Scientific Research Publishing

Journal of Information Security, 2026, 17(3), 221-242
https://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Deep Reinforcement Learning for Phishing Detection with Transformer-Based Semantic Features

Aseer Al Faisal, Atiqur Rahman

Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh
Email: aseer.faisal@northsouth.edu, atiqur.rahman@northsouth.edu

How to cite this paper: Al Faisal, A. and Rahman, A. (2026) Deep Reinforcement Learning for Phishing Detection with Transformer-Based Semantic Features. *Journal of Information Security*, **17**, 221-242.
https://doi.org/10.4236/jis.2026.173012

Received: March 11, 2026
Accepted: June 14, 2026
Published: June 17, 2026

Abstract

Phishing is a form of cybercrime in which people are deceived into exposing their personal information which can result in financial loss. These attacks are often executed via fraudulent messages, misleading advertisements and compromised legitimate websites. This study proposes a framework based on Quantile Regression Deep Q-Network (QR-DQN) that integrates RoBERTa semantic embeddings and crafted lexical features to enhance phishing detection. Instead of predicting mean returns, QR-DQN uses quantile regression to model the distribution over returns which improves stability and generalization for previously unseen phishing samples over traditional RL DQN approaches when combined with semantic embeddings. A custom crawled diverse dataset of 105,000 URLs were curated from PhishTank, OpenPhish, Cloudflare etc. The framework uses an 80/20 split of the dataset. The QR-DQN model with RoBERTa embeddings and lexical features achieved test accuracy 99.86%, precision 99.75%, recall 99.96% and F1-score 99.85% demonstrating high effectiveness. Compared to the standard DQN with lexical features, the suggested QR-DQN framework with lexical and semantic features lowers the generalization gap from 1.66 to 0.04 percent. The experiments using 5-fold cross-validation have resulted in consistent results under this protocol with a mean accuracy of 99.90% and standard deviation of 0.04%. This shows the hybrid technique which combines quantile-based value estimation with RoBERTa semantic embeddings and lexical features reports strong performance and reduced generalization gap.

Keywords

Index Words-Phishing Detection, Deep Reinforcement Learning, RoBERTa Semantic Embeddings, Quantile Regression Deep Q-Network, QR-DQN,

URL Classification, Lexical Features, Cybersecurity

1. Introduction

The modern internet faces persistent danger from evolving phishing threats, consistently identified as one of the most critical attack vectors. What used to be a straightforward trick has quickly transformed into a complex, multifaceted cybersecurity threat. Phishing is a common cyber threat that coerces users into revealing sensitive information, including credentials and financial details. The cybercrime information center nearly detected 1M phishing incidents between November 2023 and January 2024 [1]. Phishing is executed by cyber criminals who possess an understanding of human psychology, which is mostly influenced by greed, gullibility, and the desire for exploration. The manipulation technique that lures humans into disclosing their private and sensitive information is known as social engineering. People's tendency to make mistakes or to trust them easily makes them a prime target for cyber threats that result in breaches of the security system. It is one of the most common techniques to manipulate the user into revealing their confidential information. The application of various forms of machine learning algorithms to detect phishing classification problems and in particular to security and malware detection, has gained a lot of traction from the research community in recent years. These attacks intentionally trigger the victim psychologically and urge them to take immediate action, manipulating trust, exploiting the tendency to comply with authority. This approach is really effective with social engineering attacks, representing a significant portion of security breaches [2]. Even though one of the main underlying causes of Uniform Resource Locator (URL) is identifying theft and financial fraud through hijacking, ad injection attacks and URL spoofing [3].

The dynamic nature of these attacks, where the defense signatures that were effective yesterday become predictable vulnerabilities today, has revealed a significant weakness in traditional static cybersecurity approaches. Most standard detection algorithms depend on the extraction of superficial characteristics referred to as engineered lexical features. These characteristics encompass metrics like the length of the URL, the count of special characters and the existence of particular keywords. This approach makes features weak against small changes by attackers that lead to big problems in generalization. This manifests as a challenge and a significant difference is often seen in controlled training and its ability to sustain that accuracy against unseen data and a variety of attack patterns. In the field of machine learning this limitation is called the train-test generalization gap (Ggap), where a large gap signifies that the detection policy has memorized the specific pattern of known attack instances in the training data rather than learning the transferable concepts of phishing URLs. Deep reinforcement learning (DRL) has emerged as a paradigm for tackling cybersecurity challenges, especially via algorithms like DQN and DDQN, which allow the agent to learn adaptive policies [4]

[5]. Existing approaches show limitations in unpredictability and generalization. Unlike prior RL approaches relying exclusively on engineered lexical features and content-based representations, our work advances by introducing the Quantile Regression Deep Q-Network (QR-DQN) for distributional reinforcement learning. This hybrid approach, combined with distributional value learning, achieves a better generalization compared to standard DQN on identical features through uncertainty-aware, semantically-informed policy learning. This framework combines frozen RoBERTa transformer embeddings 768-dimensional semantic representations with lexical features, which allows the agent to capture richer semantic interpretation of malicious intent and stronger generalization beyond training distributions, such as domain impersonation, path obfuscation, and brand mimicry that transcend superficial pattern matching.

Previous studies investigating Deep Reinforcement Learning (DRL) for URL state representations have suffered from generalization limits and non-smoothness due to reliance on sub-optimal features [6]. By giving the reinforcement learning (RL) state more detailed contextual data representations, it learns to adapt deep generalization tactics to be able to defend against various sorts of phishing attacks. RoBERTa's bidirectional contextual encoding analyzes each URL string and produces 768-dimensional semantic embeddings with 50 engineered lexical features, building up a hybrid state representation that enables the QR-DQN agent to see deep contextual embeddings with statistical pattern recognition by which it learns to spot phishing data that were not seen in training. The architecture applies QRDQN featuring a Q-network structure of hidden units, soft target network decoupling, experience replay buffering and gradient clipping [7]. The integration of semantic URL representations supports stronger detection of novel and complex obfuscation strategies [8].

2. Related Works

As our technological devices and software systems have become increasingly advanced, the complexity of phishing is evolving at a fast pace. A variety of approaches are discussed below.

2.1. Traditional Machine Learning Approaches

A variety of machine learning techniques and algorithms have been applied in recent years that show promising results. Logistic regression, decision trees, support vector machines and neural networks have been used to solve this problem. The support vector machine (SVM) establishes a boundary that separates phishing and legitimate URLs. SVM reduces overfitting by maximizing the margin between classes and performs well on training data, but can struggle with newer types of threats. Random Forest (RF) builds many decision trees and combines their answers to make predictions. It is more resilient compared to many other ML techniques. If the dataset contains random noise, it can build trees that focus too much on this noise that result in lower accuracy in predictions. K-Nearest Neigh-

bour (KNN) has speed advantage in training but it can get quite slow as the dataset grows. KNN struggles with high dimensional spaces losing accuracy as feature count increases. It has poor performance when it comes to generalization as it primarily memorizes the training data. It is also highly sensitive to noisy data and suited well for small scale applications. Kumar *et al.* (2024) [9] investigate the nature of machine learning algorithms such as logistic regression and decision trees into phishing detection systems. These models used a wide range of features like lexical/textual, host based website characteristics and content based attributes to improve the classification of phishing. By training these models on labeled datasets they achieved better detection rates than traditional rule based models. Despite these improvements the machine learning models faced considerable limitations as they relied on static data and they had difficulty adapting quickly to new or unfamiliar phishing attempts. Those models lack the ability to learn from real-time phishing techniques.

2.2. BERT-Based Approaches

Recent developments in transformer-based deep learning models have transformed the way to detect malicious URLs. In particular, bidirectional encoder representations from transformers (BERT) feature self-attention methods to recognize semantic links among character and word-level tokens in URL strings [6]. Compared to conventional lexical or host-based feature extraction methods, BERT-based systems are able to attain a richer semantic understanding because of this bi-directional context modeling. The ability of BERT to recognize patterns of semantic maliciousness that dodge simple lexical heuristics gives it an advantage over traditional machine learning techniques. BERT learns contextual representations end-to-end from large URL datasets compared to SVM or Random Forest techniques that rely on manually created features. The caveat is that pure semantic models like BERT have difficulty capturing numeric or combined features such as URL length or the number of special characters. These features can be important indicators of phishing, since phishing URLs often show unusual lengths or symbol patterns [6] [10]. BERT works on relationships between individual tokens and does not naturally compute statistics over the whole sequence. Advanced transformers can be engineered to perform such functions but this is not their standard capability. These models excel at learning deep semantic context and meaning from textual representation but they still have limitations that can reduce detection performance when numeric, categorical and engineered features are critical for labeling phishing from safe URLs. Our approach uses a hybrid state representation that combines manually engineered structural and lexical features with transformer-based semantic embeddings with variation of BERT models. The reinforcement learning (RL) agent receives a feature vector that contains both the transformer based contextual information and explicit numeric, structural and statistical features such as URL length, special character counts and other known indicators of phishing. The agent can use both high-level semantic understanding and low-level

statistical cues to improve detection and generalization beyond what each feature type alone could achieve.

2.3. Reinforcement Learning Approaches

Several studies have used a range of reinforcement learning techniques to address the challenge of phishing detection. Early explorations in this direction used Deep Learning (DRL) where a self learning agent is trained to identify and model malicious URLs by learning both the value function and a classification policy. Chatterjee *et al.* [11] used an early deep reinforcement learning (RL) framework uses Deep-Q-Network (DQN) to model phishing URL detection as a sequential decision process which depends on 14 crafted lexical features such as URL length, IP presence, subdomain count etc. achieving 90.1% accuracy but relying solely on handcrafted URL-based attributes vulnerable to use evasion. but this approach is vulnerable to bypassing surface level attributes without altering core malicious objectives. The application of Deep Reinforcement Learning (DRL) for the purpose of intrusion detection is examined in [12]. The performance of DQN, DDQN, policy gradient and actor-critic against various machine learning (ML) algorithms on the NSL-KDD and AWID datasets. The output scores show that DDQN outperforms the other DRL based algorithms. And compared to methods like DDQN, distributional RL methods like QR-DQN offer better convergence properties and robustness in cybersecurity contexts specially in uncertain scenarios [13]. Through the process of trial, error and penalties the learning process of the reinforcement learning (RL) agent tries different actions and learns to continuously adapt detection processes by getting feedback from its results and is well suited for real-time environments where new threats are constantly evolving and learns to make better decisions. Our proposed reinforcement learning (RL) method can learn much more effectively from environment interactions when semantic embedding like BERT combined with lexical features for phishing detection. RL frameworks can learn nuanced decision policies that gain significant advantage as the integration provides both contextual depth and structural statistical information about the URLs. By integrating semantic embeddings with explicit lexical and numeric features, hybrid RL architectures empower agents to learn more nuanced and adaptive decision policies. This synergy leverages both high-level semantic insights and low-level statistical features.

3. Methodology

This paper introduces a framework that combines RoBERTa's semantic embeddings with lexical analysis of phishing detection URLs. This framework establishes superior adaptability, generalization and transfer learning capabilities across different attack vectors. This system addresses generalization and adaptability challenges in phishing detection as evolving attack techniques can reduce the effectiveness of existing detection systems on previously unseen phishing websites [14]. Conventional supervised methods treat phishing detection as astatic classification

problem where models are trained on labeled datasets and deployed with fixed parameters [15]. Attackers look for newer strategies to bypass typical detection systems. The reinforcement learning (RL) framework addresses this via trial, error and penalties and optimizes its decision making policy for phishing detection. Reinforcement learning (RL) allows updating of the policies on reinforcing. This framework is evaluated offline using a fixed dataset. An effective phishing detection system requires semantic embeddings with lexical and structural domain characteristics for deeper context of URLs. To effectively capture the semantic context RoBERTa—a Bidirectional Encoder Representations from Transformers [16] a pre-trained language model is used. It generates deep bidirectional contextual relationships with textual data by considering both left and right contexts across all of its layers [17]. In recent studies it is being shown that BERT based similar transformer models demonstrate significant improvement in detecting adversarial manipulations in URLs [18]. BERT based transformer models show very high significance in detecting adversarial manipulations in URLs such as random character insertions, homoglyph attacks or deceptive use of sub-domains, due to the deep contextual embeddings which goes way beyond shallow surface level features. As a result, BERT based models are increasingly preferred for cybersecurity applications. This framework uses 768-dimensional semantic embeddings mixed with manually crafted lexical features and input into a Quantile Regression Deep-Q-Network (QR-DQN) agent. This hybrid representation in reinforcement learning (RL) agent learns a robust policy that can adapt to changing phishing strategies through RL environment interaction and reward driven optimization.

3.1. Data Collection and Initial Processing

The dataset is made by collecting and extracting URLs from multiple sources to ensure a comprehensive representation of both malicious and legitimate web content. URLs were crawled from various sources like PhishTank, OpenPhish etc. ensuring a comprehensive representation of both malicious and legitimate URLs. These sources were chosen for their community validation processes and recognized credibility within the research community. To collect legitimate URLs, Cloudflare's domain ranking system was used for top-ranked domains filtered by high traffic volume. All in all these form a reliable real world dataset with 105k URLs. During crawling each URL feature was extracted. The script prints detailed progress that shows the current URL that is being processed. A set of manipulated urls was included in the dataset to test how well the detection model handles unusual inputs. The manipulated character level modification mimics common patterns used in phishing attacks to assess models' resilience against evolving tactics. The extraction pipeline was created along with managing network level failures effectively. Type conversion and null filling are performed for columns that are expected to be booleans. Some lexical URL features are derived from URL structure and character composition, including length-based and special-character-based properties [19].

Additional features may include obfuscation ratios, character continuation rates, percentage encoded segments, non-alphanumeric characters, and log probability of URL character sequences.

$$\text{CharLogProb}(\text{URL}) = \sum_{i=1}^{L} \log p(c_i) \quad (1)$$

Adjustable delays are added for requesting server resources for tackling networking failures and saving extracted features in a csv file. The resulting feature set includes 50 columns covering URL-based, HTML-based and derived phishing indicators, consistent with feature-engineering approaches used for phishing URL detection [20].

Category	Features (50 Total)
URL-Based	URLLength, DomainLength, IsDomainIP, URLSimilarityIndex, CharContinuationRate, TLDLegitimateProb, URLCharProb, TLDLength, NoOfSubDomain, HasObfuscation, NoOfObfuscatedChar, ObfuscationRatio, NoOfLettersInURL, LetterRatioInURL, NoOfDigitsInURL, DegitRatioInURL, NoOfEqualsInURL, NoOfQMarkInURL, NoOfAmpersandInURL, NoOfOtherSpecialCharsInURL, SpacialCharRatioInURL, IsHTTPS
HTML Structure	LineOfCode, LargestLineLength, HasTitle, DomainTitleMatchScore, URLTitleMatchScore, HasFavicon, Robots, IsResponsive, HasDescription
Redirect & Popup	NoOfURLRedirect, NoOfSelfRedirect, NoOfPopup, NoOfiFrame
Form & Link Analysis	HasExternalFormSubmit, HasSocialNet, HasSubmitButton, HasHiddenFields, HasPasswordField, NoOfSelfRef, NoOfEmptyRef, NoOfExternalRef
Indicators & Content	Bank, Pay, Crypto, HasCopyrightInfo, NoOfImage, NoOfCSS, NoOfJS

The final training dataset consists of 100,000 URLs, evenly balanced between 50,000 phishing and 50,000 legitimate samples (labels: 1 = phishing, 0 = legitimate). It was constructed by merging a large aggregated corpus of 95,524 URLs with an additional phishing feed of 4476 URLs. Within the aggregated corpus, 45,524 samples are phishing and 50,000 are legitimate, including 2000 legitimate URLs sourced from the Cloudflare Top Sites list and randomly selected URLs from the PhiUSIIL dataset, together with curated benign web sources. The merged data were processed using exact URL-string matching for deduplication, followed by handling of missing or malformed values and min-max normalization; no further filtering or sample exclusion was applied.

To avoid the data leakage, both the 80/20 train-test split as well as the 5-folds cross-validation were performed at group, not individual URLs. Before partitioning, each sample assigned to its root domain was given a canonical group. So, all related URLs and their manipulated variants were regarded as one unit. These

links comprise copies of the same pages or almost identical pages like mirror pages and those that employ adversarial variants. Adversarial variants are character substitutions, token insertions or padding, subdomain rearrangements, forms like homoglyphs and encoded forms. The next partition was done with the stratification by groups. This means that any samples that belong to the same canonical group were assigned in their entirety to either a split or a fold. Thus, any variant stemming from the same base URL or domain could not show up in the training, validation or test partitions. This process deals directly with the risk of leakage from near duplicate or processed samples, thereby assuring the results reported generalize to unseen URL families rather than memorization of related variants.

A subset of engineered features relies on successful HTTP/HTML retrieval of the target website. These are all content- and response-based attributes such as page title features (HasTitle, DomainTitleMatchScore, URLTitleMatchScore), description and robot indicators (HasDescription, Robots), structural elements (NoOfImage, NoOfCSS, NoOfJS, NoOfiFrame, NoOfPopup), form and interaction features (HasExternalFormSubmit, HasSubmitButton, HasHiddenFields, HasPasswordField), hyperlink-based statistics (NoOfSelfRef, NoOfEmptyRef, NoOfExternalRef), and redirect and responsiveness indicators (NoOfURLRedirect, NoOfSelfRedirect, IsResponsive, HasFavicon, HasSocialNet). All other features are directly derived from the URL string with no need for a page fetch. While collecting data, in some instances a very small number of samples could not be retrieved because of network errors, empty HTML responses, and inactive pages. These cases were not discarded; instead, all HTML-dependent features for such samples were deterministically set to 0 to represent the missing content. A zero-imputation strategy was applied to 497 samples (0.50% of the dataset) for which HTML retrieval failed. As part of the training process, features were normalized after zero-imputation, whereby failed retrievals were represented through zero-imputed HTML-dependent features and treated as valid but uninformative feature values. During inference, the same fallback mechanism is applied. If live HTML retrieval fails, prediction proceeds using URL-derived features together with zero-imputed content features to ensure robustness.

3.2. Contextual Feature Engineering and Hybrid State Space

The framework provides input for the QR-DQN agent by merging pre-calculated semantic embeddings with lexically created features. This state vector is designed to capture the syntactic irregularities commonly seen in basic phishing along with the semantic cues necessary to detect more advanced hidden threats. We created a custom dataset using feature engineering processes. The processed datasets are for URL and content analysis modes, utilizing structured data that includes domain-specific attributes. The main goal of this implementation is to extract structural features by analyzing the URL string through systematic parsing. A short subdomain and a less common top-level domain indication are treated as predictive of low trustworthiness. These unusual structures are reinforced with lexical

analyses that assess the ratios of digits to alphanumeric characters. The framework also checks for suspicious patterns such as IP addresses in domain names or homograph attacks [21].

In addition, probabilistic metrics are used to enhance detection. For example, the framework tracks the ratio of repeating characters, which can indicate redundant character repetition used to obfuscate text. A model that estimates the probability distribution of characters assigns a log-probability score to each URL string. The training set for this model consists only of negative items (benign URLs). This probability method identifies sequences that are statistically improbable, which is beneficial for detecting extreme anomalies and generated phishing URLs. Each URL also undergoes a semantic encoding process. The URL is tokenized using a RoBERTa Byte-Pair Encoding (BPE) model with a sequence length of 256 tokens. The output is truncated and transformed into tensor format. If a GPU is available, the model operates on the GPU; otherwise, it runs on a CPU. This design reduces computational demand while preserving the strong semantic capability of the pre-trained RoBERTa model.

3.3. Phishing Environment Design

The task of phishing detection is structured as a single-step Markov Decision Process (MDP) within a specialized PhishEnv environment developed on the Gymnasium framework [22]. At the core of this environment lies a hybrid state representation that integrates two complementary categories of features. The first category consists of a collection of meticulously crafted features, normalized to a range of [0, 1] to facilitate stable learning [23]. The second category is a contextual semantic embedding produced by a pre-trained RoBERTa model. For any given input text, such as a URL or content, the [CLS] token embedding from the final layer of RoBERTa is extracted, yielding a dense 768-dimensional vector that encapsulates intricate linguistic and structural patterns. These numerical and semantic vectors are then concatenated to create a comprehensive state vector, which defines a high-dimensional, continuous observation space.

Even though phishing detection is a single-step binary decision problem, it is presented as single-step MDP to directly optimize decisions involving asymmetric misclassification costs. The setup includes the cost in the reward function so that the model learns a policy that prevents the high-cost error (e.g., a false negative). Unlike cost-sensitive supervised classifiers which indirectly incorporate such costs through assignment of class weights or change of the decision threshold, the cost is part of the optimization objective itself. Using this formulation, QR-DQN trains a distribution of returns instead of the expected return. Using a quantile-based approach, it accounts for variation and all URL uncertainties. In phishing detection, rare but false negatives that can have a high cost must be penalized more heavily. By modeling the entire return distribution, QR-DQN allows for risk-sensitive decision-making that takes into account the worst-case scenario as well as the mean. This is not naturally captured by standard classifiers without additional

risk modeling or mean-value estimation.

The agent interacts with this state through a discrete action space containing two options corresponding to legitimate (0) or phishing (1). One key improvement of this environment is the deliberately asymmetrical reward function, which simulates the severe real-world consequences of missed phishing attacks. The agent receives a positive reward (+1) for a correct prediction. On the other hand, the penalties for mistakes are carefully calibrated. A false negative, where a phishing site is missed, receives a heavier penalty (−2) to discourage this high-risk outcome. Meanwhile, a false positive, where a legitimate site is classified as phishing, receives a lower penalty (−0.5). This reward framework instructs the learning algorithm to prioritize high recall for the phishing category, which aligns with the security objective of assigning greater cost to missing a threat.

In order to achieve class-balanced sampling, the environment's reset function samples classes so that no imbalance is introduced. When the environment resets, it randomly chooses an instance from either the legitimate category or the phishing category with equal probability (50/50). In addition, for performance and reproducibility, all RoBERTa embeddings are pre-computed at initialization and cached to disk using a hash of the dataset. This prevents repeated inference in future runs and speeds up training cycles significantly. The combined design creates a secure, efficient, and stable training environment that helps the DQN agent evolve into a precise and strong phishing detector [24]. The reward function is expressed as follows:

$$R(s_t, a_t) = \begin{cases} +1.0, & \text{correct classification of legitimate or phishing,} \\ -2.0, & \text{predicted legitimate when the true label is phishing (false negative),} \\ -0.5, & \text{predicted phishing when the true label is legitimate (false positive).} \end{cases} \quad (2)$$

3.4. Evaluation Setup

To fully assess how strong the model is against evasion attacks, the dataset was improved with modified phishing URLs. These examples were created to imitate real obfuscation techniques that criminals could use to avoid detection. The alterations included character-level changes such as google to g00gle, token insertion and padding, shuffled domain and subdomain structures such as secure.paypal.com.login.cn, and encoding-based manipulation such as Unicode variants. Through such variants, a pre-existing functional URL can remain valid even though its lexical structure is altered. We incorporated these variations into both training and testing splits to study the generalization capability of the model under obfuscation scenarios. Many phishing datasets include intentional lexical deception to evade existing rule-based and machine learning detectors [25]. URLs often imitate formats used by real-world entities or exploit similarities at the domain level to mislead users and classifiers. Including these variants in model evaluation provides an authentic stress test, ensuring that the model can generalize beyond clean datasets.

Category	Example/Pattern	Description
Homoglyph/Character Substitution	g00gle.com, paypa1.net	Characters visually similar to legitimate domains
Punycode/IDN Homograph	xn-example-9db.com	Unicode-based domain encoding to mimic trusted sites
Encoded URLs	http://example.com/%32%31	Use of percent-encoded characters for obfuscation
Look-alike TLDs	.co, .orq, .net	TLDs crafted to resemble popular legitimate domains
IP/Randomized Domains	http://192.168.0.1/login	Numeric or non-semantic domain structures

3.5. QR-DQN Agent Architecture

The phishing detection policy is developed using a Quantile Regression DQN (QR-DQN) agent that models a distribution of returns per action via multiple quantiles and the quantile Huber loss, rather than a single scalar Q-value, enabling uncertainty-aware value estimation. The primary policy network is an MLP with layers [512 → 512 → 256 → 128] using ReLU activations, tailored for a high-dimensional hybrid state that concatenates normalized numerical features (URL structural and domain-specific metrics) with 768-dimensional BERT embeddings capturing deep semantic context [26]. The online network outputs a tensor of shape [num_actions × num_quantiles], reshaped to [num_actions, num_quantiles], and a distinct target network soft-updated via Polyak averaging ($\tau = 0.005$) at a fixed interval of every 1000 steps provides stable quantile targets for distributional Bellman updates. This distributional design improves calibration of value estimates and enhances policy generalization under stochastic rewards.

The learning process uses an experience replay buffer of 150,000 transitions and begins updates after an initial 5000-step data collection phase. Training uses batches of 512 with 8 gradient updates after every 4 environment interactions, a quantile Huber loss for robust distributional regression, and gradient clipping with max norm 10 for stability. Exploration follows ε-greedy starting at $\varepsilon = 1.0$, linearly decaying to $\varepsilon = 0.02$ over the first 25% of a 300,000-timestep training budget; action selection uses the expected return computed as the mean over quantiles for each action. A high discount factor ($\gamma = 0.995$) emphasizes long-term rewards, while the target network supplies quantile targets to prevent destabilizing feedback during learning, with updates applied at a fixed interval of every 1000 steps.

QR-DQN Training Hyperparameters:

Hyperparameter	Value	Description
Architecture	512 → 512 → 256 → 128	Multi-layer perceptron hidden units for hybrid state fusion.
Replay Buffer Size	150,000	Experience transitions stored for decorrelated sampling.
Batch Size	512	Training batch size per update.
Gradient Updates/Step	8 per 4 steps	Update frequency per environment interaction.
Loss Function	Quantile Huber loss	Distributional regression for QR-DQN targets (replaces Smooth L1).
Gradient Clipping	Norm ≤ 10	Maximum allowed gradient norm for stability.

Continued

Exploration Schedule	ε: 1 → 0.02	Linear decay over the first 25% of 300,000 steps.
Training Steps	300,000	Total interaction steps.
Polyak Averaging Coefficient	0.005	Target-network smoothing parameter τ.
Target Network Interval	1000 steps	Frequency of target network updates.
Discount Factor (γ)	0.995	Future reward weighting.
Initial Experience Steps	5000	Steps collected before learning starts.

RoBERTa Parameters:

Parameter	Value	Description
Model	roberta-base	Hugging Face pretrained BERT backbone.
Token Extraction	<s> token (first)	Source token for dense representation in RoBERTa (analogous to CLS).
Embedding Dimension	768	Width of contextual feature vector for roberta-base.
Max Sequence Length	256	Tokens per URL; padded or truncated.
Tokenizer	Byte-Pair Encoding	RoBERTa's BPE tokenizer for URL segmentation.
Fine-Tuning	None	Pretrained weights used; not domain-adapted.
Inference Hardware	GPU/CPU	Run on GPU if available, otherwise CPU.
Empty/Invalid String	Zero vector	Encoding for missing strings.

3.6. Agent Environment Interaction

In QR-DQN (Quantile Regression Deep Q-Network), the Q-function is represented by a set of quantiles that estimate the distribution of possible returns instead of a single expected value as in traditional Q-learning. Each state-action pair has its own quantile values, which reflect the full distribution of returns. The Q-function representation and expected return are given by [27]:

$$Q(s,a)=\mathbb{E}\left[R_t \mid s_t=s, a_t=a, \pi\right] \tag{3}$$

For QR-DQN, the return distribution is represented using N quantile estimates, each corresponding to a quantile level [28]:

$$Z(s,a)=\left[q_1(s,a), q_2(s,a), \cdots, q_N(s,a)\right] \tag{4}$$

The Bellman target for each quantile is as follows [29]:

$$z_j = r+\gamma z_j^{-}\left(s', a^*\right) \tag{5}$$

where:

- r is the observed reward,
- γ is the discount factor,
- $z_j^{-}\left(s', a^*\right)$ is the j-th quantile predicted by the target network for the next state s' and action a^*,
- $a^* = \arg\max_{a'} \frac{1}{N}\sum_{k=1}^{N} z_k\left(s', a'\right)$.

The binary action space has two choices: either labeling the sample as phishing or legitimate. After each action, the environment provides a new observation, reward, and episode-termination flag. During training, quantile-based Bellman targets guide the agent's updates.

Reinforcement learning (RL) agents use their experiences in the environment to gradually modify their value estimates according to the rewards returned by the environment. In our study, however, the agent learns offline on a static dataset. The value estimates are updated from sampled transitions using quantile-based targets. The core temporal-difference update used in QR-DQN can be expressed as follows [28]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} \frac{1}{N} \sum_{k=1}^{N} z_k (s',a') - Q(s,a) \right] \tag{6}$$

The parameter updates take place only during offline training on a fixed dataset. The deployed model is not updated online. Therefore, the reported results are a product of offline learning rather than continual online adaptation to newly emerging threats. The overall architecture of the proposed phishing detection framework in **Figure 1**.

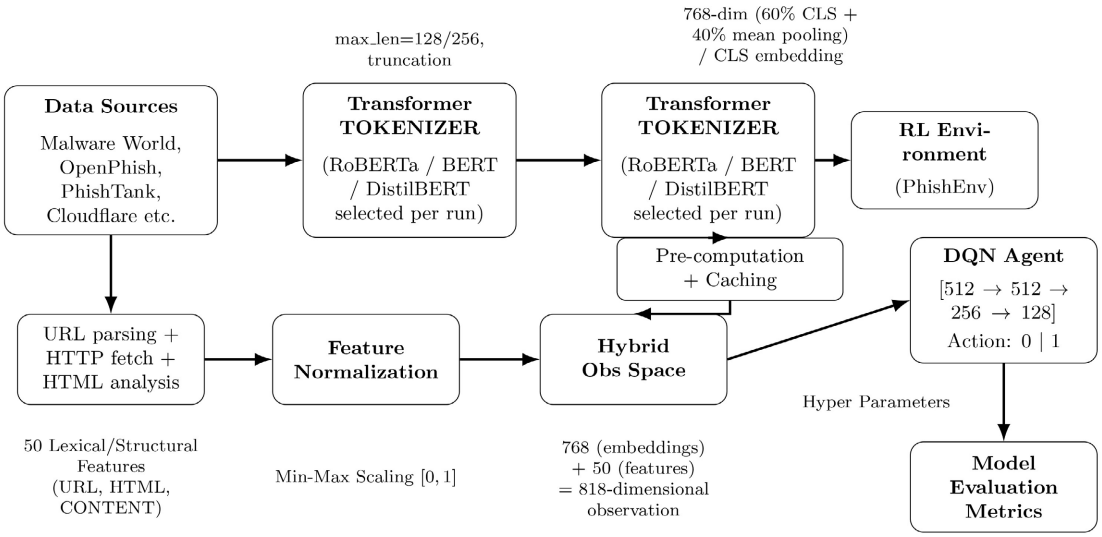

Figure 1. Architecture of the phishing detection system.

3.7. Evaluation Metrics

To evaluate the performance of the proposed phishing detection framework, several standard metrics were used to assess both predictive accuracy and generalization capability on unseen data. The framework uses predictions categorized as follows:

- **True Positives (TP):** Phishing URLs correctly classified as phishing.
- **True Negatives (TN):** Legitimate URLs correctly classified as legitimate.
- **False Positives (FP):** Legitimate URLs incorrectly classified as phishing.
- **False Negatives (FN):** Phishing URLs incorrectly classified as legitimate (missed attacks).

Based on these fundamental counts, the following metrics are calculated.

Accuracy measures the overall proportion of correct predictions across both classes.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

Balanced Accuracy accounts for class imbalance by computing the arithmetic mean of recall and specificity.

$$\text{Balanced?Accuracy} = \frac{\text{Recall} + \text{Specificity}}{2} \tag{8}$$

Precision measures the ratio of correct positive predictions, revealing the model's dependability when it flags a URL as phishing.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

Recall measures the model's capacity to identify attacks by quantifying the fraction of real phishing URLs correctly detected.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

F1-score is the harmonic mean of precision and recall. It combines the model's ability to identify positive cases while minimizing both false positives and false negatives.

$$F1 = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{11}$$

Specificity measures the proportion of legitimate URLs correctly identified as legitimate.

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{12}$$

False Negative Rate (FNR) quantifies the proportion of phishing attacks that evade detection.

$$\text{FNR} = \frac{FN}{FN + TP} = 1 - \text{Recall} \tag{13}$$

False Positive Rate (FPR) measures the proportion of legitimate URLs incorrectly flagged as phishing, which affects system usability.

$$\text{FPR} = \frac{FP}{FP + TN} = 1 - \text{Specificity} \tag{14}$$

The cost of different types of errors is asymmetric. Missing a phishing attack (false negative) has far more severe consequences than incorrectly flagging a legitimate site (false positive). The asymmetric reward function discussed in Section 3.3 reflects this imbalance by applying a penalty of −2.0 for false negatives and −0.5 for false positives, indicating that missing an attack is more costly than triggering a false alarm. Machine-learning-based security systems also face the challenge of generalization, since overfitting occurs when a model memorizes static patterns rather than learning transferable behavior for unseen threats.

The accuracy gap measures how much performance varies on test data compared with training data.

$$G_{\text{gap}} = \text{Accuracy}_{\text{train}} - \text{Accuracy}_{\text{test}} \tag{15}$$

The F1 gap similarly measures generalization capability using the F1-score.

$$F1_{\text{gap}} = F1_{\text{train}} - F1_{\text{test}} \tag{16}$$

Lower F1-gap values indicate that the model preserves its F1-score better on test data. Models with smaller generalization gaps remain effective even when phishing tactics differ from the training examples. This indicates less overfitting and stronger robustness to newly emerging attacks, making the model more reliable over time.

- **Recall prioritization:** This prioritizes maximizing recall, emphasizing the proportion of true phishing URLs that are correctly identified.
- **Balanced evaluation:** Metrics such as F1-score consider both precision and recall jointly, providing a fair assessment by accounting for false positives and false negatives together. This is particularly important for imbalanced datasets.
- **Generalization emphasis:** Static pattern memorization does not hold up against evolving attacks. The generalization-gap metrics directly evaluate a model's ability to adapt to new attack variations, which is a main benefit of the proposed BERT-enhanced method.
- **Operational transparency:** By reporting the full confusion matrix (TP, TN, FP, FN), security analysts can determine whether the system's error profile aligns with their organization's risk tolerance and operational constraints.

4. Results

4.1. Experimental Setup and Configuration

The balanced phishing-URL dataset was used after performing stratified train–test splits. We combined engineered numerical features with pre-computed RoBERTa embeddings and characterized the detection problem as a one-step Markov decision process (MDP), where each episode evaluates a single sample and the action space is binary. A deep multilayer perceptron (MLP)-based QR-DQN agent was trained with hyperparameters selected according to the dataset scale. The reward function emphasized reducing missed phishing attempts by assigning a penalty of −2.0 to false negatives. It also applied a penalty of −0.5 to false positives while awarding +1.0 for correct predictions. Metric comparisons were conducted on the held-out test set using accuracy, F1-score, recall, and generalization-gap measures.

4.2. Comparative Performance Analysis

The QR-DQN agent enhanced with RoBERTa obtained test accuracy of 99.86%, precision 99.75%, recall 99.96%, F1 score 99.85%, with only 4 false negatives and 25 false positives out of 20,000 test samples. When we compare, the baseline agent with lexical features only achieves 98.30% test accuracy, 99.82% precision, 96.76%

recall, and 98.27% F1-score with 323 false negatives and 17 false positives. The RoBERTa-based model's accuracy score increased by 1.56% while the number of false negatives decreased by 1.56%.

An ablation study was carried out to isolate the contribution of distributional reinforcement learning by running QR-DQN (dotted) and standard DQN (dot-dashed) in the same lexicon-only feature setting. The DQN baseline achieved 98.30% accuracy, 96.75% recall, and 98.27% F1-score while QR-DQN achieved a test accuracy of 98.12%, a recall of 96.39%, and an F1-score of 98.08% in this setting. The QR-DQN produced a larger number of false negatives (359 versus 323) and a slightly larger generalization gap (1.77% versus 1.63%).

The findings imply that distributional RL does not provide a significant advantage over standard DQN in this framework, nor does it further enhance generalization in a single-step deterministic phishing detection setting that only contains lexical features.

However, when semantic embeddings from RoBERTa are added, QR-DQN consistently outperforms DQN with respect to recall, reduced false negatives, and a significantly smaller generalization gap. The advantage of using QR-DQN appears in rich state representation settings when modeling the return distribution makes uncertain and risk-sensitive decision making more effective.

4.3. Generalization Capability

To quantify the generalization of the models, the gap in accuracy was measured based on train-test data. The QR-DQN agent using RoBERTa semantic embeddings and lexical features has a very small generalization gap of 0.042 percent. The total loss of baseline DQN agent using only lexical features showed a significantly larger gap of 1.63 percent. As measured by the generalization gap, this shows a 39-fold reduction, which indicates that the RoBERTa-enhanced agent learns generalizable concepts of malicious URLs and not training data patterns.

The QR-DQN model, which relies solely on lexical features, shows an even larger generalization gap (Accuracy Gap: 1.77%, F1 Gap: 1.81) than the lexical DQN baseline. It is shown that distributional RL does not improve generalization by itself and that semantic embeddings are required for this.

4.4. Semantic Feature Integration Benefits

Using BERT-style embeddings substantially improved contextual understanding and robustness. Because the transformer encoder attends to both left and right context, the model can better capture brand-name manipulation, typosquatting, and semantic obfuscation. The semantic features proved more resistant to common evasion strategies, including character-level obfuscation, homoglyph attacks, insertion of special characters, subdomain manipulation, and URL shortening.

4.5. Training Stability and Convergence

The RoBERTa-enhanced agent also demonstrated improved training stability and

better convergence properties. Policy optimization became more stable, and state exploration was guided more effectively by semantic information, resulting in reduced Q-value fluctuations. The asymmetric reward structure imposed a severe penalty on false negatives, which carry greater security risk. This helped the training process prioritize attack detection while still managing false positives at a level compatible with practical usability.

4.6. Computational Efficiency Considerations

The computational overhead of extracting RoBERTa embeddings was reduced through pre-computation and disk caching, along with efficient batching and parallelized processing. These design choices made training substantially more practical. Successful model training within 300,000 timesteps suggests that the framework is feasible for real-world deployment settings.

4.7. Experimental Reliability and Evaluation Consistency

The model achieved high accuracy, precision, recall, and F1-score on the held-out test set. With mean accuracy of 99.9% and standard deviation of 0.04%, performance under 5-fold cross validation was found to be stable. The assessment emphasizes descriptive performance in this experimental context and does not extend to multi-seed analysis, confidence interval estimates, or formal statistical significance tests. A static dataset was used for all experiments executed offline. The QR-DQN policy was trained in this offline regime and used during inference with no online updates. Thus, the results show offline phishing detection performance under the specified train/test split and cross-validation protocol. The association of QR-DQN with the semantic embeddings of RoBERTa, along with lexical features explains the observed improvement in results.

4.8. Results Tables

For completeness, Figure 2 together with Tables 1-3 summarize the train-split performance, test-split performance, and overfitting behavior of the evaluated models.

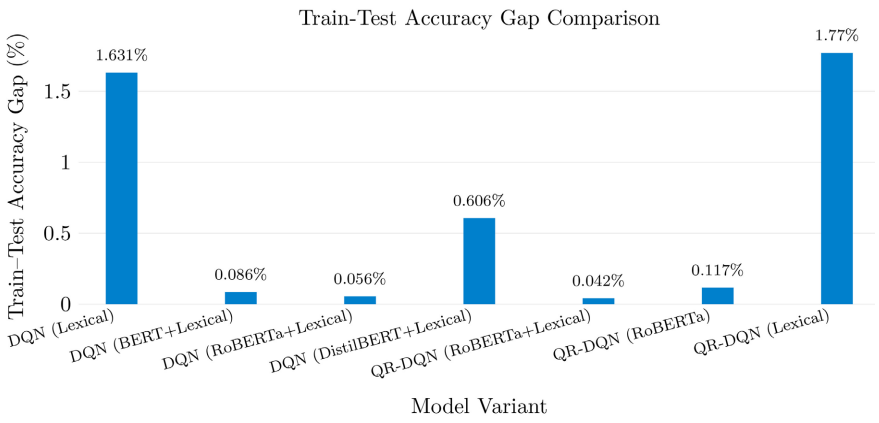

Figure 2. Train-test accuracy gap comparison across the evaluated phishing-detection models.

Table 1. Train-split performance comparison of the evaluated phishing-detection models.

Model Name	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	FP	FN
DQN (Lexical Features)	99.93	99.93	99.93	99.93	29	26
QR-DQN (RoBERTa + Lexical Features)	99.90	99.83	99.97	99.90	70	12
QR-DQN (Lexical Features)	99.92	99.91	99.92	99.92	32	28
DQN (RoBERTa + Lexical Features)	99.85	99.72	99.97	99.85	112	11
DQN (BERT + Lexical Features)	99.83	99.77	99.88	99.83	92	47
QR-DQN (RoBERTa)	99.69	99.53	99.85	99.69	189	61
DQN (DistilBERT + Lexical Features)	99.34	99.27	99.41	99.34	291	236

Table 2. Test-split performance comparison of the evaluated phishing-detection models.

Model Name	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	FP	FN
QR-DQN (RoBERTa + Lexical Features)	99.86	99.75	99.96	99.85	25	4
DQN (RoBERTa + Lexical Features)	99.79	99.63	99.95	99.79	37	5
QR-DQN (Lexical Features)	98.12	99.82	96.40	98.08	17	359
DQN (BERT + Lexical Features)	99.74	99.64	99.84	99.74	36	16
QR-DQN (RoBERTa)	99.57	99.38	99.76	99.57	62	24
DQN (DistilBERT + Lexical Features)	98.73	99.29	98.16	98.72	70	183
DQN (Lexical Features)	98.30	99.82	96.76	98.27	17	323

Table 3. Overfitting and generalization-gap comparison across model variants.

Model Name	Test Accuracy (%)	Accuracy Gap (%)	Test F1 (%)	F1 Gap (%)
QR-DQN (RoBERTa + Lexical Features)	99.86	0.042	99.85	0.043
DQN (RoBERTa + Lexical Features)	99.79	0.056	99.79	0.057
QR-DQN (Lexical Features)	98.12	1.77	98.08	1.81
DQN (BERT + Lexical Features)	99.74	0.086	99.74	0.087
QR-DQN (RoBERTa)	99.57	0.117	99.57	0.118
DQN (DistilBERT + Lexical Features)	98.73	0.606	98.72	0.618
DQN (Lexical Features)	98.30	1.631	98.27	1.663

5. Discussion

The proposed RoBERTa + QR-DQN framework shows substantial improvement over prior phishing-detection approaches, but the key point is that QR-DQN does not improve generalization on its own. Compared with transformer-only methods such as URLTran, which reported an 86.80% true positive rate under a strict false-positive threshold, the proposed hybrid framework achieved 99.86% accuracy and 99.96% recall on held-out test data, indicating a major improvement in detection capability. Relative to the Deep Q-Network (DQN) model of Chatterjee *et al.*, which used a single agent and 14 lexical features, the RoBERTa-QR-DQN agent achieved higher overall accuracy. DDQN-based unbalanced classifiers such as the

approach discussed by Maci *et al.* explicitly address skew through ICMDP reward design and report strong performance under varying imbalance ratios without data sampling [30]. Compared with earlier CNN-BiGRU hybrid systems, which also reported strong accuracy and recall [31], the RoBERTa-QR-DQN framework offers a major advantage in generalization only when QR-DQN is combined with lexical features and semantic embeddings.

The results suggest that the reduction in the train–test accuracy gap is driven by the hybrid state representation rather than by distributional reinforcement learning alone. In the lexical-only setting, QR-DQN does not outperform the corresponding DQN baseline in generalization; instead, the strongest gains appear when QR-DQN is paired with RoBERTa semantic embeddings together with engineered lexical features. This combined representation enables the model to capture advanced evasion patterns such as brand mimicry, typosquatting, URL obfuscation, and structural manipulation more effectively than lexical features alone. In that setting, the quantile-based distributional reinforcement learning framework contributes uncertainty-aware and risk-sensitive decision-making, while the asymmetric reward structure encourages the agent to reduce costly security mistakes and the learned offline cost-sensitive policy helps control false alarms without sacrificing phishing detection. Overall, the experimental evidence indicates that QR-DQN improves generalization only when it operates on a richer hybrid representation that includes both lexical and semantic embeddings.

6. Conclusions

This study shows that combining semantic feature enhancement with advanced reinforcement learning improves test recall and substantially reduces the train–test generalization gap relative to lexical-feature-only baselines. The main conclusion is not that QR-DQN by itself is better than DQN with lexical-only features; rather, the advantage appears when reinforcement learning is paired with contextual embeddings such as RoBERTa together with lexical features. While the DQN agent using only lexical inputs achieves high training accuracy, it suffers from a larger generalization gap and a substantially higher false negative rate, indicating that lexical-only pattern matching is insufficient against diverse phishing strategies. In contrast, agents that incorporate semantic embeddings achieve higher test accuracy and, more importantly, stronger generalization and resilience against adversarial manipulation.

These improvements suggest that the model benefits from learning semantic intent and structural meaning from URLs rather than merely exploiting superficial artifacts. From a security perspective, the findings emphasize that missing a phishing attack is substantially more costly than raising a false alarm, and that reinforcement-learning models become more effective in shrinking the generalization gap when they are augmented with semantic embeddings. Overall, the best-performing model was QR-DQN with RoBERTa embeddings and lexical features, achieving 99.86% test accuracy, 99.75% precision, 99.96% recall, 99.85% F1-score,

25 false positives, 4 false negatives, and a train-test accuracy gap of only 0.04%.

7. Limitations and Future Work

7.1. Limitations

This study has several limitations that create important directions for future investigation.

1) **Computational cost of semantic embeddings:** Generating BERT-based embeddings is computationally expensive. Although pre-computation and caching reduce the online training burden, the initial feature-extraction stage remains resource-intensive. Very high-throughput real-time systems may therefore face latency and scalability constraints.

2) **Static offline training setting:** The agent was trained on a large but fixed static dataset. As a result, the current model does not adapt online to newly emerging phishing campaigns during deployment. Regular retraining with recent data may help maintain performance under a rapidly evolving threat landscape.

3) **Limited feature scope:** The present hybrid model combines BERT embeddings with 50 engineered lexical features, but it does not incorporate other potentially useful modalities such as visual similarity to known brands, DNS-level indicators, or network-traffic features. Including these additional signals may further improve detection capability and robustness.

7.2. Future Work

Several directions can be pursued to extend this work.

- **Online and continual learning:** A key future objective is to evolve the current static RL framework into an online continual-learning system. This would allow the agent to update its policy gradually from a continuous stream of URLs, thereby adapting to emerging attack strategies while mitigating catastrophic forgetting.
- **Domain-specific BERT fine-tuning:** Future work can explore fine-tuning the transformer encoder on large labeled corpora of phishing and benign URLs. Domain-adaptive fine-tuning may improve semantic representation quality and further strengthen downstream RL policy optimization.
- **Integration of explainable AI:** Improving the interpretability of agent decisions is important for operational trust and adoption. Future extensions may incorporate explainability methods such as SHAP or LIME to provide post hoc explanations indicating which lexical or semantic URL components most strongly influenced the agent's decision.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Putra, F.P.E., Ubaidi, U., Zulfikri, A., Arifin, G. and Ilhamsyah, R.M. (2024) Analysis

of Phishing Attack Trends, Impacts and Prevention Methods: Literature Study. *Brilliance: Research of Artificial Intelligence*, **4**, 413-421. https://doi.org/10.47709/brilliance.v4i1.4357

[2] Akbar, N. (2014) Analysing Persuasion Principles in Phishing Emails. Ph.D. Thesis, University of Twente.

[3] Ejaz, A., Mian, A.N. and Manzoor, S. (2023) Life-Long Phishing Attack Detection Using Continual Learning. *Scientific Reports*, **13**, Article No. 11488. https://doi.org/10.1038/s41598-023-37552-9

[4] Nguyen, T.T. and Reddi, V.J. (2023) Deep Reinforcement Learning for Cyber Security. *IEEE Transactions on Neural Networks and Learning Systems*, **34**, 3779-3795. https://doi.org/10.1109/tnnls.2021.3121870

[5] Sarker, I.H. (2021) Deep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective. *SN Computer Science*, **2**, Article No. 154. https://doi.org/10.1007/s42979-021-00535-6

[6] Su, M. and Su, K. (2023) Bert-Based Approaches to Identifying Malicious URLs. *Sensors*, **23**, Article 8499. https://doi.org/10.3390/s23208499

[7] Mnih, V., Kavukcuoglu, K., Silver, D., *et al.* (2015) Human-Level Control through Deep Reinforcement Learning. *Nature*, **518**, 529-533. https://www.nature.com/articles/nature14236

[8] Liu, R., Wang, Y., Xu, H., Qin, Z., Zhang, F., Liu, Y., *et al.* (2025) PMANet: Malicious URL Detection via Post-Trained Language Model Guided Multi-Level Feature Attention Network. *Information Fusion*, **113**, Article 102638. https://doi.org/10.1016/j.inffus.2024.102638

[9] AVS Kumar, S., *et al.* (2024) Phishing Email Detection Using Machine Learning. *International Journal of Artificial Intelligence and Data Analysis*, **11**, 48-59.

[10] Rao, R.S., Kondaiah, C., Pais, A.R. and Lee, B. (2025) A Hybrid Super Learner Ensemble for Phishing Detection on Mobile Devices. *Scientific Reports*, **15**, Article 16308. https://doi.org/10.1038/s41598-025-02009-8

[11] Chatterjee, M. and Namin, A. (2019) Detecting Phishing Websites through Deep Reinforcement Learning. 2019 *IEEE* 43*rd Annual Computer Software and Applications Conference* (COMPSAC), Milwaukee, 15-19 July 2019, 227-232. https://doi.org/10.1109/compsac.2019.10211

[12] Lopez-Martin, M., Carro, B. and Sanchez-Esguevillas, A. (2020) Application of Deep Reinforcement Learning to Intrusion Detection for Supervised Problems. *Expert Systems with Applications*, **141**, Article 112963. https://doi.org/10.1016/j.eswa.2019.112963

[13] Terranova, F., *et al.* (2024) Leveraging Deep Reinforcement Learning for Cyber-Attack Path Discovery. ACM Digital Library.

[14] Sahingoz, O.K., Buber, E., Demir, O. and Diri, B. (2019) Machine Learning Based Phishing Detection from URLs. *Expert Systems with Applications*, **117**, 345-357. https://doi.org/10.1016/j.eswa.2018.09.029

[15] Kheddar, H., Dawoud, D.W., Awad, A.I., Himeur, Y. and Khan, M.K. (2024) Reinforcement-Learning-Based Intrusion Detection in Communication Networks: A Review. *IEEE Open Journal of the Communications Society*, **5**, 2115-2141.

[16] Devlin, J., Chang, M., Lee, K. and Toutanova, K. (2019) BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North, Minneapolis, 2 June-7 June 2019, 4171-4186. https://doi.org/10.18653/v1/n19-1423

[17] Young, T., Hazarika, D., Poria, S. and Cambria, E. (2018) Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Computational Intelligence Magazine*, **13**, 55-75. https://doi.org/10.1109/mci.2018.2840738

[18] Maneriker, P., Stokes, J.W., Lazo, E.G., Carutasu, D., Tajaddodianfar, F. and Gururajan, A. (2021). URLTran: Improving Phishing URL Detection Using Transformers. *MILCOM 2021-2021 IEEE Military Communications Conference* (*MILCOM*), San Diego, 29 November-2 December 2021, 197-204. https://doi.org/10.1109/milcom52596.2021.9653028

[19] Saleem Raja, A., Vinodini, R. and Kavitha, A. (2021) Lexical Features Based Malicious URL Detection Using Machine Learning Techniques. *Materials Today*: *Proceedings*, **47**, 163-166. https://doi.org/10.1016/j.matpr.2021.04.041

[20] Ari Kustiawan, Y. and Ghauth, K.I. (2025) Evaluating the Impact of Feature Engineering in Phishing URL Detection: A Comparative Study of URL, HTML, and Derived Features. *IEEE Access*, **13**, 126756-126768. https://doi.org/10.1109/access.2025.3579223

[21] Asif, A.U.Z., Shirazi, H. and Ray, I. (2023) Machine Learning-Based Phishing Detection Using URL Features: A Comprehensive Review. In: Dolev, S. and Schieber, B., Eds., *Lecture Notes in Computer Science*, Springer, 481-497. https://doi.org/10.1007/978-3-031-44274-2_36

[22] Brockman, G., Cheung, V., Pettersson, L., *et al.* (2016) OpenAI Gym. arXiv:1606.01540. https://arxiv.org/abs/1606.01540

[23] Luong, N.C., Hoang, D.T., Gong, S., Niyato, D., Wang, P., Liang, Y., *et al.* (2019) Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Communications Surveys & Tutorials*, **21**, 3133-3174. https://doi.org/10.1109/comst.2019.2916583

[24] Mnih, V., Kavukcuoglu, K., Silver, D., *et al.* (2013) Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602. https://arxiv.org/abs/1312.5602

[25] Kim, T., Park, N., Hong, J. and Kim, S. (2022) Phishing URL Detection: A Net-Work-Based Approach Robust to Evasion. *Proceedings of the* 2022 *ACM SIGSAC Conference on Computer and Communications Security*, Los Angeles, 7-11 November 2022, 1679-1782. https://doi.org/10.1145/3548606.3560615

[26] Otieno, D.O., Abri, F., Namin, A.S. and Jones, K.S. (2023) Detecting Phishing URLs using the BERT Transformer Model. 2023 *IEEE International Conference on Big Data* (*BigData*), Sorrento, 15-18 December 2023, 1303-1310.

[27] Sutton, R.S. and Barto, A.G. (2018) Reinforcement Learning: An Introduction. 2nd Edition, MIT Press.

[28] Dabney, W., Rowland, M., Bellemare, M. and Munos, R. (2018) Distributional Reinforcement Learning with Quantile Regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, **32**, 2892-2901. https://doi.org/10.1609/aaai.v32i1.11791

[29] Bellemare, M.G., Dabney, W. and Munos, R. (2017) A Distributional Perspective on Reinforcement Learning. *Proceedings of the* 34*th International Conference on Machine Learning*, Sydney, 6-11 August 2017, 449-458.

[30] Maci, A., Santorsola, A., Coscia, A. and Iannacone, A. (2023) Unbalanced Web Phishing Classification through Deep Reinforcement Learning. *Computers*, **12**, Article 118. https://doi.org/10.3390/computers12060118

[31] Egigogo, O.E., Idris, I.A., Olalere, O.M., Abisoye, O.G. and Ojeniyi, B.A. (2022) Development of Hybridized CNN-BiGRU Framework for Detection of Website Phishing Attacks. *Nigerian Journal of Technological Research*, **3**, 45-54.