Q** function represents the maximum achievable reward for an optimal policy (π^*). For small state spaces, Q-values can be stored in tables, but for high-dimensional cases, Deep Q-Networks (DQNs) approximate *Q** via gradient descent, using experience replay and mini-batch sampling with a loss function *L(θ)* (Eq. 8). The predicted Q-value, denoted as y, is computed using the Bellman Eq. (Eq. 9) to update *Q** by incorporating discounted future rewards and next-state estimates. This process enables DQNs to improve class label predictions, with terminal conditions (T/F) marking whether the maximum cumulative reward has been reached.

$$\pi^* = \begin{cases} 1 & a = \mathbf{argmax}_a Q^*(s, a) \\ 0 & otherwise \end{cases} \quad (7)$$

$$L(\theta) = \sum_{(s_1, a_t, r_t, s_{t+1}) \in Bm} (y - Q(s, a, \theta_k))^2 \quad (8)$$

$$y = \begin{cases} r_j & terminal_j = T \\ r_j + \gamma \mathbf{max}_{a_{t_1}} Q(s_{t+1}, a_{t+1}, \theta_{k-1})), & terminal_j = F \end{cases} \quad (9)$$

3.10. Classification based on Deep Reinforcement Learning

In the proposed phishing classification task, RL agents use input vectors of URL, content, and DOM. Each episode updates training samples, with states defined by a vector space *T* containing three matrices of size ((96+96+16) × WP_T), where 208 is the number of feature vectors and WP_T the number of webpages. Actions A are binary {0,1}, representing Phishing or Benign. The Q-function determines classification: values ≥0.5 are normalized to 1, and <0.5

to 0. The reward (*R)* provides feedback on the agent's performance, indicating correct or incorrect classification. If l_t is the true label of u_t, R is defined by Eq. (10).

$$\boldsymbol{R} = \begin{cases} 1\,, & \boldsymbol{a_t} = \boldsymbol{l_t} \\ -1\,, & \boldsymbol{a_t} \neq \boldsymbol{l_t} \end{cases} \quad \textbf{(10)}$$

3.11. Training The Network

Our model employs three DQNs for URL, content, and DOM features, each with two fully connected layers (ReLU) and a softmax output. Training used batch size 64, learning rate 0.001, Adam optimizer, and 100 epochs. Predicted and target Q-values were compared to compute loss, with random batches processed and the highest Q-value selected. All DQN weights were updated after each batch. The reward function reflected label prediction accuracy. The architecture is shown in Figure 2, and training/classification follows Algorithm 1, based on DQN learning [25], [43]. Actions are chosen via an ε-greedy policy

3.12. Highest Confidence Score

After training and extracting the feature vectors outputs from each DQN (URL, Content, Dom), the highest confidence score as the final output. For example, the confidence scores are (1, 1, 0), so the highest confidence score is 1.The algorithm 1 randomly selects an action using Eq. (7) with probability a, and chooses the action that yields the highest cumulative reward over the learning epochs.

Algorithm 1: The algorithm used for both training and classification

Input: URL, CONTENT, DOM samples and their class labels
Output: Optimum Q-values
Initialize experience memory M
Initialize Q-value function with random weights θ
Define the number of episodes K
Initialize target Q-value function with random weights $\theta^* = \theta$;
For each episode k=1 to K do the following:
a. Initialize state s_1 = (URL, CONTENT, DOM)
b. Initialize pre-processed sequence function $\varphi_1 = \varphi(s_1)$;
c. For each time step t=1 to T do the following:
i. Perform an action a_t based on ε-greedy strategy:
- With probability ε, select a random action
- Otherwise, select the action a_t that maximizes $Q(\varepsilon(s_t), a; \theta)$
ii. Observe the reward r_t for action a_t
iii. Set the next state s_{t+1} = (URL, CONTENT, DOM), a_t, v_{t+1};
iv. Set pre-processed sequence function $\varphi_{t1+1} = \varphi(s_{t+1})$
v. Save (φ_{t1}, a_t, r_t, φ_{t1+1}, $terminal_j$) in experience memory M
vi. Randomly sample from M and set y_j using Eq. 9
vii. Perform gradient descent using Eq. 8
viii. If the terminal state is reached (i.e., $terminal_t$ = True), break out of the loop
End of episode
End of the algorithm.

4. EXPERIMENTAL RESULTS AND ANALYSIS

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

4.1. 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.

4.2. Parameter Setting

The MADRL-Phish 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.

4.3. 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 5. 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.

4.4. Dataset

Data plays a crucial role in the performance of machine learning models, with both the quality and quantity being critical factors [48]. 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 [44], which contains 96,018 URLs (48,009 legitimate and 48,009 phishing URLs), and the ISCX-URL2016 [46], which includes 35,378 legitimate and 9,965 phishing URLs. Additionally, approximately 490,000 legitimate URLs were obtained from an open Kaggle project [47], supplemented by daily updates from the Phish Tank platform[45].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 6 reflects critical aspects related to data sourcing, processing, and integrity, which directly impacted the model's performance.

4.5. MADRL-Phish Using Varied Datasets

In this experiment, multiple datasets were evaluated using the DQN classifier, with particular focus on KPT and ISCX due to their unique strengths. The KPT dataset, compiled from Kaggle, Phish Storm, and Phish Tank, provides diverse real-world phishing and legitimate URLs and is precisely balanced, preventing bias and ensuring reliable evaluation.

The ISCX dataset achieved the highest accuracy, while KPT-14 delivered the best overall performance (99.20% accuracy, 99.11% F1 score), as shown in Table 7, making it highly effective for phishing detection. The assessment also considered false positive and false negative rates, and a 0.75:0.25 train-test split ensured both robust training and reliable validation. Overall, using KPT and ISCX achieved a strong balance of diversity, accuracy, and error reduction, confirming the effectiveness of MADRL-Phish.

4.6. Differential Analysis

MADRL-Phish demonstrates competitive performance compared to deep learning methods like CNNs, RNNs, and hybrid models. Table 8 shows LSTM slightly outperforms MADRL-Phish in accuracy (99.57% vs. 99.20%), but MADRL-Phish offers a superior balance of precision and recall, reducing false positives and negatives. By dynamically extracting features from URL, HTML, DOM, and 48 complementary features, it effectively detects novel and zero-day phishing attacks. Its multi-agent architecture enhances computational efficiency, enables parallel processing, and adapts to evolving phishing strategies. Cross-dataset validation on KPT and ISCX confirms robust generalization. Using the same datasets as Tables 7 and 8 ensures fair comparison, highlighting MADRL-Phish's combination of accuracy, robustness, generalization, and operational efficiency, making it suitable for real-world deployment and future enhancements such as ensemble learning and real-time threat intelligence integration.

4.7. Compared Methods

To validate MADRL-Phish, we compared it with several deep learning-based phishing detection methods. PhishDet [30] uses a Long-term Recurrent Convolutional Network and Graph Convolutional Network to analyze URL and HTML features. RNN-GRU [10] applies a CNN to extract features from website screenshots, followed by an LSTM for classification. WEB2VEC [12] integrates CNN for local feature extraction and BiLSTM for global semantic representation. Hybrid DLM [31] combines LSTM analyzing URL data with a separate CNN for HTML features. URLNet [53] employs CNNs with character- and word-level embeddings to extract features from URLs automatically. MPURNN [54]uses CNN for character-level embeddings and LSTM for additional feature extraction. MADRL-Phish was evaluated using CNN, RNN, LSTM, and hybrid networks such as CNN-RNN, CNN-LSTM, and CNN-BiLSTM. Traditional ML methods like SMO, BN, SVM, and AdaBoost were not compared.

4.8. Experiment and Results Evaluation

This section summarizes four experiments conducted to evaluate the MADRL-Phish model's performance in phishing website detection. The experiments focus on different aspects of the model's functionality, such as feature extraction, multi-agent approaches, and comparison with other models.

4.8.1. Experiment 1: MADRL-Phish model Detection Effect

The first experiment, referred to as Experiment 1, focuses on evaluating the effectiveness of the MADRL-Phish model in detecting phishing webpages. This was done by comparing MADRL-Phish with several other classic phishing webpage detection methods, including PHISH_DET, RNN-GRU, WEB2VEC, Hybrid DLM, URLNet, and MPURNN. The results, shown in Table 9, indicate that MADRL-Phish outperforms WEB2VEC and RNN-GRU in overall performance, with an FPR of 0.0020, recall of 0.9844, precision of 0.9902, and accuracy of 0.9920. However, MADRL-Phish performs slightly worse than WEB2VEC when additional features are included, likely due to data limitations in deep learning networks.

Methods like URLNet and MPURNN, which rely on a single deep learning network focusing mainly on URLs, show suboptimal performance. URLNet, for example, has an FPR of 0.0031,

recall of 0.8503, and accuracy of 0.9633, while MPURNN has an FPR of 0.0374 and accuracy of 0.9326. in conclusion, MADRL-Phish demonstrates superior performance by incorporating multi-dimensional feature extraction, while models focusing on single features struggle with detection accuracy and efficiency.

4.8.2. *Experiment 2: The Effectiveness Of DQN with additional features*

Experiment 2 investigates the impact of the feature extraction process on the MADRL-Phish model by using a Deep Q-Network (DQN). In this experiment, various models, including CNN-BILSTM, CNN-LSTM, CNN-RNN, LSTM, RNN, and CNN, are used in place of DQN in the MADRL-Phish model. The results of this comparison are summarized in Table 10.the findings show that the DQN network outperforms CNN-BILSTM in terms of classification detection, suggesting that incorporating additional feature extraction improves performance in multi-agent scenarios. Specifically, the DQN model achieves a False Positive Rate (FPR) of 0.0020, an F1 score of 0.9938, a True Positive Rate (Recall/TPR) of 0.9844, Precision of 0.9902, and an overall Accuracy of 0.9920.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 DQN. these results indicate that DQN 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.

4.8.3. 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 11, 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 0.0020, an *F1* score of 0.9938, a recall of 0.9844, precision of 0.9902, and accuracy of 0.9920. 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.

4.8.4. Experiment 4: Impact of Static Features on MADRL-Phish

Experiment 4 evaluates the impact of adding 16 static features to MADRL-Phish. Two variants are compared : Dynamic-only MADRL-Phish and Dynamic + Static MADRL-Phish. Results in Table 12 show that including static features improves performance across all metrics: *FPR* decreases from 0.0031 to 0.0020, *F1*-score rises from 0.9872 to 0.9938, Recall from 0.9765 to 0.9844, Precision from 0.9820 to 0.9902, and Accuracy from 0.9875 to 0.9920. These results indicate that static features enhance robustness and generalization in phishing detection.

4.8.5. Experiment 5: Evaluating The Effectiveness Of A Multiagent Approach

Experiment 5 evaluated the MADRL-Phish model's multi-agent (3 DQN) approach compared to a single-agent model. Results in Table 13 show that the multi-agent model achieved superior performance, with faster, more stable training and testing. The multi-agent model recorded an *FPR* of 0.0020, *F1* score of 0.9938, recall of 0.9844, precision of 0.9902, and overall accuracy of 0.9920, 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 MADRL-Phish model effectively represents webpages and improves classification by leveraging multi-agent deep learning and additional feature extraction, showcasing excellent prediction performance.

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 MADRL-Phish, 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 MADRL-Phish outperforms existing advanced phishing detection techniques, achieving an impressive accuracy of 99.20% and a false positive rate as low as 0.20%. 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.

REFERENCES

1. Opara, C., Chen, Y., and Wei, B., "Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics", *Expert Systems with Applications*, **236**,pp. 121183 (2024). https://doi.org/10.1016/j.eswa.2023.121183
2. Wang, M., Song, L., Li, L., et al., "Phishing webpage detection based on global and local visual similarity", *Expert Systems with Applications*, **252**,pp. 124120 (2024). https://doi.org/10.1016/j.eswa.2024.124120
3. Sánchez-Paniagua, M., Fernández, E.F., Alegre, E., et al., "Phishing URL detection: A real-case scenario through login URLs", *IEEE Access*, **10**,pp. 42949-42960 (2022). https://doi.org/10.1109/ACCESS.2022.3168681
4. Zhang, Z., Devaraj, M., Bai, X., et al., "Ethereum Phishing Scams Detection: A Survey", *2024 International Conference on Artificial Intelligence and Digital Technology (ICAIDT)*, pp. 70-75 (2024). https://doi.org/10.1109/ICAIDT62617.2024.00023
5. Roy, S.S., Awad, A.I., Amare, L.A., et al., "Multimodel Phishing URL Detection Using LSTM, Bidirectional LSTM, and GRU Models", *Future Internet*, **14**(11), pp. 340 (2022). https://doi.org/10.3390/fi14110340
6. Purwanto, R.W., Pal, A., Blair, A., et al., "PhishSim: Aiding Phishing Website Detection with a Feature-Free Tool", *IEEE Transactions on Information Forensics and Security*, **17**,pp. 1497-1512 (2022). https://doi.org/10.1109/TIFS.2022.3164212

7. Ozcan, A., Catal, C., Donmez, E., et al., "A hybrid DNN–LSTM model for detecting phishing URLs", *Neural Computing and Applications*, **35**,pp. 4957–4973 (2023). https://doi.org/10.1007/s00521-021-06401-z
8. Zhang, J., Sui, H., Sun, X., et al., "GrabPhisher: Phishing scams detection in Ethereum via temporally evolving GNNs", *IEEE Transactions on Services Computing*, **17**(6), pp. 3727-3741 (2024). https://doi.org/10.1109/TSC.2024.3411449
9. Hannousse, A. and Yahiouche, S., "Towards benchmark datasets for machine learning based website phishing detection: An experimental study", *Engineering Applications of Artificial Intelligence*, **104**,pp. 104347 (2021). https://doi.org/10.1016/j.engappai.2021.104347
10. Tang, L. and Mahmoud, Q.H., "A Deep Learning-Based Framework for Phishing Website Detection", *IEEE Access*, **10**,pp. 1509-1521 (2021). https://doi.org/10.1109/ACCESS.2021.3137636
11. Rao, R.S. and Pais, A.R., "Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach", *Journal of Ambient Intelligence and Humanized Computing*, **11**(9), pp. 3853-3872 (2020). https://doi.org/10.1007/s12652-019-01637-z
12. Feng, J., Zou,L., Ye,O., et al., Ou Ye , Jingzhou Han, "Web2Vec: Phishing Webpage Detection Method Based on Multidimensional Features Driven by Deep Learning", **8**,pp. 221214 - 221224 (2020). https://doi.org/10.1109/ACCESS.2020.3043188
13. Basit, A., Zafar, M., Liu, X., et al., "A comprehensive survey of AI-enabled phishing attacks detection techniques", *Telecommunication Systems*, **76**(1), pp. 139-154 (2021). https://doi.org/10.1007/s11235-020-00733-2
14. Schesny, M., Lutz, N., Jägle, T., et al., "Enhancing Website Fraud Detection: A ChatGPT-Based Approach to Phishing Detection", *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1494-1495 (2024). https://doi.org/10.1109/COMPSAC61105.2024.00205
15. Petrukha, I., Stulova, N., and Kryvoblotskyi, S., "Position Paper: Think Globally, React Locally—Bringing Real-Time Reference-Based Website Phishing Detection on macOS", *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 448-455 (2024). https://doi.org/10.1109/EuroSPW61312.2024.00057
16. Ashwatha, K. and Smilarubavathy, G., "Machine Learning Strategies to Detect Phishing Website", *2024 5th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pp. 347-351 (2024). https://doi.org/10.1109/ICICV62344.2024.00060
17. Korkmaz, M., Sahingoz, O.K., and Diri, B., "Feature selections for the classification of webpages to detect phishing attacks: a survey", *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1-9 (2020). https://doi.org/10.1109/HORA49412.2020.9152934
18. El Aassal, A., Baki, S., Das, A., et al., "An in-depth benchmarking and evaluation of phishing detection research for security needs", *IEEE Access*, **8**,pp. 22170-22192 (2020). https://doi.org/10.1109/ACCESS.2020.2969780
19. Merugula, S., Kumar, K.S., Muppidi, S., et al., "Stop Phishing: Master Anti-Phishing Techniques", *2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*, pp. 1-5 (2022). https://doi.org/10.1109/NKCon56289.2022.10126569
20. Sharma, B., Singh, P., and Kaur, J., "Cosine and Soft Cosine Similarity-Based Anti-Phishing Model", *New Approaches for Multidimensional Signal Processing:*

Proceedings of International Workshop, NAMSP 2020, pp. 165-174 (2021). https://doi.org/10.1007/978-981-33-4676-5_12

21. Liu, X. and Fu, J., "SPWalk: similar property oriented feature learning for phishing detection", *IEEE Access*, **8**,pp. 87031-87045 (2020). https://doi.org/10.1109/ACCESS.2020.2992381
22. Aleroud, A. and Zhou, L., "Phishing environments, techniques, and countermeasures: A survey", *Computers & Security*, **68**,pp. 160-196 (2017). https://doi.org/10.1016/j.cose.2017.04.006
23. Basit, A., Zafar, M., Liu, X., et al., "A comprehensive survey of AI-enabled phishing attacks detection techniques", *Telecommunication Systems*, **76**,pp. 139-154 (2021). https://doi.org/s11235-020-00733-2/10.1007
24. Elsadig, M., Ibrahim, A.O., Basheer, S., et al., "Intelligent Deep Machine Learning Cyber Phishing URL Detection Based on BERT Features Extraction", *Electronics*, **11**(22), pp. 3647 (2022). https://doi.org/10.3390/electronics11223647
25. Chatterjee, M. and Namin, A.-S., "Detecting phishing websites through deep reinforcement learning", *2019 IEEE 43rd annual computer software and applications conference (COMPSAC)*, pp. 227-232 (2019). https://doi.org/10.1109/COMPSAC.2019.10211
26. Sameen, M., Han, K., and Hwang, S.O., "PhishHaven—an efficient real-time ai phishing URLs detection system", *IEEE Access*, **8**,pp. 83425-83443 (2020). https://doi.org/10.1109/ACCESS.2020.2991403
27. Ali, W. and Malebary, S., "Particle swarm optimization-based feature weighting for improving intelligent phishing website detection", *IEEE Access*, **8**,pp. 116766-116780 (2020). https://doi.org/10.1109/ACCESS.2020.3003569
28. Guo, B., Zhang, Y., Xu, C., et al., "HinPhish: An effective phishing detection approach based on heterogeneous information networks", *Applied Sciences*, **11**(20), pp. 9733 (2021). https://doi.org/10.3390/app11209733
29. Alsariera, Y.A., Adeyemo, V.E., Balogun, A.O., et al., "Ai meta-learners and extra-trees algorithm for the detection of phishing websites", *IEEE Access*, **8**,pp. 142532-142542 (2020). https://doi.org/10.1109/ACCESS.2020.3013699
30. Ariyadasa, S., Fernando, S., and Fernando, S., "Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML", *IEEE Access*, **10**,pp. 82355-82375 (2022). https://doi.org/10.1109/ACCESS.2022.3196018
31. Ariyadasa, S., Fernando, S., and Fernando, S., "Detecting phishing attacks using a combined model of LSTM and CNN", *International Journal of Advanced And Applied Sciences*, **7**(7), pp. 56-67 (2020). https://doi.org/10.21833/ijaas.2020.07.007
32. Aljofey, A., Bello, S.A., Lu, J., et al., "BERT-PhishFinder: A Robust Model for Accurate Phishing URL Detection with Optimized DistilBERT", *IEEE Transactions on Dependable and Secure Computing*, **22**(4), pp. 4315-4329 (2025). https://doi.org/10.1109/TDSC.2025.3545771
33. Aldakheel, E.A., Zakariah, M., Gashgari, G.A., et al., "A deep learning-based innovative technique for phishing detection in modern security with uniform resource locators", *Sensors*, **23**(9), pp. 4403 (2023). https://doi.org/10.3390/s23094403
34. Meda, S., Srinivas, V.S., Rao, K.C.B., et al., "A dual-phase deep learning framework for advanced phishing detection using the novel OptSHQCNN approach", *PeerJ Computer Science*, **11**,pp. e3014 (2025). https://doi.org/10.7717/peerj-cs.3014
35. Li, Y., Huang, C., Deng, S., et al., "{KnowPhish}: Large language models meet multimodal knowledge graphs for enhancing {Reference-Based} phishing detection",

33rd USENIX Security Symposium (USENIX Security 24), pp. 793-810 (2024). https://doi.org/10.48550/arXiv.2403.02253

36. Cao, T., Huang, C., Li, Y., et al., "Phishagent: a robust multimodal agent for phishing webpage detection", *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 27869-27877 (2025). https://doi.org/10.1609/aaai.v39i27.35003
37. Gualberto, E.S., De Sousa, R.T., Vieira, T.P.D.B., et al., "The answer is in the text: multi-stage methods for phishing detection based on feature engineering", *IEEE Access*, **8**,pp. 223529-223547 (2020). https://doi.org/10.1109/ACCESS.2020.3043396
38. Louati, F. and Ktata, F.B., "A deep learning-based multi-agent system for intrusion detection", *SN Applied Sciences*, **2**(4), pp. 1-13 (2020). https://doi.org/10.1007/s42452-020-2414-z
39. Mohammad, R.M., Thabtah, F., and McCluskey, L., "An assessment of features related to phishing websites using an automated technique", *2012 international conference for internet technology and secured transactions*, pp. 492-497 (2012). ISBN:978-1-908320-08-7
40. Ozalp, R., Ucar, A., and Guzelis, C., "Advancements in deep reinforcement learning and inverse reinforcement learning for robotic manipulation: Toward trustworthy, interpretable, and explainable artificial intelligence", *IEEE Access*, **12**,pp. 51840-51858 (2024). https://doi.org/10.1109/ACCESS.2024.3385426
41. Nguyen, T.T., Nguyen, N.D., Vamplew, P., et al., "A multi-objective deep reinforcement learning framework", *Engineering Applications of Artificial Intelligence*, **96**,pp. 103915 (2020). https://doi.org/10.1016/j.engappai.2020.103915
42. Ali, H., Chen, D., Harrington, M., et al., "A survey on attacks and their countermeasures in deep learning: Applications in deep neural networks, federated, transfer, and deep reinforcement learning", *IEEE Access*, **11**,pp. 120095-120130 (2023). https://doi.org/10.1109/ACCESS.2023.3326410
43. Du, W. and Ding, S., "A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications", *Artificial Intelligence Review*, **54**(5), pp. 3215-3238 (2021). https://doi.org/10.1007/s10462-020-09938-y
44. Wang, W., Zhang, F., Luo, X., et al., "PDRCNN: Precise phishing detection with recurrent convolutional neural networks", *Security and Communication Networks*, **2019**(1), pp. 2595794 (2019). https://doi.org/10.1155/2019/2595794
45. *PhishTank > See All Suspected Phish Submissions. Accessed: Oct. 20, 2021. www.phishtank.com.[Online]. Available: https://www.phishtank.com/phish_archive.php*.
46. *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*.
47. Kumar., S., *Malicious and Benign URLs.kaggle.com*. 2019.
48. Gupta, N., Mujumdar, S., Patel, H., et al., "Data quality for machine learning tasks", *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 4040-4041 (2021). https://doi.org/10.1145/3447548.3470817
49. Zhang, J., Ou, Y., Li, D., et al., "A prior-based transfer learning method for the phishing detection", *Journal of Networks*, **7**(8), pp. 1201 (2012). https://doi.org/10.4304/jnw.7.8.1201-1207
50. Jiang, F., Ma, R., Gao, Y., et al., "A reinforcement learning-based computing offloading and resource allocation scheme in F-RAN", *EURASIP Journal on*

Advances in Signal Processing, **2021**,pp. 1-25 (2021). https://doi.org/10.1186/s13634-021-00802-x

51. Somesha, M., Pais, A.R., Rao, R.S., et al., "Efficient deep learning techniques for the detection of phishing websites", *Sādhanā*, **45**,pp. 1-18 (2020). https://doi.org/10.1007/s12046-020-01392-4

52. Adebowale, M.A., Lwin, K.T., and Hossain, M.A., "Intelligent phishing detection scheme using deep learning algorithms", *Journal of Enterprise Information Management*, **36**(3), pp. 747-766 (2020). https://doi.org/10.1108/JEIM-01-2020-0036

53. Le, H., Pham, Q., Sahoo, D., et al., "URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection", *arXiv e-prints*, pp. arXiv: 1802.03162 (2018). https://doi.org/10.48550/arXiv.1802.03162

54. Bahnsen, A.C., Bohorquez, E.C., Villegas, S., et al., "Classifying phishing URLs using recurrent neural networks", *2017 APWG symposium on electronic crime research (eCrime)*, pp. 1-8 (2017). http://dx.doi.org/10.1109/ECRIME.2017.7945048

Table 1: Comparative Analysis Of Phishing Detection Methods

Table2: URL Based Features

Table 3: Content HTML Based Features

Table 4: DOM Tree Based Features

Table 5. Evaluation indicator

Table 6. Data Source

Table 7. The performance of various dataset in detecting phishing webpages is evaluated.

Table 8.The MADRL-Phish model with different datasets.

Table 9. Comparison with classic phishing webpage detection methods

Table 10. The performance of various feature extraction models in detecting phishing webpages is evaluated.

Table 11. Detection effects of different feature combinations

Table 12. Comparison of MADRL-Phish performance with and without static features

Table 13. Detection effect of multiagent

Fig1.Model Framework

Fig2. Deep Q-Network.

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

48				
	Features / Methods	**Traditional Methods**	**Deep Learning Methods**	**Proposed MADRL-Phish**
1	Main Approaches	Blacklist-based, Heuristic Visual similarity[13],[19], ML-based [17], [22],[23],[24],[25].	Artificial feature engineering [30], Automatic feature learning[12] , Hybrid approaches[24].	Multi-Agent Deep Reinforcement Learning (MADRL)
2	Input Features	URL statistics, Whois/DNS info, Content-based features [17], [23],[24],[25].	URL, Content, DOM [30], Semantic features (Word2Vec [12], BERT [32])	Multi-dimensional: URL + Dynamic + Static + DOM + Content + Word2Vec
3	Advantages	Simple and fast, low resource consumption [18].	Automatic feature learning, High accuracy [30], [32], [33], [34], [35], [36],[37]	Automatic feature learning, Higher accuracy & F1, Zero-day detection, Robustness against adversarial attacks
4	Disadvantages	Cannot detect novel websites, High false positives[13],[19].	Requires large datasets, Computationally expensive [30], [31]	Higher complexity, Longer training time, More computational resources
5	Example Tools/Methods	HinPhish [28]	CNN, LSTM, GAN [33], [34], [35], [36],[37], BERT-PhishFinder [32]	MADRL-Phish (this work)
6	Emerging Trend	ML + Rule-based approaches [17], [20], [21].	Transformers, Hybrid CNN–LSTM [32], [33], [34], [35], [36],[37]	Ensemble of multi-agent RL with multi-feature integration

1

Table2			
URL Feature		**Explanation for 0**	**Explanation for 1**
1	IP	URL does not contain an IP address	URL contains IP address (e.g., matches regex for IPv4)
2	U_Length	URL length ≤ 75 characters	URL length > 75 characters
3	at_U	No '@' character in URL	URL contains '@' character
4	Dots_U	Number of dots in URL ≤ 3	Number of dots in URL > 3
5	TLD_P	No popular TLD (e.g., .com, .net) used in path	Path contains popular TLD (e.g., /com/, /net/)
6	https	URL uses HTTPS	URL does not use HTTPS
7	hyphen_U	No '-' character in domain	Domain contains '-' character
8	spe_chars_U	Special characters (e.g., %, $, &, =, etc.) ≤ 2 in URL	More than 2 special characters in URL
9	Sens_words	No phishing-sensitive words in URL	Contains words like "login", "verify", "secure", etc.
10	Length_S	Subdomain length ≥ 5 characters	Subdomain length < 5 characters
11	Brand_Name	No known brand name in domain	Known brand name (e.g., PayPal, Amazon) appears in domain
12	Det_Keywords	No phishing-related keywords in URL	Contains keywords such as "update", "confirm", "security"
13	Slash_U	Number of slashes "//" ≤ 5	Number of slashes > 5
14	Prefix	URL starts with protocol (e.g., http://, https://)	URL lacks standard protocol
15	Port	Uses standard ports (80, 443)	Uses non-standard ports (e.g., 8080, 21)
16	Punctuation	Punctuation count in URL ≤ 5	More than 5 punctuation marks in URL

2

Table 3			
	Content Feature	**Explanation for 0**	**Explanation for 1**

1	Anchor	External link ratio ≤ 30%	External link ratio > 30%
2	U_request	≤ 5 object requests from external domains	> 5 object requests from different domains
3	Popup	No popups requesting sensitive data	At least 1 popup requesting sensitive info
4	Links_in_tags	No blacklisted links in <meta>, <script>, <link>	At least one blacklisted link in these tags
5	Abn_req_URL	Request rate similarity with own domain ≥ 80%	Request rate < 80% matching, suggesting mimicry
6	Cookies	Cookies are present	No cookies used
7	Iframe	No iframe or iframe points to same domain	Iframe points to external domain
8	Submit	≤ 1 submit button	> 1 submit button
9	Form	≤ 2 HTML form elements	> 2 form elements
10	Favicon	Favicon loaded from same domain	Favicon loaded from another domain
11	Mailto	No <mailto:> links or limited use	Presence of <mailto:> sending sensitive data
12	IMG_Hyperlink	Image links point to same domain	Image links point to other domains
13	Susp_links	Visible text matches href domain	Mismatch between link text and destination domain
14	Right_click	Right-click enabled	Right-click disabled
15	Security	Uses secure forms (e.g., with action=https)	Insecure forms (e.g., http or empty action)
16	Action	Uses absolute form action URLs	Uses relative URLs for form actions

1

2

3

Table 4

	DOM Feature	**Explanation for 0**	**Explanation for 1**
1	Number of DOM Nodes	Total DOM nodes ≤ 1000	Total DOM nodes > 1000
2	DOM Tree Depth	Depth ≤ 20	Depth > 20
3	Tag Count	Normalized tag frequency within standard range	Outlier frequency of `<script>`, `<form>`, or `<input>` tags
4	CSS Class Count	≤ 20 different CSS classes	> 20 CSS classes
5	DOM Element Types	No uncommon tags (e.g., `<object>`, `<embed>`)	Presence of uncommon or potentially malicious elements
6	Text Length	Total visible text ≤ 2000 characters	Text > 2000 characters
7	Child Elements	≤ 50 child elements per parent node	> 50 child elements in any parent node
8	Hidden Form Elements	No hidden input fields	At least one hidden form element
9	Pop-up Windows	No JavaScript-based pop-ups	Use of `alert()`, `confirm()`, or modal dialogs
10	Iframe Elements	No iframe or only internal iframe	Presence of external iframe
11	External Resources	≤ 5 external JS/CSS/image resources	> 5 external resources
12	Script Tags	≤ 10 `<script>` tags without obfuscation	> 10 or obfuscated/minified scripts
13	Meta Refresh Tags	No meta-refresh tags	At least one meta-refresh tag
14	Interactive Elements	≤ 3 interactive elements (forms, buttons)	> 3 interactive elements
15	Mismatched Domains	Domain in HTML matches URL domain	Domain mismatch detected
16	Hover URLs	Link hover matches destination URL	Hover displays misleading or fake URL

4

1

2

3

Table 5	
Evaluation Indicator	**Calculation Formula**
Accuracy	$(TP + TN) / (TP + TN + FP + FN)$
Precision	$TP / (TP + FP)$
Recall (TPR)	$TP / (TP + FN)$
$F1$	$2 \times (Precision \times Recall) / (Precision + Recall)$
False Positive Rate (FPR)	$FP / (TN + FP)$

4

Table 6		
Data Source	**Legitimate URLs**	**Phishing URLs**
PhishStorm [44]	48,009	47,902
PhishTank [45]	0	178,495
ISCX-URL2016 [46]	35,378	9,965
Kaggle [47]	345,738	0

5

Table 7					
Dataset	**Accuracy**	**Precision**	**F1-score**	**TPR (Recall)**	**FPR**
PhishTank (21,303 phishing URLs)	0.9909	0.9890	0.9822	0.9757	0.0022
PhishStorm (95,911 URLs: 48,009 legitimate, 47,902 phishing)	0.9859	0.9826	0.9821	0.9813	0.0023
ISCX (45,343 URLs: 35,378 legitimate, 9,965 phishing)	0.9946	0.9867	0.9907	0.9826	0.0025
KPT-4 (40,000 URLs)	0.9819	0.9995	0.9803	0.9756	0.0024
KPT-6 (60,000 URLs)	0.9888	0.9972	0.9150	0.9623	0.0023
KPT-8 (80,000 URLs)	0.9908	0.9924	0.9616	0.9325	0.0022
KPT-10 (100,000 URLs)	0.9910	0.9922	0.9628	0.9351	0.0020
KPT-12 (120,000 URLs)	0.9917	0.9911	0.9751	0.9754	0.0021
KPT-14 (140,000 URLs)	0.9920	0.9915	0.9911	0.9828	0.0021

6

Table 8		
Model	**Dataset**	**Accuracy**
MADRL-Phish	Websites (PhishTank,Kaggle, Phish Storm, ISCX-URL-2016); 166103 Instance:81330 Phishing URLs, 84800 Legitimate URLs; Character-level Features based on URL ,Content,DOM with 48 supplementary features based on URL string domain, HTML and DOM.	**99.20%**
Reinforcement Learning[25]	Website (PhishTank,Yandex Search engine); 73,575 instance:37,175 phishing URLs,36,400 legitimate URLs; 14 features based on URL string domain,and HTML.	90.1%
RNN-GRU[10]	Websites (PhishTank,Kaggle); 120000 Instance:60000 Phishing URLs,60000 Legitimate URLs;	99.18%

	Character-level Features based on URL string	
Transfer Learning[49]	Website (Huawi Symantec); 177,417 intances :36,560 phishing URLs,14,0857 legitimate URLs; 15 features based on URLs,14,0857 legitimate URLs; 15 features based on URL string,domain, And sensitive words.	97%
Convolutional Autoencoder[50]	Websites (Phish Tank,Phish Storm, ISCX-URL-2016); 222,54 instances:127,628 legitimate URLs,94,913 phishing URLs; Character-Level features based on URL string.	97.82%
lSTM[51]	Websites (PhishTank,Alexa); 3526 instances:2119 phishing URLs,1407 legitimate URLs; 18 features are extracted from URL string ,third-party-based features.	99.57%
CNN+BILSTM[12]	Websites (PhishTank,Alexa); 46,103 instances: 21,303 phishing URLs, 24,800 legitimate URLs; Character-Level features based on URL string and features extracted from HTML	99.05%
CNN+LSTM[52]	Websites (PhishTank,Common Crawl,WHOIS); 1 million URLs,Over 10,000 images; Character-Level features based on URL string and features extracted from images.	93.28%

Table 9

Ref.	Model	Accuracy	Precision	TPR (Recall)	F1	FPR
This work	MADRL-PHISH	**0.9920**	0.9902	0.9844	0.9938	**0.0020**
[12]	WEB2VEC	0.9905	0.9869	0.9826	0.9908	0.0025
[10]	RNN-GRU	0.9918	0.9869	0.9859	0.9915	0.0104
[31]	Hybrid DLM	0.9695	0.9698	0.9628	0.9651	0.0150
[30]	PHISH_DET	0.9642	0.9640	0.9644	0.9642	0.0172
[53]	URLNET	0.9633	0.9620	0.8503	0.9027	0.0031
[54]	MPURNN	0.9326	0.9514	0.8980	0.9252	0.0374

Table 10

Algorithm	Accuracy	Precision	TPR (Recall)	F1	FPR
DQN	0.9920	0.9902	0.9844	0.9938	0.0020
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	0.9759	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 11					
Feature Combination	**Accuracy**	**Precision**	**TPR (Recall)**	**F1**	**FPR**
URL + HTML + DOM	0.9920	0.9902	0.9844	0.9938	0.0020
URL + HTML	0.9915	0.9919	0.9913	0.9914	0.0127
HTML + DOM	0.9870	0.9812	0.9834	0.9887	0.0080
URL + DOM	0.9790	0.9798	0.9758	0.9710	0.0026
HTML	0.9712	0.9767	0.9700	0.9514	0.0030
DOM	0.9372	0.9978	0.9733	0.9363	0.0221
URL	0.9010	0.8670	0.8800	0.9416	0.0259

Table 12					
Features Used	**Accuracy**	**Precision**	**TPR (Recall)**	**F1**	**FPR**
Dynamic Only	0.9875	0.9820	0.9765	0.9872	0.0031
Dynamic + 16 Static Features	0.9920	0.9902	0.9844	0.9938	0.0020

Table 13					
Agent Type	**Accuracy**	**Precision**	**TPR (Recall)**	**F1**	**FPR**
Multi-Agent	**0.9920**	0.9902	0.9844	0.9938	**0.0020**
Single-Agent	0.9110	0.8670	0.8811	0.8711	0.0030

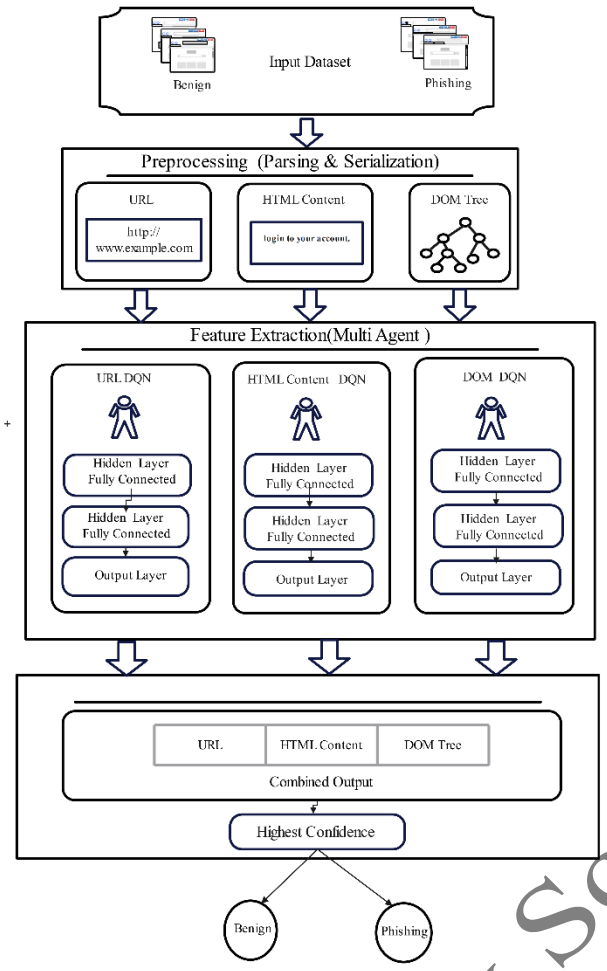

Fig1

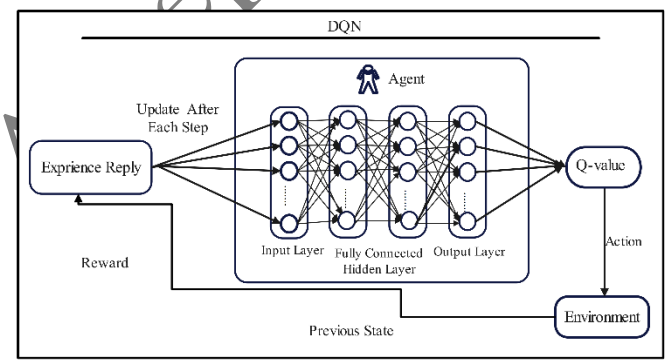

Fig2

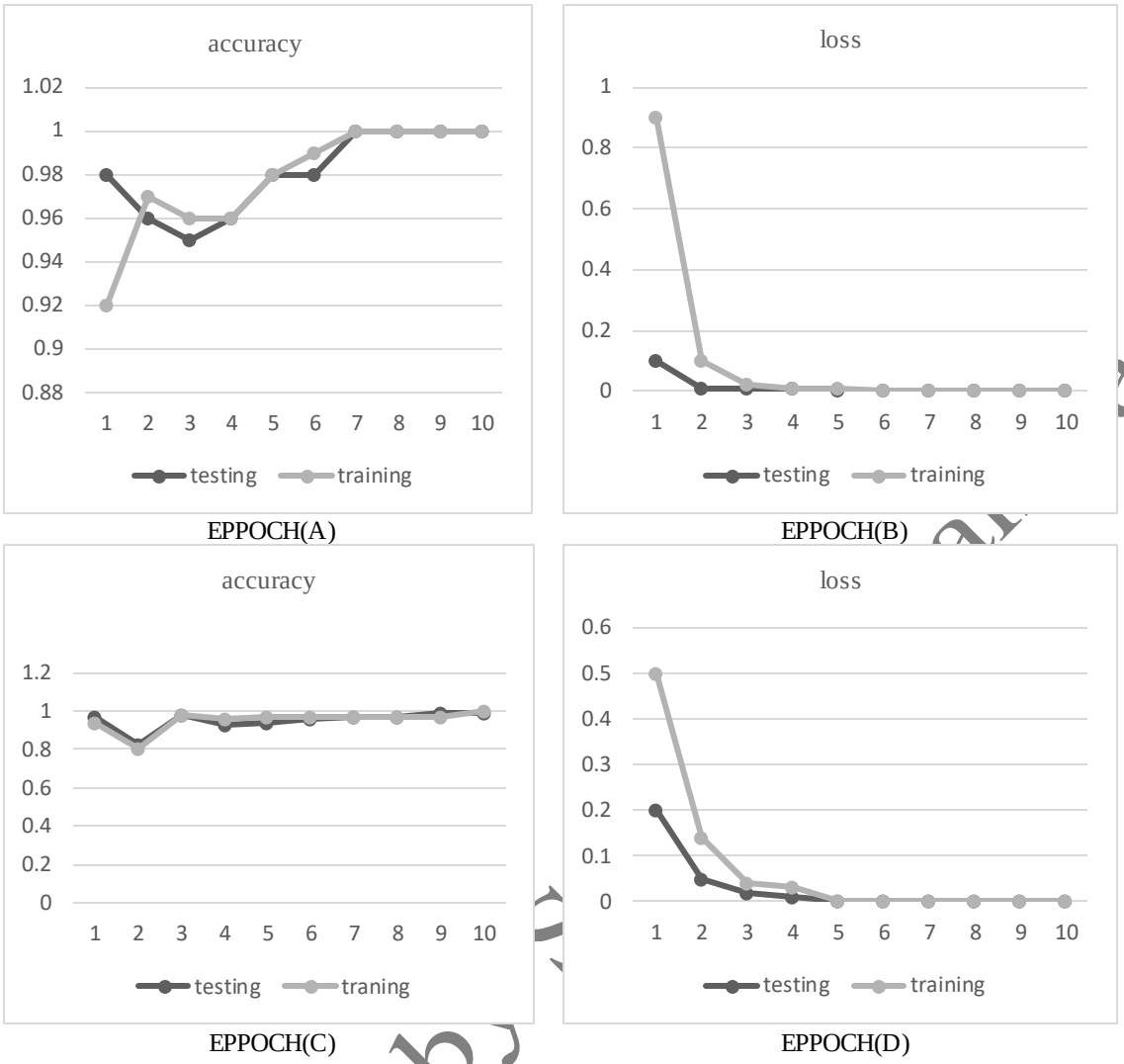

Fig 3.

Biographies

First Author

Ziba Jafari is a Ph.D. candidate in Computer Engineering at the Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran, where she is also involved in teaching undergraduate courses. Her research interests include deep learning, deep reinforcement learning, data mining, phishing detection, cybersecurity, and intelligent systems. Her current research focuses on multi-agent deep reinforcement learning models for web-based threat detection and intelligent decision-making systems. She has contributed to several research projects in artificial intelligence, cybersecurity, and data-driven analysis.

Second Author

Seyyed Hamid Ghafouri received his B.S. degree in Computer Engineering from Kharazmi University, Tehran, Iran in 2000, and his M.SC. degree in Computer Engineering from Najafabad Branch, Islamic Azad University, Isfahan, Iran in 2004, and his Ph.D degree Computer Engineering form Science and Research Branch Islamic Azad University, Tehran, Iran in 2021. He is currently an Associate Professor at the Computer Engineering Faculty of the Kerman Branch of Islamic Azad University and his research interests are Service Computing, Data Mining, Deep Learning, and IoT.

Third Author

Mohammad Ahmadinia is an Assistant Professor in the Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran. His research interests focus on cybersecurity, pattern recognition, machine learning, and data analytics. He has been actively involved in academic teaching and research activities and has published scientific papers in peer-reviewed journals and international conferences. His recent work includes phishing detection systems and intelligent web security frameworks.