IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 June 2022, accepted 26 July 2022, date of publication 3 August 2022, date of current version 10 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3196018

RESEARCH ARTICLE

Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML

SUBHASH ARIYADASA[1,2], SHANTHA FERNANDO[3], AND SUBHA FERNANDO[1], (Member, IEEE)

[1]Department of Computational Mathematics, University of Moratuwa, Moratuwa 10400, Sri Lanka
[2]Department of Computer Science and Informatics, Uva Wellassa University, Badulla 90000, Sri Lanka
[3]Department of Computer Science and Engineering, University of Moratuwa, Moratuwa 10400, Sri Lanka

Corresponding author: Subhash Ariyadasa (188077d@uom.lk)

ABSTRACT Phishing, a well-known cyber-attack practice has gained significant research attention in the cyber-security domain for the last two decades due to its dynamic attacking strategies. Although different solutions have been exercised against phishing, phishing attacks have dramatically increased in the past few years. Recent studies have shown that machine learning has become prominent in the present anti-phishing context, and the techniques like deep learning have extensively improved anti-phishing tools' detection ability. This paper proposes PhishDet, a new way of detecting phishing websites through Long-term Recurrent Convolutional Network and Graph Convolutional Network using URL and HTML features. PhishDet is the first of its kind, which uses the powerful analysis and processing capabilities of Graph Neural Network in the anti-phishing domain and recorded 96.42% detection accuracy, with a 0.036 false-negative rate. It is effective against zero-day attacks, and the average detection time which is 1.8 seconds could also be considered realistic. The feature selection of PhishDet is automatic and occurs inside the system, as PhishDet gradually learns URLs and HTML content features to handle constantly changing phishing attacks. This has outperformed similar solutions by achieving a 99.53% f1-score with a public benchmark dataset. However, PhishDet requires periodic retraining to maintain its performance over time. If such retraining could be facilitated, PhishDet could fight against phishers for a more extended period to safeguard Internet users from this Internet threat.

INDEX TERMS Cyberattack, deep learning, graph neural networks, internet security.

I. INTRODUCTION

Phishing is a cyber-attack that lures the victim, using a technological bait to collect personal or confidential information [1]. It started in 1995 with the American Online (AOL) attack. Afterwards, the phishers-individuals or teams steering the phishing attacks, moved to more profitable targets such as online banking and e-commerce services [1], [2]. Financial gains are the primary motivating factors for phishers; however, fame and notoriety are also interesting psychological aspects of phishing [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Seifedine Kadry.

Phishing is an Internet threat and has a top rank in the cyber threat landscape [4]. It has become a leading cyber threat to the financial sectors [5] and has spread into many sectors [6]. The recent statistics show the number of phishing attacks doubled in 2020 compared to the past, and nearly 84% of phishing sites have been recorded in the latter part of 2020 which used SSL protection [6]. It indicates that HTTPS is not a vital feature when detecting phishing attacks at present. In fact, half of the phishing attacks' lifetime ended in less than a day [7], and new phishing trends have also emerged rapidly due to the dynamic nature of phishing attacks [6].

In the past few years, researchers have used different approaches to fight against phishers (Table 1), and these

approaches could be categorised into machine learning and non-machine learning. Machine learning that learns things independently [8] shows some success in the phishing domain in the past due to its advantages which includes coping with frequent data changes and automating the learning process [2]. However, most machine learning-based solutions use manual feature engineering techniques (Table 2) with some drawbacks [9].

In manual feature engineering, the features are handcrafted, and the attackers often bypass these handcrafted features due to the visibility of these to the outside [9]. Furthermore, selecting the best set of features in a specific domain is also a challenging task in manual feature engineering, and it highly depends on experts' knowledge [2], [9]. Therefore, many recent phishing detection studies [2], [9]–[13] have focused on integrating representation learning that learns a representation of data through multiple abstraction levels to extract features for the learning process [14].

However, many of these efforts [10], [11], [15], [16] have concentrated on the URL-based phishing detection category, and just a few researches [12], [13] have efficiently applied representation learning in HTML content analysis. These researchers have used Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) for HTML content analysis. However, they have not used the Graph Neural Network (GNN) technique, a powerful technique for extracting phishing detection features from HTML contents [13]. Since HTML content has a graph structure, GNN would be ideal for analysing its phishing characteristics [13]. Therefore, this study focuses on a differentiated phishing detection approach that uses the GNN technique for the first time in the anti-phishing domain to cope with the increasing number of phishing attacks.

PhishDet is proposed as a representation learning-based phishing detection solution that uses the URL and the HTML content of a website. It combines two deep networks – i.e. the Long-term Recurrent Convolutional Network (LRCN) and the Graph Convolutional Network (GCN). It performed well during the experiments and achieved 96.42% detection accuracy with a real-world public dataset. PhishDet has been further tested with a benchmark dataset, and it has outperformed similar solutions [12], [13] by recording 99.57% detection accuracy.

The key contributions of this paper are, 1). PhishDet - A phishing detection solution that automatically selects features from a website's raw URL and HTML content; 2). The first GNN based phishing detection approach that uses raw HTML content.

The rest of this paper is organised as follows. In Section II, related work in the phishing detection area is discussed, while Section III is the proposed solution, with Section IV explaining how the experiment was done. Then the results obtained and the performance of the proposed solution is presented in Section V. Finally, in Section VI, the paper summarises the work done with some future directions.

II. RELATED WORK

Phishing is a prevalent Internet threat that attached various definitions due to its changing nature [2], [17]. The previous studies defined phishing in the context of its use and more prominently under the social engineering-based cyber attacks [3], [17], [18]. Phishers are not only targeting human psychology but are also keen to use technical methods like malicious software installation and pharming to steal users' digital assets directly or indirectly [17]. Therefore, phishing attacks could be classified into two categories: deceptive phishing and technical subterfuge [17]. Deceptive phishing is the most common type of phishing attack experienced at present, and spoofed website is the dominant technique today [17]. Anti-Phishing Working Group (APWG) showed the situation precisely by detecting more than fifteen hundred thousand unique phishing websites in 2020 [6].

Phishing is defined as a social engineering crime in the current study that leads a victim to a fake website to steal personal or confidential information [6], [17], [18]. The phishers impersonate trusted third parties when developing these fake websites, and different tactics like clicking a link are exercised to move victims to the fake websites [1], [17], [18].

In the past two decades, academia and the industry researched better detection approaches to combat phishing attacks, but it seemed challenging due to the nature of phishing attacks [7]; however, there are now differentiated solution that protects the privacy of Internet users against phishing. These solutions could be clustered into several approaches, and these approaches could be categorised into machine learning and non-machine learning, as shown in Table 1. Although existing solutions were classified into two categories, they differ from each other, as shown in Table 2. Therefore, past work was reviewed closely by two topics related to the current study: feature selection and engineering, and the phishing detection approach. Consequently, the findings gathered through this analysis are connected in Section II-C to discuss how the proposed solution is designed based on the literature.

A. FEATURE SELECTION AND ENGINEERING

Feature selection plays an essential role in the anti-phishing domain since it directly affects phishing detection accuracy [2], [38]. Phishing detection features have been categorised under different feature sets and those used in different ways in previous studies. One such categorisation includes four main features: URL-based lexical features, URL-based host features, website page content, and visual similarity [2]. URL-based lexical features, such as URL length, number of dots, number of sub-domains, and the use of HTTPS protocol, have been considered features that can be extracted directly from a URL. The URL-based host features primarily rely on third-party services such as WHOIS data.

The next two categories are based on the web page. These include the content characteristics like page rank, links and forms, and visual appearances like texts, images, and colours.

TABLE 1. Overview of the standard phishing detection approaches.

Category	Approach	Limitations / Remarks	Sources
Machine Learning	Supervised Learning • A model trains from known phishing and legitimate data	• It depends on a set of features (i.e. URL features) • A learning algorithm uses to adjust the weights of these features to achieve optimum performance	[8]–[11], [14], [19]
	Reinforcement Learning • An agent is used to gather its experience in web surfing for sequential decision making	• Agent produces an action (i.e. access or block a website) from a set of website features • The correctness of an agent's action is measured to have an effective learning process	[20], [21]
Non-machine Learning	User Awareness • This technique trains Internet users to access the Internet services safer to protect them from phishing attacks	• A machine-centric approach • Game-based education has been found as an effective method when improving the user-awareness • Expecting users to get educated about technological things like phishing is not practical	[22], [23]
	Blacklisting & Whitelisting • Blacklisting contains a list of phishing website URLs • Whitelisting is a list of legitimate website URLs	• It requires exact matching of the website URLs • It fails when detecting zero-day attacks since those may not include in the lists • Practical difficulties exist to have an up-to-date list	[19], [24]–[27]
	Rule-based Heuristics • A technique that uses a set of rules when detecting phishing attacks	• Domain expertise is essential to constructing high-end rules • The rules need frequent updates to keep alive • The cost of updating rules is high	[26], [28]–[31]
	Visual Similarity • This technique uses the visual appearance of the web page in phishing detection	• It depends on a threshold value, and difficult to find the optimum value • It uses visual features such as text, HTML tags, CSS and images • Maintaining an up-to-date database is challenging; therefore, it fails to detect zero-day attacks • The detection time is relatively high	[5], [32]–[34]
	Data Mining • A technique that extracts data to discover hidden phishing patterns in a given dataset to implement predictive models	• It is not categorised under machine learning since the model does not learn over time • It focuses on finding new and interesting phishing patterns without getting a specific goal from the domain • The best features selection is a challenging task • The findings are used with rule-based heuristics to enhance the classification accuracy	[2], [8], [35]–[37]

In a separate study [38], phishing detection features were listed under URLs and web pages. URLs and web pages were mainly divided into lexical, script, and network classes. Each of these was further divided into the syntactic, semantic and pragmatic levels to form a complete relationship among the available phishing detection features. Furthermore, another study [39] proposed six categories of features, and yet a different study [19] grouped thirty features under four different feature sets.

However, in real-time phishing detection systems, features that rely on third-party services like page ranking and domain age should be avoided due to the increased detection time, high development cost, and limitations on the services themselves [7], [40]. Therefore, Li *et al.* [7] have proposed URL and HTML-related features that do not rely on third-party services. It is interesting to note that if a feature cannot be mined quickly, no matter how valid, it results in user inconveniences through service delays [2].

The current study focuses on three main feature sets when evaluating existing anti-phishing solutions. These are URL-based features, content-based features and external features. URL-based features are lexical features that can be extracted directly from the URL, and content-based features are extracted from HTML content. All the other features that are not directly available with URL or HTML content, such as page rank, browser history, google index, and domain age, are listed under external factors. Table 2 shows a summary of the selected feature sets in different solutions.

According to Table 2, some solutions depend only on the URL. However, the URL shortening services that hide original URLs and URL simulation tools such as DeepPhish [41] that generate malicious URLs could challenge

these solutions in the long run [42]. The Generative Adversarial Networks (GAN) have been identified as a challenge for URL-only solutions, and those could be used to evade the URL-based phishing detection solutions [43]. Further, the possibility of benign URLs becoming malicious in the future has been considered a key element in URL-based anti-phishing solutions [42]. Therefore, URL information alone is not practical since it does not represent the phishing characteristics entirely, and multidimensional features are vital when developing anti-phishing solutions [44]. As a result, the current study combines URL-based features with content-based features to form better phishing detection.

Like the feature sets, feature engineering (a.k.a., feature extraction) also significantly impacts the detection accuracy of an anti-phishing solution [2]. If a feature can easily be manipulated by the attackers and has a strong relationship with accuracy, the solution will no longer work effectively. Hence, if a solution used a complicated feature and took more time to mine, the solution would not be practical again due to service delays [2]. Therefore, feature engineering is crucial when building a phishing detection solution. After exploring the literature, the current study classified feature engineering techniques essentially into two categories: manual and representation learning. The manual technique involves human experts, and the extraction also can be done using a computer program built by a human expert [45]. However, the human user actively participates in the feature engineering process [45].

Many of the solutions available in Table 2 used manual feature engineering techniques due to the controllable environment it gives, such as carefully picking the most important features through different analyses [27]–[29], [33]. However, the validity of the manually selected features for the next few years is a debatable question. 'HTTPS' in anti-phishing solutions can showcase the situation precisely. The 'HTTPS' feature was famous in older anti-phishing solutions [19], [39], but it is not practical at present [2], [6]. It indicates that the significant phishing detection features are outdated over time, and the features need to be updated frequently to have a robust detection. However, such updating is difficult with the manual feature extraction technique due to experts' involvement, and challenges exist when selecting the best features [9]. Even though the manually extracted features have not changed for a more extended period, these features are primarily handcrafted, and the extracted features are visible [9]. Hence attackers could easily bypass these solutions by targeting these features.

Representation learning is ideal for overcoming such challenges since it is more suitable for classification problems like phishing [14]. Therefore, it is popular in the anti-phishing domain at present [9], [10], [12], [13]. Representation learning gets raw data as inputs and automatically discovers the relevant representations needed through multiple levels of abstractions for a given task without any manual feature engineering [14]. Since this approach has achieved significant results in the research field [9], [10], [12], [13], the current study also focuses on a representation learning-based approach to detect phishing attacks.

B. PHISHING DETECTION APPROACHES

The explored phishing detection solutions have been inspired by machine learning and non-machine learning approaches, including improving user awareness, blacklisting/whitelisting, visual similarity, rule-based heuristic and data mining, as shown in Table 2.

1) MACHINE LEARNING-BASED PHISHING DETECTION

Sir Arthur Samuel defined machine learning as "a field of study that gives computers the ability to learn without being explicitly programmed" [8]. It originated more than 70 years ago and became a popular technique after the new millennium due to a large amount of data available in different sectors. Machine learning mainly focuses on classification problems like phishing and uses previous examples to build effective models [8]. Most machine learning-based phishing detection approaches have been categorised under supervised learning, and reinforcement learning has also been practised in one study, as shown in Table 2. As a result, the previous phishing detection efforts in machine learning could be categorised as supervised learning and reinforcement learning. Table 1 describes these categories in detail.

According to Table 2, phishing detection was more towards machine learning during the last decade, and it all began with a single-layer neural network solution. This solution used four URL features and six heuristics rules to achieve 98.43% detection accuracy [46]. Although this solution achieved an interesting accuracy, it proposed further enhancements for the solution with more heuristics and large data sources. Phish-Safe was proposed as another way to detect phishing attacks with the support of the Support Vector Machine (SVM) technique [36]. This solution used fourteen URL features and was evaluated using 32,951 phishing data collected from Phish-Tank. Phish-Safe recorded more than 90% detection accuracy during the evaluation. In a study done by Sahingoz *et al.* [40], seven different algorithms were exercised to find the best machine learning algorithm against phishing attacks. This approach used Natural Language Processing (NLP) based features and word vectors that focused on the words in a URL. This study has shown that the NLP-based features with Random Forest (RF) algorithm perform well in phishing detection.

Similarly, Subasi *et al.* [47] also found that the RF algorithm is better in phishing detection since it recorded higher accuracy and fast detection time during the experiment. Moreover, a self-structuring Multi-Layer Perceptron (MLP) network with seventeen HTML and URL features has been exercised in the literature to detect phishing attacks [39]. This self-structuring network differs from the previous solutions because this network can automatically adapt the neural network structure during the learning process, which has not been seen in others. However, this solution only achieved 92.5% detection accuracy during the experiment.

In a separate study, Pratiwi *et al.* [48] proposed a neural network approach using the same features available in the literature [39]. Even though it used the same features, only 83.4% detection accuracy was achieved during this study.

Further, the first stack model introduced in the anti-phishing domain used Gradient Boosting Decision Tree (GBDT), eXtreme Gradient Boosting (XGBoost), and Light Gradient Boosting Machine (LightGBM) in multiple layers [7]. This model used eight URLs and twelve HTML-based features that did not rely on third-party services. Interestingly, the first stack model achieved 97.3% detection accuracy during the experiment. Similarly, a Multidimensional Feature Phishing Detection (MFPD) approach [44] with XGBoost and CNN-LSTM techniques was also proposed in the past. It first used the CNN-LSTM deep network to classify phishing URLs based on character sequence features. Then that decision was combined with other multidimensional features such as URL statistical features, webpage code and text features in an XGBoost classifier when producing the final decision. However, MFPD recorded a higher detection time during the experiment, and it was minimised to 3.5 seconds through a specific threshold value by controlling the feature extraction process. In a separate study, PhiDMA [49] used five layers during phishing detection. During the implementation, PhiDMA considered visually impaired users and therefore included specific support.

Bahnsen *et al.* [10] proposed an LSTM network against phishing URLs and recorded 98.7% detection accuracy. Later, another similar solution [11] achieved 99.1% detection accuracy against phishing URLs. However, the approach used in this study differs from the one by Bahnsen *et al.* from the feature engineering perspective because this study used manually extracted features. Literature has also shown that a combination of LSTM with CNN is better when detecting malicious URLs [50]. As a result, an anti-phishing solution named PDRCNN was proposed, which used LSTM and CNN to have a successful phishing detection [15]. This used only the URLs, and the experiment was performed with 500,000 data from PhishTank and Alexa. Further, PDRCNN used 0.4ms to detect a given URL, and as a result, it was considered a speedier solution. However, the PDRCNN dataset is problematic because the combination of Alexa and PhishTank URLs could prioritise the URL length feature during the training, and it may produce a wrong interpretation of the model performance [38], [51].

In the past, another LSTM and CNN-based deep network [52] achieved 98.3% detection accuracy. However, this solution is not like PDRCNN since this approach uses URL and HTML features. Furthermore, representation learning was used only to extract URL features in this solution, while the HTML features were extracted manually. In contrast, HTMLPhish [9] used representation learning to extract features from HTML content. It achieved 97.2% detection accuracy, and the HTML document was used directly in the learning process. A few months later, the HTMLPhish team introduced another phishing detection solution called WebPhish [12], a deep learning-based phishing detector that selects features from the raw URLs and HTML content. It was the first representation learning approach that supports raw URLs and HTML in phishing detection, and this solution achieved 98% accuracy on a real-world dataset. However, both HTMLPhish and WebPhish demonstrated that their phishing detection ability was declining with time, and a successful retraining phase was used to support them in regaining their earlier performance.

Like WebPhish, Web2Vec [13] is another anti-phishing solution that uses raw URLs and HTML content in phishing detection. It extracted features automatically in run time and outperformed most of the latest solutions (Table 2) by achieving 99% detection accuracy. Web2Vec inputted the URL, HTML content and Document Object Model (DOM) structure of webpages to a deep hybrid network and applied an attention mechanism to strengthen the solution. However, this work highlighted the effectiveness of GNN in HTML content analysis over the proposed architecture and mentioned it in their future work. Although WebPhish and Web2Vec showed some interesting accuracy, these solutions used Alexa and PhishTank to collect the required URLs and did not mention any specific strategy to avoid the URL length issue mentioned earlier. This indicates that these solutions may also be biased to the URL length issue, and it may have affected the presented performances [38], [51].

Even though Web2Vec and WebPhish detected phishing attacks through multidimensional feature sets, a pure URL-based approach called PhishHaven [16] sensed the phishing problem differently. It divided the phishing URLs into human-crafted and AI-generated phishing URLs since the solutions like DeepPhish [41] could generate challenging phishing URLs automatically. PhishHaven was the first approach found in the literature that emphasises the importance of a separate detection mechanism for AI-generated phishing URLs. It achieved 98% accuracy, and the authors mentioned that it could detect AI-generated phishing URLs with higher accuracy; theoretically, it could be 100%.

The only solution listed under reinforcement learning (Table 2) was proposed by Chatterjee and Namin [20]. It was a URL-based phishing detection that attempted to apply dynamic behaviour to detect phishing attacks. The proposed solution achieved only 90% detection accuracy, and the f1-score of the solution was 87.3%. However, this work opened a new window for researchers to consider continuous learning support in future anti-phishing solutions.

Even though machine learning has shown some promising results in previous work, it faces unique challenges when dealing with constantly changing phishing attacks. One such challenge is data drifting [38], [42]. It mainly affects the solution's performance, and the solution needs to be retrained occasionally to retain its performance [38]. However, retraining is again a challenge in the anti-phishing domain, as collecting a considerable amount of labelled phishing data is difficult in the current context [42], [53]. Furthermore, adversarial attacks are also considered a threat to machine

learning-based anti-phishing solutions [42]. These attacks can convert well-performing models into incorrect predictions to reduce the trustworthiness of the solutions.

2) OTHER PHISHING DETECTION APPROACHES

Other than the machine learning-based solutions, the existing phishing detection solutions could be classified into five main approaches, as shown in Table 1. However, some explored solutions have come under more than one approach (e.g., PhishNet and AIWL) because of their characteristics. The following describes each of these categories in detail.

a: USER AWARENESS

AntiPhishing Phil [22] and Smells Phishy [23] are interactive educational games found in the literature that reduce phishing victims through user awareness. These solutions used a gaming environment to teach phishing concepts joyfully. Other than the gaming approach, some organisations have published training materials to improve phishing attack awareness [26]. Microsoft Online safety, OnGuardOnline phishing tips, and National Consumer League Fraud tips are some famous examples of training materials [26]. Further, how the Internet users interact with phishing attacks and how they behave in such attacks have been identified as an important aspect of user awareness [54]. This information assist system designers and security professionals in predicting future victims, making them aware of these attacks [54].

Even though user awareness is being practised to minimise phishing impact, it may not be a viable approach because phishing attacks have constantly been changing, responding to the latest security countermeasures [17]. Therefore, user awareness is considered a costly approach since users need to regularly keep up to date on these attacks, which requires many resources such as people, time, and physical equipment [17], [26]. Further, the user awareness technique also expects considerable security knowledge from the trainees to have a successful training program which is again impractical in phishing detection [17], [26].

b: BLACKLISTING/WHITELISTING

Blacklisting and whitelisting are more popular phishing detection techniques that use simple text-matching to detect phishing attacks [19]. Google Safe Browsing (GSB) API,[1] maintained by Google LLC, is a famous blacklist used by different Internet services. Even though this list-based technique has been using a simple strategy to detect phishing websites, maintaining a practical list is challenging due to the number of new websites appearing on today's Internet [6]. As stated by Khonji *et al.* [26], 47% to 83% of phishing URLs took twelve hours to appear on a blacklist from their first appearance. It is a considerable delay because nearly 63% of phishing websites end their duties within the first two hours [26]. This indicates that these phishing attacks may have victimised many users once the list is updated. Therefore, the blacklisting/whitelisting technique is always vulnerable to zero-day attacks, which is considered a significant drawback of this technique [19], [26]. Further, an effective list always includes reporting and confirmation process to maintain the quality of its content [27]. It is again a challenge in phishing detection due to the number of phishing attacks added to the Internet and the life-time of these attacks.

The literature has introduced several alternative approaches to the blacklisting/whitelisting technique as a solution to these limitations. One such solution is PhishNet [24]. PhishNet used approximate matching of URLs integrated heuristic rules with the blacklisting technique. It also used a predictive blacklist mechanism that predicts a set of URLs based on a given URL to defend against dynamic phishing attacks. An Automated Individual White-List (AIWL) [25] is another solution that has maintained an individual whitelist with the support of login user interfaces. Even though it takes time to collect legitimate URLs specific to the user, this AIWL will be more stable and fits the user with time since the users are typically using the same set of websites [25]. Similarly, Jain and Gupta [27] also proposed a White-List maintainer, which automatically maintains a whitelist. It used hyperlink features to decide the legitimacy of a web page added to the whitelist. Even though these alternative solutions tried to eliminate the limitations that exist with the blacklisting/whitelisting technique, El-Alfy [19] mentioned in another study that zero-day attack detection is still a problem in these list-based techniques.

c: VISUAL SIMILARITY

This is a time-consuming approach due to the comparison of visual elements [2], [5]. In this approach, the suspicious web page is compared to a list of legitimate web pages and checks the similarity based on a threshold value [5], [32], [34]. PhishZoo [34] is one solution that has used a profile-based approach using the website's URL, textual content, images, scripts, and SSL certificate. In another study, GoldPhish [33] was proposed as a browser-based plug-in to detect phishing attacks. It was very effective against popular organisations and could detect zero-day phishing attacks. However, GoldPhish's accuracy depended on optical character recognition, logo image, and Google ranking [5], [55]. Therefore, it is not a practical solution against non-popular legitimate web pages since such web pages have not been indexed in Google.

Rosiello *et al.* [32] DOM tree approach, which used HTML DOM in similarity measuring, and site signature-based phishing detection approach [56] are other visual similarity approaches found in the literature. Despite these approaches, visual similarity detection faces several challenges, including defining a clear similarity value, maintaining detection databases, ineffectiveness against zero-day attacks, and concerns with embedded object recognition [5]. As a result, visual similarity detection cannot be considered effective against modern phishing attacks, as they are mainly constructed through software tools that can create visually similar web pages.

[1] https://safebrowsing.google.com/

TABLE 2. Explored phishing detection solutions.

Solution	Detection Approach								Features			Feature Extraction Technique		Accuracy
	Supervised Learning	Reinforcement Learning	User Awareness	Blacklisting	Whitelisting	Rule-based Heuristics	Visual Similarity	Data Mining	URL-based	Content-based	External	Manual	Representation Learning	
SpoofGuard [30]						✓			✓	✓	✓	✓		*NS*
Anti-Phishing Phil [22]			✓						✓			✓		87.0%
DOMAntiPhish [32]							✓			✓		✓		*NS*
CANTINA [28]						✓			✓	✓	✓	✓		90.0%
AIWL [25]					✓	✓	✓			✓	✓	✓		100.0%
PhishGuard [29]						✓					✓	✓		*NS*
PhishNet [24]				✓		✓			✓			✓		*NS*
GoldPhish [33]						✓				✓	✓	✓		98.0%
PhishZoo [34]							✓		✓	✓	✓	✓		96.0%
Self-structuring MLP Network [39]	✓								✓	✓	✓	✓		92.5%
Single-layer Neural Network [46]	✓								✓		✓	✓		98.0%
Smells Phishy [23]			✓						✓	✓		✓		75.0%
White-List Maintainer [27]					✓	✓				✓	✓	✓		89.4%
Phishing URL Detection [35]								✓	✓			✓		93.0%
Probabilistic Neural Network (PNN) [19]	✓								✓	✓	✓	✓		96.8%
LSTM Network [10]	✓								✓				✓	98.7%
Random Forest Classifier [47]								✓	✓	✓	✓	✓		97.4%
Phish-Safe [36]	✓								✓			✓		90.0%
LSTM Recurrent Neural Network [11]	✓								✓			✓		99.1%
Phishing Detection using ANN [48]	✓								✓	✓		✓		83.4%
Deep RL based Detection [20]		✓							✓	✓	✓	✓		90.1%
Machine Learning based Detection [40]	✓								✓			✓		98.0%
HTMLPhish [9]	✓									✓			✓	97.2%
Stacking Model [7]	✓								✓	✓		✓		97.3%
PDRCNN [15]	✓								✓				✓	97.0%
MFPD Model [44]	✓								✓	✓	✓	✓	✓	99.0%
HybridDLM [52]	✓								✓	✓		✓	✓	98.3%
PhiDMA [49]	✓				✓				✓	✓	✓	✓		92.7%
PhishHaven [16]	✓								✓			✓		98.0%
WebPhish [12]	✓								✓	✓			✓	98.0%
Web2Vec [13]	✓								✓	✓			✓	99.0%

NS means 'Not Specified'. It is used when the solution's accuracy cannot be found.

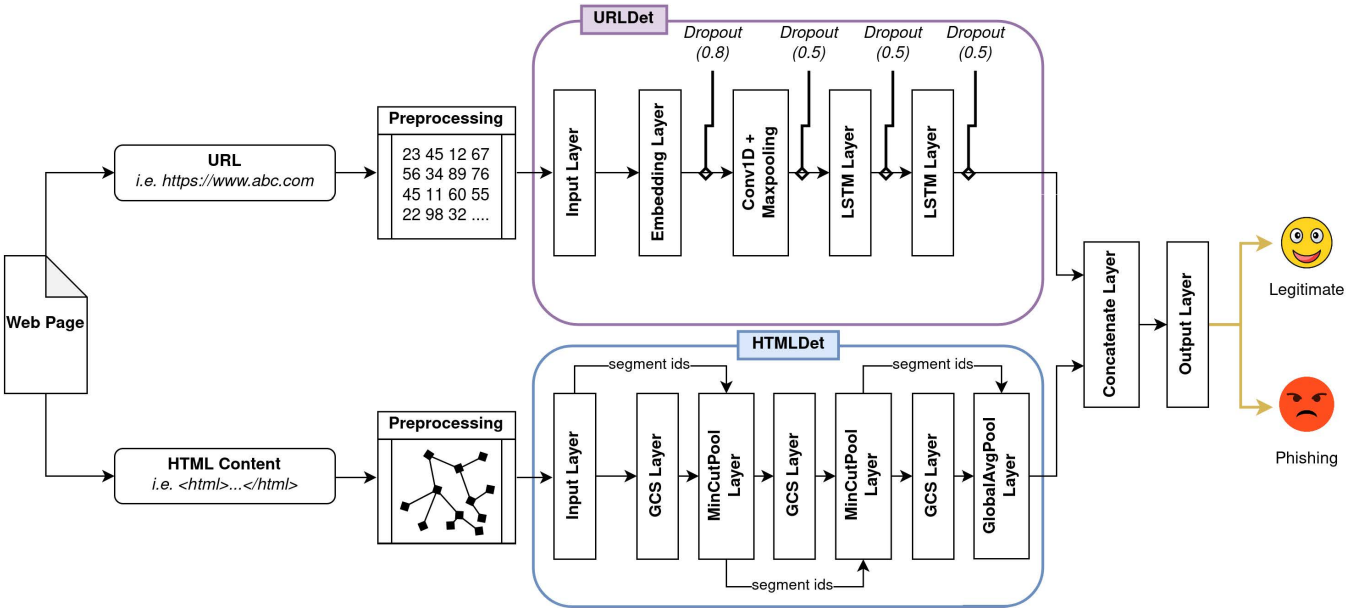


FIGURE 1. The architecture of the PhishDet model.

d: RULE-BASED HEURISTICS

SpoofGuard [30] is a rule-based heuristic approach that uses seven heuristics in phishing detection. It was developed as an anti-phishing toolbar for the Internet Explorer browser to transmit passive warnings to users about suspected data requests. PhishGuard [29] is another heuristic solution that uses HTTP digest authentication to detect phishing attempts. It has been an effective solution against zero-day attacks and is applied only in login interfaces. CANTINA [28] is a content-based solution that has been used in phishing detection with the Term Frequency/Inverse Document Frequency (TF-IDF) technique. Since CANTINA has shown a high false-positive rate, eight heuristics were applied later to reduce CANTINA's false-positive rate from 6% to 1%.

The rule-based heuristic technique is more successful than blacklisting/whitelisting since it can detect zero-day attacks [26]. Although it detects zero-day attacks, more generic algorithms are susceptible to misclassifying legitimate websites, a significant drawback of this strategy [26]. Heuristic-based techniques also have several drawbacks, such as rule visibility to the outside world, which supports phishers in devising solutions, rule validity owing to the continuously changing nature of phishing, and the cost of upgrading rules [26], [31].

e: DATA MINING

Data mining has been practised in the anti-phishing domain to detect phishing attacks. The fuzzy logic-based phishing detection solution is one data mining approach that uses three layers with six different phishing criteria [37]. In another study [35], eighteen association rules were also exercised with apriori and predictive apriori to achieve 93% detection accuracy. Although data mining techniques were used to identify phishing, Aburrous *et al.* [37] observed that finding a suitable collection of phishing detection features in data mining is challenging. On the other hand, data mining aims for new and interesting patterns that should be combined with other methodologies such as rule-based or machine learning to provide successful outcomes.

C. UNDERLYING CONCEPTS FOR STRONGER PHISHING DETECTION

This study extracted several underlying concepts supporting a successful phishing detection solution based on the analysis presented in previous literature sections. These ideas will be linked in the following paragraphs to provide a background for the techniques chosen when designing the PhishDet solution.

As illustrated in Table 2, machine learning attracted the most current phishing detection interest due to its unique learning ability, past success, and ineffectiveness of alternative techniques in the constantly changing phishing nature. Even though machine learning systems demonstrated some promising results, data drifting challenges have generally affected these solutions. As a result, regularly updating the core phishing detection features is critical for these anti-phishing solutions to function correctly [38].

However, due to the advantages that representation learning algorithms offered over manual methods, representation learning played a crucial role in feature extraction when updating significant phishing detection features. As a result, the representation learning approach called deep learning has had much success, and it has been utilised in many recent studies to extract features from raw data. Although deep

learning has been employed in phishing detection, it has primarily been utilised in URL-based phishing detection studies, with several drawbacks. However, as mentioned previously, the deep learning technique has recently been successfully applied to HTML content analysis in WebPhish and Web2Vec.

When it comes to detecting malicious URLs, LSTM has been considered a successful technique [10], [11]. However, in the past, combining LSTM and CNN increased the detection of malicious URLs [50], [52]. LRCN has been identified as one such combination which employs CNNs as a feature extractor for LSTMs in the front end [57]. Similarly, LSTM and CNN have been used to obtain notable results in HTML content analysis [12], [13]. Even though these techniques have been used effectively, past findings emphasised that GNN could be a powerful technique for analysing HTML content because HTML naturally has a graph structure [13]. Furthermore, a deeper analysis of HTML content might be possible with GNN to have robust phishing detection features.

GNN is generally a deep representation learning technique that is applied on graphs [13], [58]. When exploring GNN, GCN has been identified as a compelling neural network architecture that operates on graphs [59]. As an improvement to the GCN architecture, a new filtering mechanism over the traditional polynomial filters called Auto-Regressive Moving Average (ARMA) filters has also been proposed [58]. This study has shown that ARMA achieves higher mean accuracy and lower standard deviation than the traditional GCN approach in graph classification [58]. Furthermore, this study has also proposed Graph Convolutional Skip (GCS) layer with ARMA filters for better performance of the GCN architecture.

Even though literature has emphasised the advantages of GNN in HTML analysis, none of the explored solutions has used the powerful analysis and processing capabilities of GNN to find a more differentiated phishing webpage detection method. PhishDet proposed in this paper is designed to use the effective LRCN for URL-based feature extraction and GCN for effective HTML content analysis for more robust phishing detection.

III. PROPOSED MODEL: PhishDet

PhishDet uses URL and HTML content in phishing detection without experts' knowledge. It uses two separate deep networks to handle these two components, and the networks are concatenated later to produce a final decision.

Phishing detection is a binary classification task that contains two classes: legitimate and phishing. Suppose a dataset has S amount of data where each data consists of three parts: website URL (u), HTML content (w) and the label (y). A data item can be represented as u_i, w_i, y_i, where i indicates the index of the data item. Then, $y_i \epsilon \{0, 1\}$; $y_i = 1$ corresponds to a phishing website, and $y_i = 0$ represents a legitimate website.

As shown in Fig. 1, PhishDet combines two deep learning architectures called LRCN and GCN. The LRCN processes URLs and the GCN processes the HTML contents. These two models are named URLDet and HTMLDet, respectively.

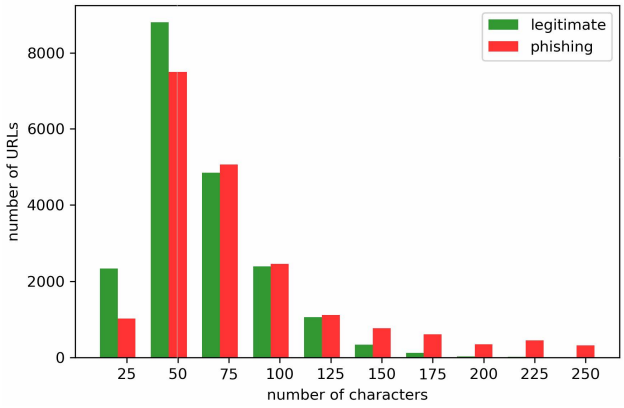


FIGURE 2. Character length distribution of the URLs.

A. URLDet

As identified by literature, the deep network architecture called LRCN is the most effective technique for designing the URLDet model. The primary input to the URLDet is the URL of the website. At first, each character of a URL is considered a word. Then, these words are transformed into a machine-understandable format using a tokeniser that allows one to vectorise a corpus by turning each word into an integer sequence. However, the input layer of the URLDet is a tensor, and the tensor must have the same shape throughout the training process. Therefore, to avoid different URL sizes that make different input shapes, all the transformed URLs are normalised into 150 maximum character lengths. In that process, the URLs with more than 150 characters in length are chopped at 150th character, and the URLs with less than 150 are filled with 0s to have 150 lengths. However, the maximum character length was decided after analysing the URL character length distribution of Dataset A (*see* Section IV-A), as shown in Fig. 2.

Once the URLs are normalised, they are first passed to the input layer. Then, the input layer passes those to the embedding layer to have a vector representation of the input. The URLDet's embedding layer is a 100 vocabulary-sized layer configured to have 256 dense embeddings and 150 length input sequences. It also used a 1e-5 valued L2 regulariser. After the embedding layer's task is completed, its output is input to the 1D convolutional and Maxpooling layers. The layers then collectively extract the local features from the embedding matrix. In here, the 1D convolutional layer uses a size three window and 256 output size with the rectified linear activation function (ReLU). Further, the 'he_uniform' initialiser with 'zeros' bias is used in the 1D convolutional layer, and the 1e-5 valued L2 regulariser is also applied. Once the features are extracted, those are passed to the URLDet's LSTM layers.

The LSTM layers are responsible for carrying out the representation learning task using the features extracted from the

1D convolutional and Maxpooling layers. In the URLDet, the LSTM layers have 32 output spaces and use the hyperbolic tangent (tanh) activation function. Further, these use 1e-5 valued L2 regularisers to introduce additional information to prevent overfitting. Once the first LSTM layer receives the output of the Maxpooling layer, the last moment output then inputs to the URLDet's second LSTM layer. After the second LSTM layer processes its input further, the output of the second LSTM layer is input to the URLDet's output layer, a dense layer with a sigmoid activation function. Once the output layer receives an input from the LSTM layer, it classifies the input and returns whether a given URL is phishing or not.

The URLDet is configured to use the Adam optimiser with binary cross-entropy loss function to minimise the target loss. Further, as shown in Fig. 1, the dropout strategy is applied in several places to prevent overfitting in the feature extraction process.

B. HTMLDet

The HTML content has provided some essential features when detecting phishing attacks [7], [9], [12], [13], [39], [47], [48]. Therefore, HTML content analysis is vital to detect phishing attacks, and HTMLDet is the responsible component for analysing HTML content in PhishDet. HTMLDet, a GCN-based architecture, is a novel phishing detection approach that was not applied elsewhere in the phishing domain. Since the HTMLDet is a representation learning approach, it does not use experts' knowledge when extracting features.

1) GRAPH CONSTRUCTION

In general, graph (G) is a pair of nodes (V) and edges (E) which can denote G(V, E) [60]. A GCN takes an input feature matrix (X) and graph structure (A). The input feature matrix is an N $\times$ D matrix, where N is the number of nodes, and D is the number of input features for each node [59]. Similarly, the graph structure is an N $\times$ N adjacency matrix representation [59].

Technically, the HTML content of a web page contains HTML tags called elements, and elements have been attributed to providing additional information about the element. Further, an attribute usually comes in name/value pair like name = 'value'. Generally, the HTML content has a tree structure, as illustrated in Fig. 3, and the tree structure can be used to generate a graph. In HTMLDet, the HTML tags and those tags' attributes contemplate the graph's nodes. Then, the node features become node labels and values. The HTML tags do not contain any values; those contain only the labels. However, the attributes usually have labels and values. Therefore, in HTMLDet, the input X becomes N $\times$ 2 matrices since each node has only two features: label and value.

In the HTMLDet input generation process, first, a script traverses through the HTML code to generate the node list and the features of each node. Then, a graph is constructed from the generated node, and the edges of the graph are added hierarchically based on the parent-child relationship. Concurrently, feature matrix X is also generated using the features of each node. However, the X needs a machine-understandable representation since the labels and values contain textual content. Therefore, a domain-specific doc2vec model is constructed using the Dataset A corpus, and the trained doc2vec model is used to transform the X's textual content into a size one vector format. The following HTML code example explains how the graph is constructed from a given HTML content.

```
<p>
  <img src='a.jpg' alt='example-image'/>
</p>
```

In the above example, the HTML tags, 'p' and 'img', and 'img' tag attributes 'src' and 'alt' are considered nodes. Therefore, the example has four nodes: p, img, src, and alt. These nodes get a unique identification number as 1, 2, 2_1, and 2_2, respectively. Now the edges list is generated as (1, 2), (2, 2_1), and (2, 2_2). Then a graph is constructed from these edges. Graph A (Fig. 4) shows a construction for the above example HTML code, and graph B shows the generated graph for Fig. 3 HTML code. Next, the feature matrix is also generated, and Table 3 shows the feature matrix generated for the above example.

TABLE 3. Input X values (The values are transformed to a vector format in real execution).

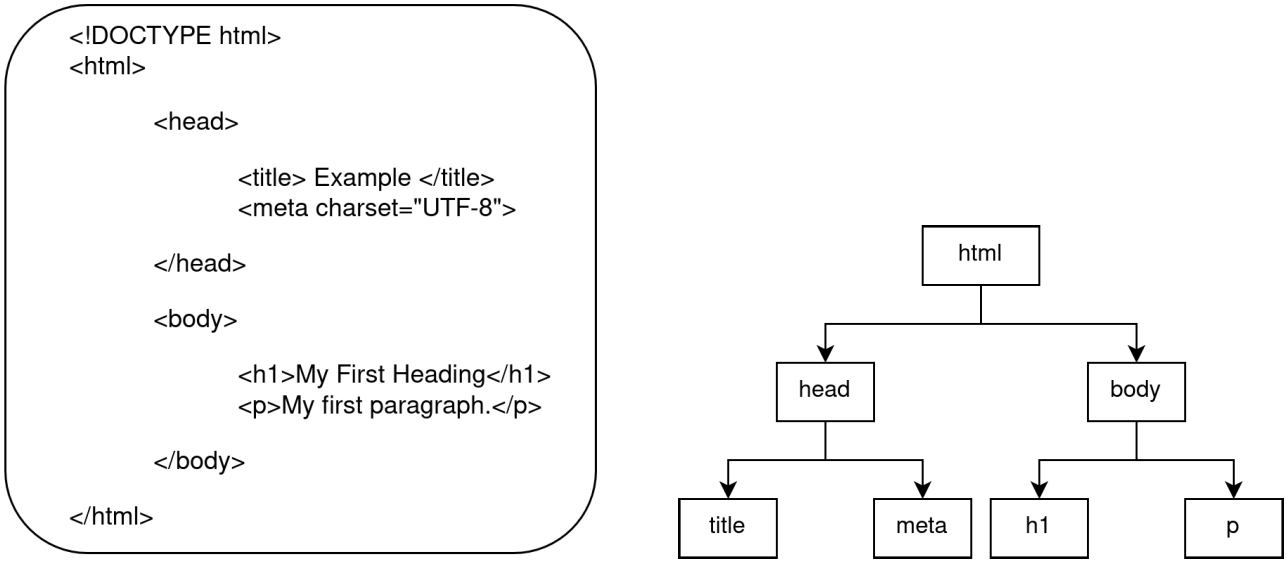
Node / Feature	Label	Value
1	p	
2	img	
2_1	src	a.jpg
2_2	alt	example_image

2) MODEL IMPLEMENTATION

The HTMLDet model, as illustrated in Fig. 1, is designed using GCN architecture. It uses an input layer and three GCS layers with pooling layers. HTMLDet requires three inputs: the adjacency matrix, the feature matrix, and segment ids. The input layer in HTMLDet has three input tensors, and one of these is a sparse tensor to accept an adjacency matrix. In the HTMLDet model, the adjacency matrix and feature matrix are first constructed using Section III-B1. The third input, segment ids, is generated using an inbuilt Spektral[2] function. Then these three are inserted into the input layer. After that, the HTMLDet passes the adjacency matrix and feature matrix to the GCS layer to carry out graph nodes' feature representation. In HTMLDet, all GCS layers consist of 32 channels and use the ReLU activation function. These layers use 1e-3 valued L2 regularisers and 'he_uniform' initialiser with 'zeros' bias.

In the next step, the GCS output inputs to the MinCutPool layer. In HTMLDet, the MinCutPool layers used the

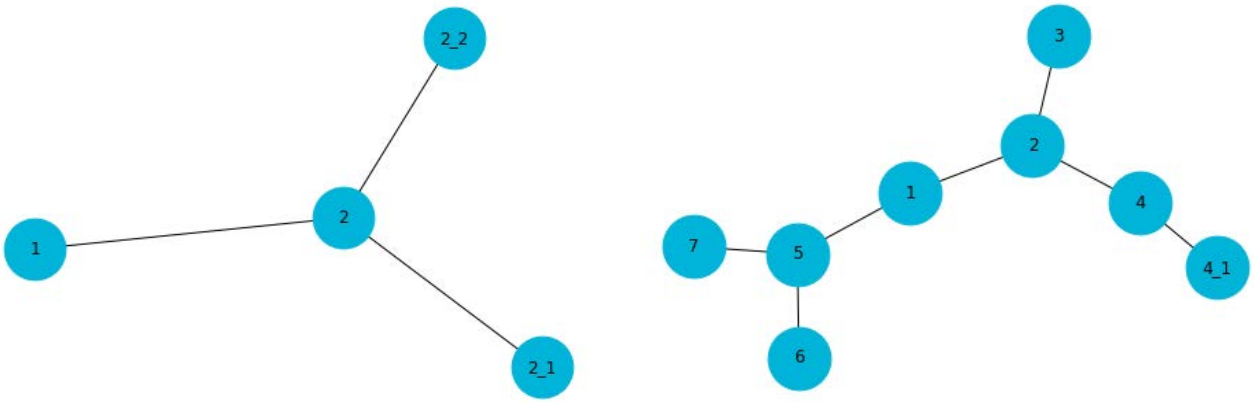
[2]A specific library available in Keras to perform graph classification tasks.



(a). Sample HTML code

(b). Tree view of the given HTML code

FIGURE 3. Example of an HTML page in a tree view.



(a). Graph A

(b). Graph B

FIGURE 4. Example graphs constructed from the graph construction process.

ReLU activation function, 1e-3 valued L2 regularizer and 'he_uniform' initializer with 'zeros' bias. The k value of the MinCutPool layer is decided using the number of average nodes in the training dataset. The MinCutPool layer needs segment ids to construct the aggregated features. Therefore, the input layer links the initial segment ids to the first MinCutPool layer for feature construction. Then, the aggregated features are inputs to the second GCS layer for more abstract feature representation. The GCS and MinCutPool process the node features similar to the previous layers by outputting the relevant feature matrix, adjacency matrix, and segment ids. The second level feature matrix again inputs to the third GCS layer for a deeper feature representation. The third GCS layer's output inputs to a GlobalAvgPool layer to generate one feature map for the classification task. Finally, to classify the HTML page, the GlobalAvgPool layer's output inputs to the HTMLDet's output layer, a dense layer with a softmax activation function. The HTMLDet has also used adam optimiser and categorical cross-entropy loss function to adjust the network's weights.

C. PhishDet

As visualised in Fig. 5, the URLDet and HTMLDet networks are concatenated to have the proposed anti-phishing solution, PhishDet. The two networks, URLDet and HTMLDet, are separately trained and combined through a concatenation layer by eliminating the output layers of each. Then a dense layer with two neurons is added as the output layer of PhishDet, and the softmax activation is enabled to have a phishing and legitimate probability for a given input. Further, PhishDet has used adam optimiser with

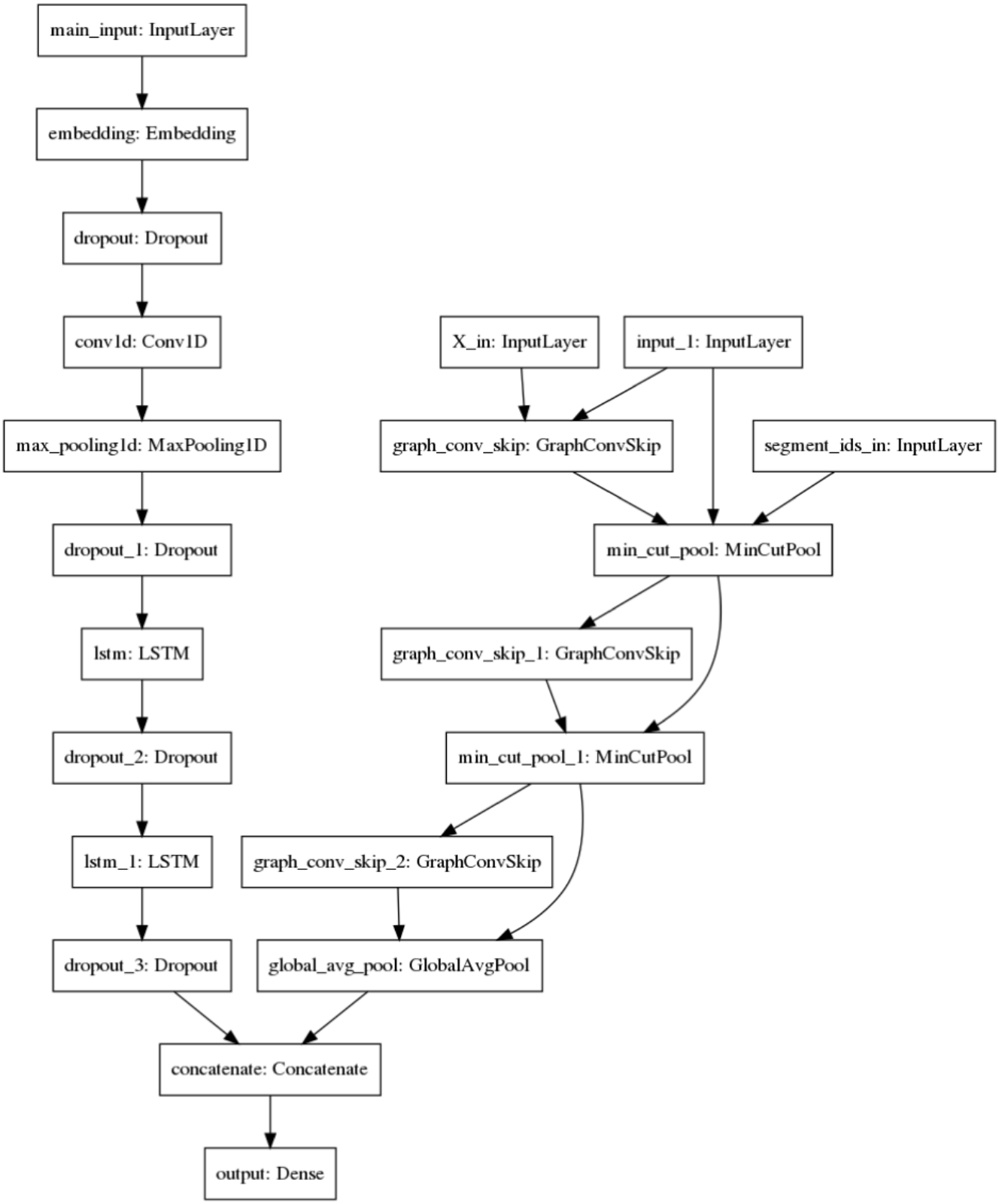


FIGURE 5. A plot of PhishDet model graph.

a categorical cross-entropy loss function to have the least difference between actual and predicted outputs.

IV. EXPERIMENT

The proposed solution, which combines LRCN and GCN deep networks, was implemented with Keras 2.3.1 and Spektral 0.3.0 deep learning libraries. The experiment environment depended on TensorFlow 2.1.0 deep learning framework and Python 3.7.6 programming language.

A. DATASETS

Generally, phishing websites exist only for a limited time on the web [53]. Therefore, it is not easy to construct more reliable datasets for a study if the study depends on URL and HTML content like the current study. However, in previous studies, PhishTank, OpenPhish and APWG were mainly used when constructing phishing datasets, and legitimates were generated mainly through Alexa [53]. Although these sources exist to construct a dataset for a study, a diverse and extensive dataset is the key to accurate and unbiased detection [38].

TABLE 4. Details of the used datasets.

Dataset	Sub-dataset	Legitimate	Phishing	**Total**
Dataset A [40,000]	Training Set (Tr_A)	13,895	14,105	**28,000**
	Validation Set (Val_A)	2,006	1,994	**4,000**
	Testing Set (Te_A)	4,099	3,901	**8,000**
Dataset B [50,000]	Training Set (Tr_B)	17,500	17,500	**35,000**
	Validation Set (Val_B)	2,500	2,500	**5,000**
	Testing Set (Te_B)	5,000	5,000	**10,000**
Benchmark Dataset [46,096]	Training Set (Tr_{Bm})	17,360	14,906	**32,266**
	Validation Set (Val_{Bm})	2,480	2,130	**4,610**
	Testing Set (Te_{Bm})	4,960	4,260	**9,220**

The current study used three datasets during the experiment time. Out of these, two are publically available datasets, and one is limited to this study. For ease of use, the study named these datasets Dataset A, Dataset B and the Benchmark Dataset. These three datasets were used on different scales during the experiment to pursue training, testing and validation, and Table 4 presents the details of those datasets in each task.

Dataset A contained 20,000 phishing data and 20,000 legitimate data. Both legitimate and phishing data contained a URL and the relevant web page. The legitimate data was collected from the Google search engine. A script was used to pass a word list to the Google search engine, and the websites that appeared within the top 10 search results were selected as legitimate websites. The word list consisted of words borrowed from the Internet and self-generated words.

The phishing data was mainly constructed using PhishTank records submitted before September 2019 and a public phishing websites dataset [61]. Even though the public phishing dataset is a trusted source, these data were again verified using PhishTank and GSB API to guarantee the label. After constructing the Dataset A, 40,000 data items included there were divided randomly into three sub-datasets: training, validation, and testing. 70% of data was used for the training, and 10% and 20% were used for validation and testing. The amount of legitimate and phishing data available in these sub-datasets is mentioned in Table 4.

Dataset B, which contained 50,000 data items consists of an equal amount of legitimates and phishing which was constructed from a public phishing website dataset [62]. Then, the dataset was mainly split into two sub-datasets as training and testing in a 4:1 ratio, and 10% of training data were used for the validation task during the model training. Since the experiment used a balanced training environment, an equal number of legitimates and phishing data were used under all sub-datasets, as displayed in Table 4.

Benchmark Dataset is again a publically available dataset.[3] It is used to build a similar solution to the proposed one [13], but with a different technique. Since a few data had some issues during the extraction, the study could select 24,800 legitimate and 21,296 phishing data from the original dataset to construct Benchmark Dataset. Then, these data were spitted into training, validation and testing, as shown in Table 4.

Out of these three datasets, Dataset B and Benchmark Dataset were initially selected for the final evaluation process since Dataset A contained old phishing data used in initial level training. In an anti-phishing study, a diverse dataset is essential for a more generalised model [38]. Therefore, the diversity of Dataset B and Benchmark Dataset was evaluated via a specific analysis. Since there is no widely accepted method to check the diversity of a phishing dataset [38], the number of different domains and the number of different top-level domains (TLDs) used in the past [38] were used alongside the URL length and HTTPS presence during the analysis. The URL length and HTTPS presence were especially considered because earlier findings have shown that HTTPS in phishing attacks and URL length distribution are essential to consider in the current phishing attack nature to have unbiased, accurate model training [6], [51].

Fig. 6 to Fig. 9 show the results obtained through the diversity analysis. In there, the domains and TLDs distribution analysis followed a specific procedure. First, a list of unique domains and TLDs was separately collected from each dataset, and those frequencies were calculated. Then, the top fifty domains and TLDs were selected and the percentage of those proportion to the size of the relevant dataset was calculated accordingly. Finally, the percentages were plotted to have domains and TLDs distribution, as shown in Fig. 6 and Fig. 7. This domain and TLDs analysis show that both datasets were not biased toward specific domains or TLDs.

However, the URL character length and the presence of HTTPS analysis illustrated a different view of these two datasets. It has shown that the Benchmark Dataset is not diverse based on URL character length and HTTPS presence. Fig. 8 shows that the legitimate URL character lengths are positively skewed, and Fig. 9 shows that most URLs in the Benchmark Dataset are in the HTTP category. It concludes that a model trained with the Benchmark Dataset may result

[3]https://github.com/Hanjingzhou/Web2vec

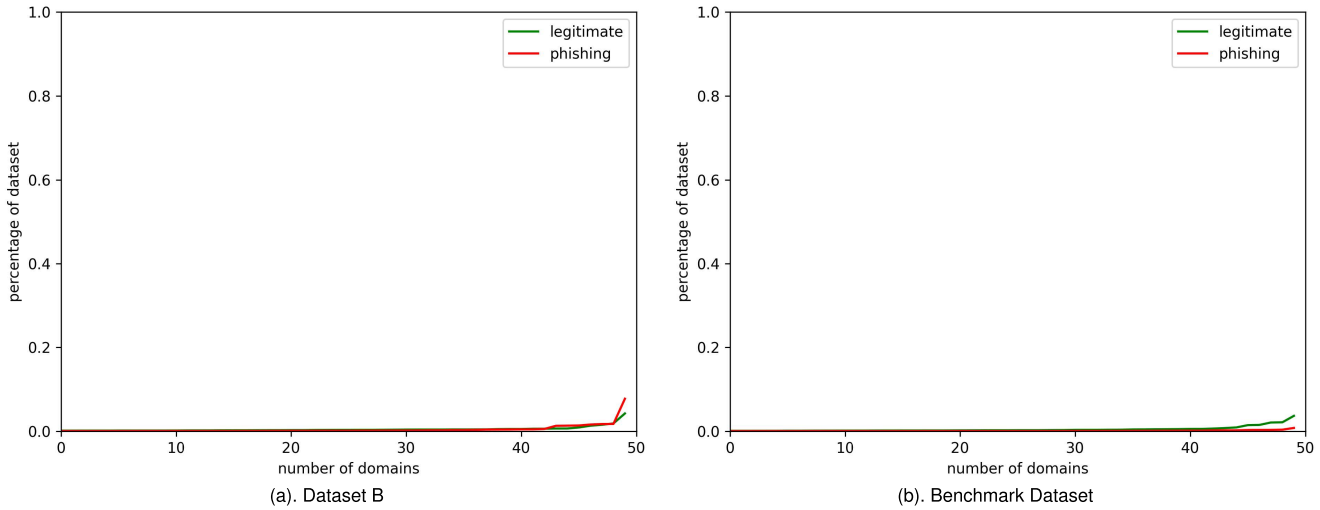


FIGURE 6. Distribution of domains in Dataset B and benchmark dataset.

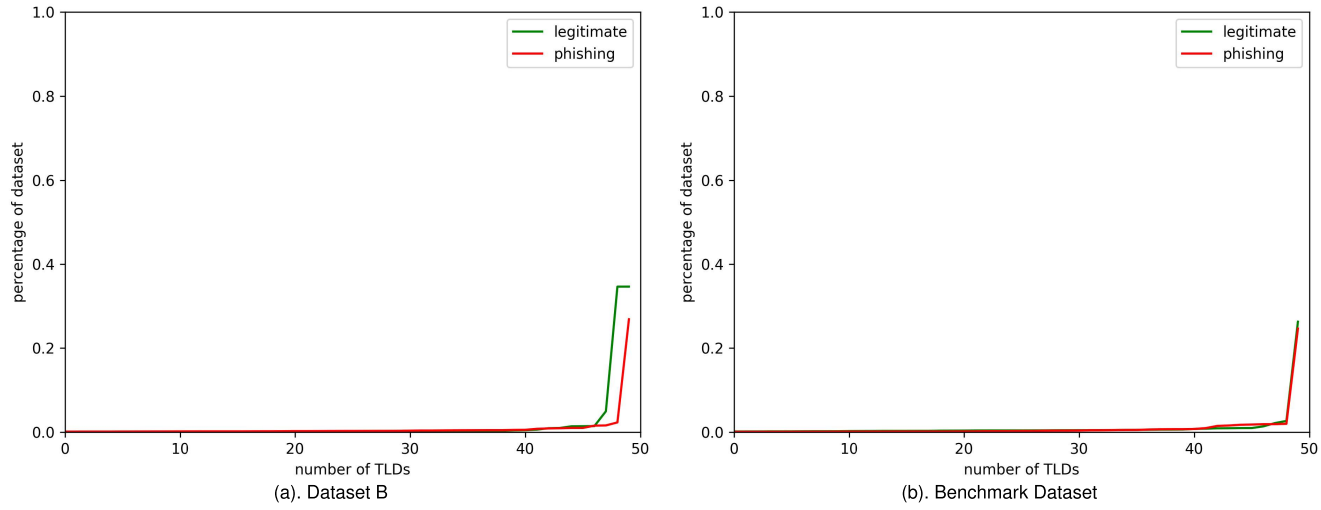


FIGURE 7. Distribution of TLDs in Dataset B and benchmark dataset.

in a biased model and not presents the current phishing status [38], [51]. Therefore, a more realistic evaluation of the proposed solution was feasible with Dataset B compared to the Benchmark Dataset. Thus, Dataset B was selected to evaluate the proposed solution's overall performance, and Benchmark Dataset has been used only to benchmark the solution with other similar solutions.

B. PERFORMANCE METRICS

The confusion matrix is an improved approach to evaluate a classification model's correctness and errors. More importantly, the confusion matrix gives a better insight into the types of errors the classifier is being made during the classification process. Therefore, when evaluating the performance of the PhishDet, a confusion matrix was used, as shown in Fig. 10.

In this experiment, detecting a phishing website is considered a positive case. Therefore, in the confusion matrix, 'True Positive' (TP) is used for correctly predicted phishing sites, 'False Positive' (FP) for incorrectly predicted phishing sites, 'True Negative' (TN) for correctly predicted legitimate sites, and 'False Negative' (FN) for incorrectly predicted legitimate sites. Although the confusion matrix provides an understanding of the model performance, advanced classification metrics like accuracy, precision, recall, and f1 score were used during the experiment to understand the model performance in a better manner.

C. TRAINING

PhishDet was trained in two phases. Initially, all the models, URLDet, HTMLDet and PhishDet, were trained using Dataset A. Then Dataset B was used to retrain PhishDet to incorporate the latest phishing knowledge.

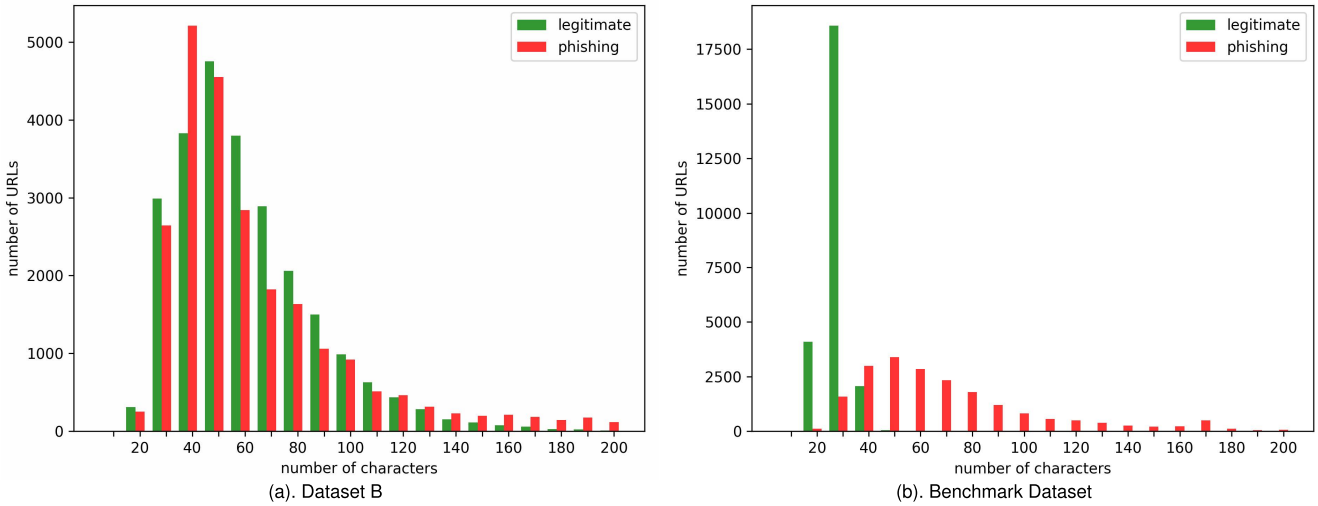


FIGURE 8. Distribution of URL length in Dataset B and benchmark dataset.

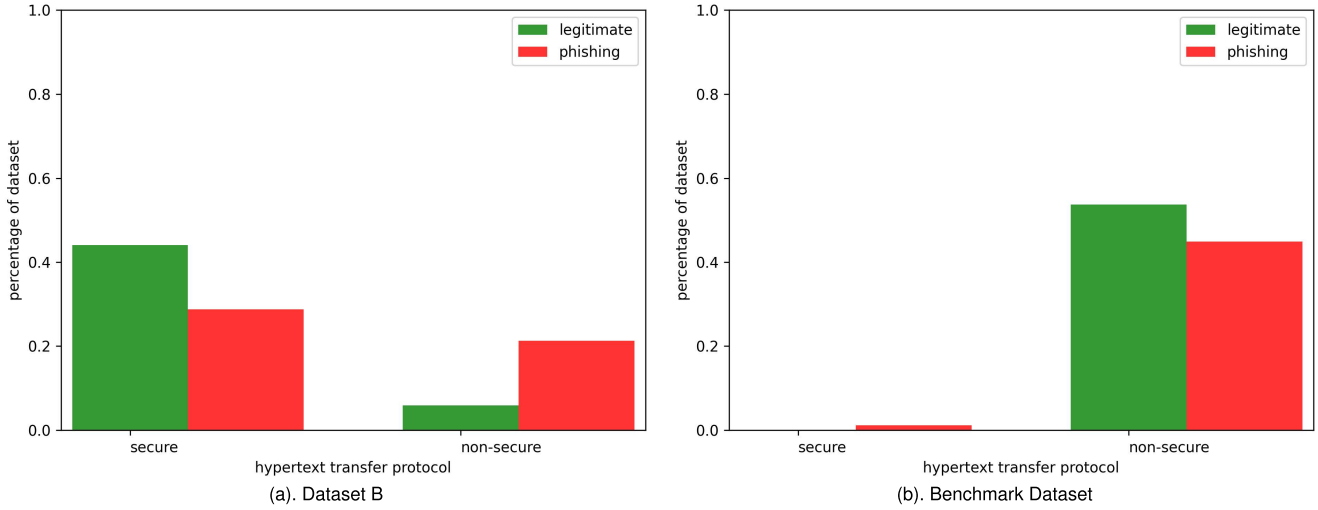


FIGURE 9. Distribution of HTTPS in Dataset B and benchmark dataset.

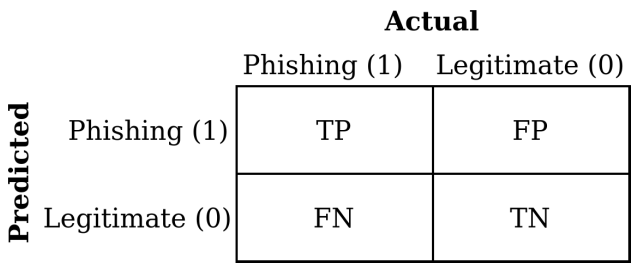


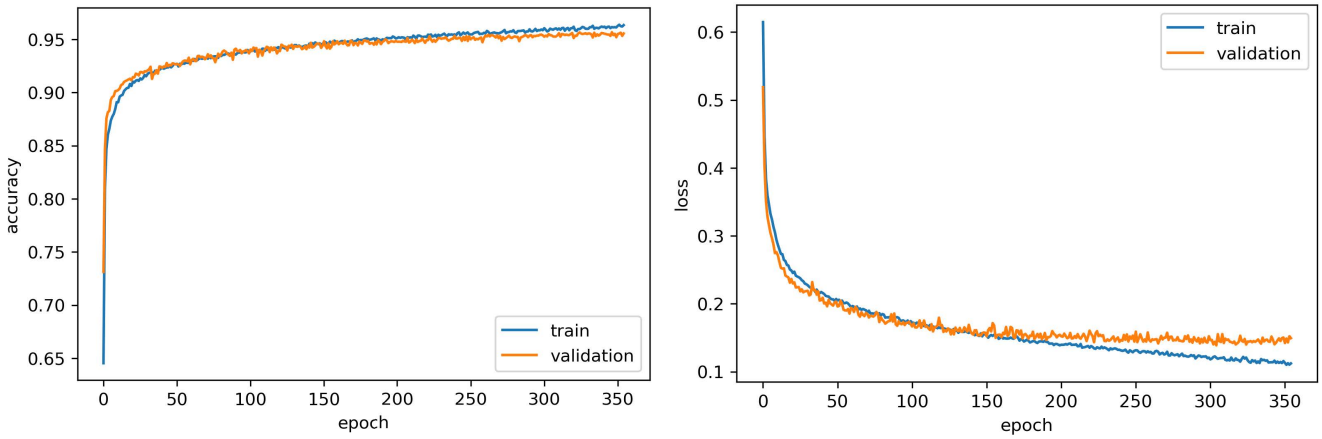
FIGURE 10. Confusion matrix.

1) TRAINING WITH DATASET A

The training process was executed in three steps. First, the URLDet model was trained with 64 batch sizes and a learning rate of 1e-4 for 500 epochs. Since the early stopping technique was used, the training process was stopped at the 355th epoch. Fig. 11 shows the accuracy and loss functions' performance of URLDet for both training and validation datasets. Next, the HTMLDet model was trained using a batch size of one (size one is a requirement in ARMA filters) and a learning rate of 1e-3. The training process was stopped in the 272nd epoch due to the early stopping technique (Fig. 12). Finally, PhishDet was trained with a batch size of one and a learning rate of 1e-5 and the training was stopped at the 79th epoch. Fig. 13 illustrates the phase one training's accuracy and loss functions.

2) TRAINING WITH DATASET B

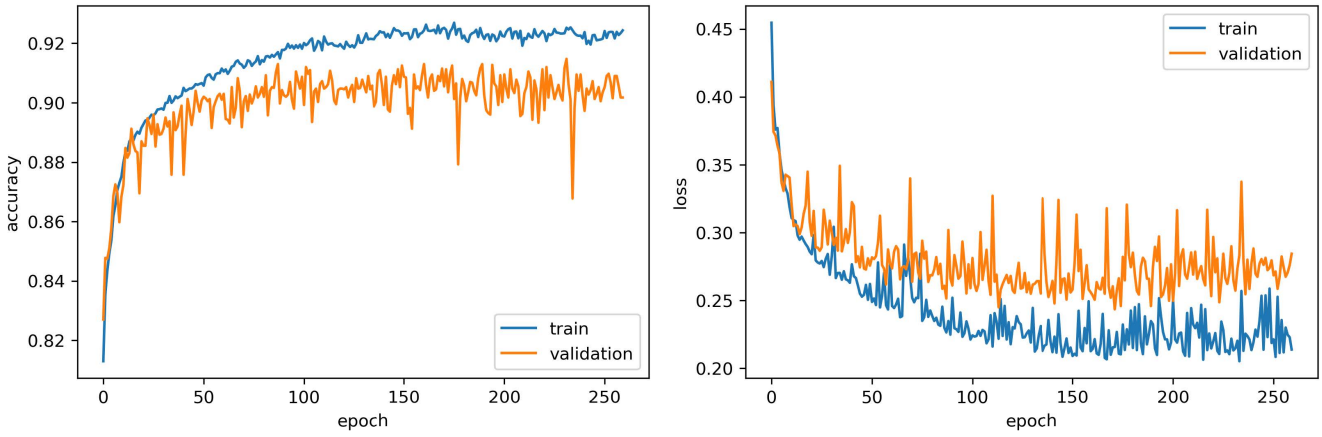
After the initial training, the PhishDet was retrained with Dataset B to explore the latest phishing trends. The retraining used the same batch size and learning rate in phase one and completed training within 292 epochs. Fig. 14 shows the accuracy and loss functions of the phase two training.



(a). Accuracy curve for train and validation set

(b). Loss curve for train and validation set

FIGURE 11. URLDet performance curves.



(a). Accuracy curve for train and validation set

b). Loss curve for train and validation set

FIGURE 12. HTMLDet performance curves.

D. EVALUATION

PhishDet was evaluated under five criteria to check the proposed solution's effectiveness.

1) OVERALL PERFORMANCE

After the initial training of the PhishDet, the solution was evaluated on Te_A and Te_B datasets. Once the phase two training concluded, the model was re-evaluated with the Te_B dataset. Table 5 shows the results obtained during these evaluations.

TABLE 5. PhishDet performance evaluation.

Dataset	Phase	Accuracy	Precision	Recall	F1
Te_A	Phase 1	96.64%	97.59%	95.80%	96.69%
Te_B	Phase 1	87.29%	86.35%	88.58%	87.45%
	Phase 2	**96.42%**	**96.40%**	**96.44%**	**96.42%**

2) BENCHMARKING

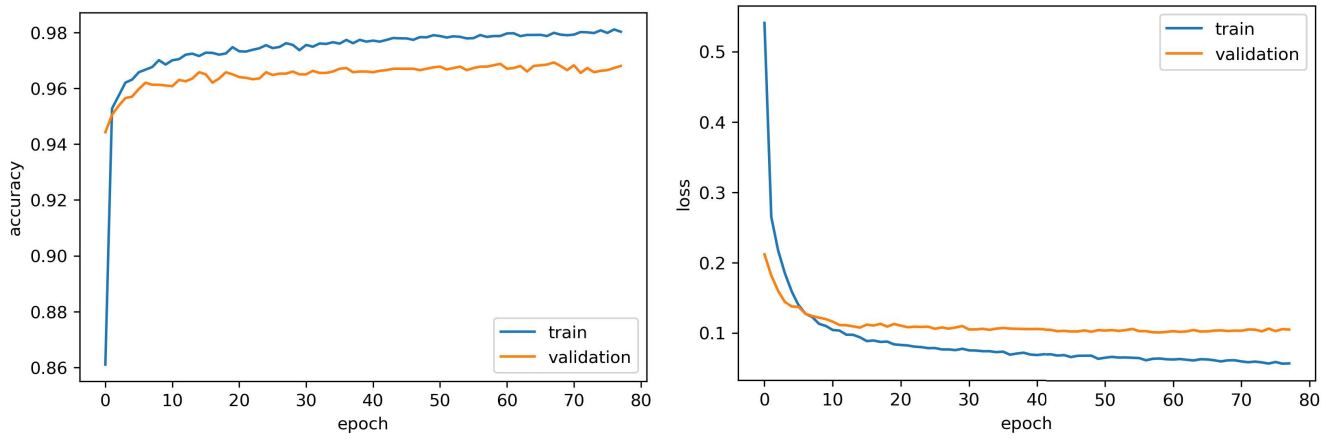
PhishDet was benchmarked in two different experiments. First, it was used with five benchmark solutions: URLDet, HTMLDet, URLNet[4] [63], StackModel[5] [7] and HybridDLM[6] [52], to compare the performance. Secondly, PhishDet was evaluated using the Benchmark Dataset.

In the first experiment, all the benchmark solutions were initially trained with Tr_B dataset. Then, the performance was evaluated using the Te_B dataset. Table 6 shows the obtained results of the first experiment. The second experiment trained the phase 1 model on the Tr_{Bm} dataset and evaluated the before and after model performance using the Te_{Bm}

[4]https://github.com/Antimalweb/URLNet

[5]The study could not found the original implementation of the StackModel. Therefore, the StackModel implemented by [64] was used.

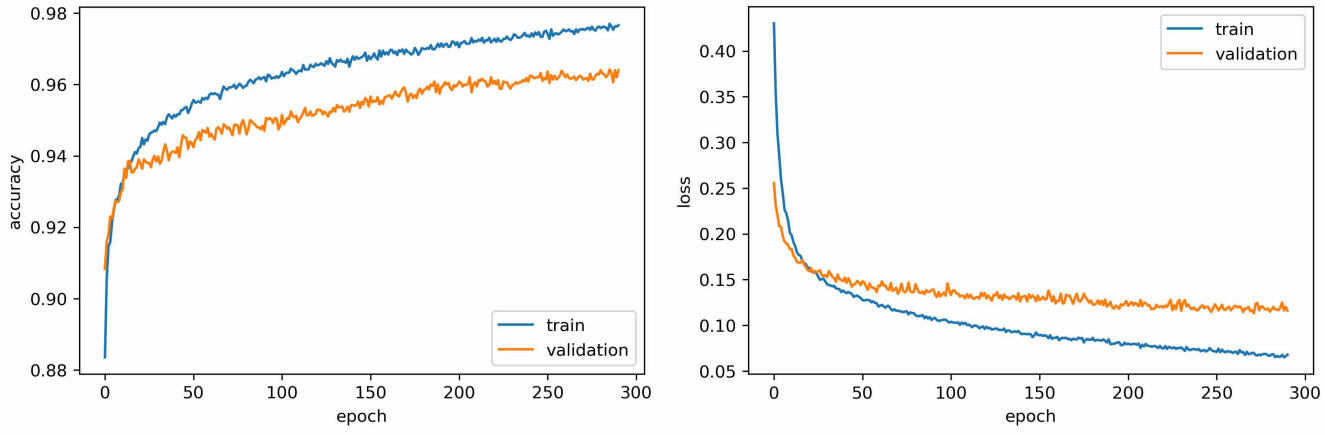
[6]https://github.com/sna-hm/HybridDLM



(a). Accuracy curve for train and validation set

(b). Loss curve for train and validation set

FIGURE 13. PhishDet performance curves in phase 1 training.



(a). Accuracy curve for train and validation set

(b). Loss curve for train and validation set

FIGURE 14. PhishDet performance curves in phase 2 training.

dataset. Table 7 presents the results obtained by the second experiment.

3) ZERO-DAY ATTACK DETECTION

A zero-day attack, essentially a new or not yet reported attack detection, is vital in any anti-phishing solution. Therefore, PhishDet was also evaluated against zero-day attacks through a specific testing procedure exercised in previous studies [65], [66] with the support of a famous blacklist, GSB.

During the experiment, the PhishTank was continuously monitored through an automated script. The script was responsible for calling PhishTank API every hour and collecting verified phishing URLs submitted in the last hour. Then the collected URLs were submitted to PhishDet and GSB API. The results provided by these two solutions were recorded separately. After a while, the second attempt was carried out on the URLs that were not marked as phishing by Google during the first attempt. However, the study recorded a more than 24 hours' delay between the first and second attempts since 47%-83% of phishing URLs are added to the blacklists after 12 hours [26]. Then, if a submitted URL did not receive a phishing flag from Google in the first attempt and received it in the second attempt, it was considered a zero-day attack. Finally, the correct decisions made by PhishDet over the zero-day attacks were calculated to get a conclusion about PhishDet's zero-day attack detection ability. The experiment was done for three days, and the results are tabulated in Table 8.

4) DETECTION TIME

The experiment was set up on an Intel(R) Xeon(R) CPU @ 2.20GHz machine with four cores and 32 GB of memory. The average detection time for a given website was calculated using 3,000 randomly selected data from the Te_B dataset. Fig. 15 shows the PhishDet detection time over each website.

TABLE 6. Comparison with selected phishing detection solutions.

Model	Feature Set(s)	Accuracy	Precision	Recall	F1
URLDet	URL Only	94.81%	95.57%	93.98%	94.77%
HTMLDet	HTML Only	89.87%	91.18%	88.28%	89.71%
URLNet	URL Only	91.71%	92.62%	90.64%	91.62%
StackModel	URL* + HTML*	93.83%	93.18%	94.58%	93.87%
HybridDLM	URL + HTML*	96.95%	96.98%	96.28%	96.51%
PhishDet	URL + HTML	96.42%	96.40%	96.44%	96.42%

*Manually extracted features

TABLE 7. PhishDet performance evaluation with the benchmark dataset.

	Accuracy	Precision	Recall	F1
Before Train with Tr_{Bm}	86.65%	94.58%	75.42%	83.92%
After Train with Tr_{Bm}	99.57%	99.41%	99.65%	99.53%

TABLE 8. Details of the zero-day attack detection experiment.

1st Attempt Date*	2nd Attempt Date	f_1	f_2	f_3	f_4	f_5
20-11-2021	22-11-2021	53	49	16	04	04
21-11-2021	23-11-2021	217	188	101	18	17
22-11-2021	24-11-2021	135	128	85	06	06

f_1 - Number of phishing websites downloaded from PhishTank
f_2 - Number of phishing websites correctly detected by PhishDet
f_3 - Number of phishing websites correctly detected by GSB
f_4 - Number of zero-day attacks (*see* Section IV-D3)
f_5 - Number of zero-day attacks correctly detected by PhishDet

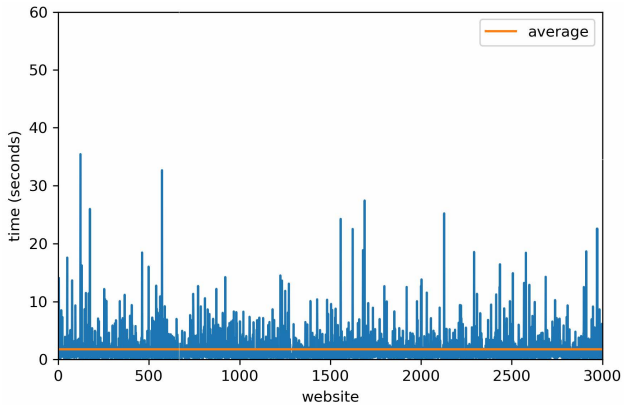


FIGURE 15. PhishDet detection time curve.

5) PERFORMANCE ON AN IMBALANCED DATASET

On the real Internet, the occurrence of phishing is much lower compared to the legitimate. Therefore, evaluating an anti-phishing tool on an imbalanced dataset is essential for a realistic evaluation of the model's performance [38]. To have such an experiment, the current study constructed imbalanced datasets by having different legitimate to phishing ratios from 1:1 to 1:10. However, the number of legitimate counts was kept constant in all cases, and different ratios were maintained by changing the number of phishing data. The experiment was performed twice using the Te_B and Te_{Bm} datasets to have an unbiased, accurate conclusion at the end. The f1-score was used to measure the performance instead of accuracy since it is the most appropriate metric on imbalanced datasets [38].

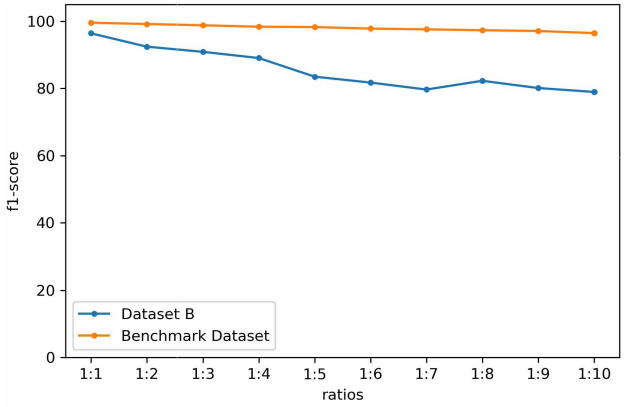


FIGURE 16. PhishDet performance on different legitimate to phishing ratios.

Fig. 16 demonstrates the model's performance on the different legitimate to phishing ratios.

V. RESULTS AND DISCUSSION

Phishing attacks are at historic highs at present. It is then necessary to experiment solutions to this increasing phishing trend. In that context, PhishDet comes as an alternative solution that uses LRCN with GCN to fight against phishers. PhishDet has experimentally shown that it can detect phishing attacks with 96.42% accuracy, 96.40% precision, 96.44% recall, and 96.42% f1-score. The FP and FN rates were also at 0.036.

Moreover, the proposed solution was compared with five different solutions in Section IV-D2. In that experiment, the URLDet and URLNet are based only on URLs, and the HTMLDet is based only on HTML content. However, StackModel and HybridDLM use both URL and HTML content during the detection, and StackModel extracts both URL and HTML features manually, while HybridDLM only collects HTML features manually. Table 6 result shows that PhishDet performed well over URLDet, URLNet, HTMLDet and StackModel. However, HybridDLM recorded a marginally high performance over PhishDet. It could be due to the careful selection of phishing detection features in HybridDLM because of the feature extraction technique. However, as mentioned in Section II-A, manual feature extraction has several drawbacks. PhishDet is free from such drawbacks due to the used representation learning approach. It could also gradually learn the significant features from the raw URLs and HTML contents when the phishing detection features

are constantly changing. Therefore, PhishDet is better than HybridDLM from the perspective of feature engineering.

According to Table 2, the Web2Vec solution achieved the highest phishing detection accuracy among recent solutions, and it is 99%. As discussed in Section II-B1, Web2Vec and WebPhish solutions are similar to PhishDet. Further, Web2Vec and WebPhish have used a similar approach when collecting data for the study. Therefore, PhishDet experimented with Web2Vec's dataset in Section IV-D2 to benchmark the solution with similar solutions. PhishDet recorded 99.53% detection accuracy in that experiment, which is better than Web2Vec recorded accuracy. However, as discussed in Section IV-A, the Benchmark Dataset was not diverse. Therefore, the accuracy recorded with the Benchmark Dataset was not selected as the actual accuracy of PhishDet. However, it shows how a dataset could mislead the final accuracy of a phishing detection solution. Again, it is proof of what [38] highlighted about the importance of a diverse dataset in a phishing detection study.

Further, the literature has suggested that if Alexa is used to collect legitimate URLs, specific strategies should be applied to have a diverse set of legitimate URLs [38]. Since Benchmark Dataset failed to apply such strategies, PhishDet shows an unrealistic performance with it. Moreover, the latest is better when collecting phishing data from any public source since phishing detection features are constantly changing. Otherwise, the collected dataset may not represent the current phishing status, and then the trained solution may not be effective in a real attacking environment.

Although PhishDet performed well in a balanced environment, Fig. 16 shows that the proposed model's performance declines due to its imbalanced nature. The f1-score was downgraded from 96.42% to 78.95% when the phishing to legitimate ratio changed from 1:1 to 1:10 in the Te_B dataset, and it is proportionally a 17.47% decline. Further, the FN rate had increased from 0.036 to 0.066 during that experiment. However, the experiment done with the Te_{Bm} dataset had only a 3.09% drop in the f1-score when the ratios have changed similarly. It again emphasises the effect of diversity in datasets in phishing detection studies and how the performances of the studies are misled based on the used dataset. However, there is no well-accepted legitimate phishing ratio to test the imbalanced nature thoroughly, and it is also known that the classifier performance declines when the dataset becomes more and more imbalanced [38]. In such context, PhishDet averagely maintained an 85.51% of f1-score with the Te_B dataset and 98.05% of f1-score with the Te_{Bm} dataset in the presented experiment.

Moreover, the detection time and zero-day attack detection are crucial for any anti-phishing solution to be effective in a real environment. PhishDet has gained positive points for these criteria during the performed experiments. Fig. 15 shows that the average detection time of PhishDet is 1.8 seconds. It is a notable achievement by PhishDet because [44] have tried different ways to decrease the detection time in their study and finally came to an average value of 3.5 seconds. The zero-day experiment (Section IV-D3) also shows that PhishDet could detect 27 out of 28 zero-day attacks, which is 96.43%. Further, the solution's overall accuracy was 90.12% during the experiment period, and it is well above the favourite blacklist, GSB, which detected phishing attacks with 49.88% accuracy.

However, the current performance of PhishDet could be decreased over time. The problems like data drifting could affect the performance of PhishDet in the long run since the phishing nature is rapidly changing, and new attacking strategies have evolved [42]. Thus, PhishDet must be retrained occasionally to become more effective against future attacks [38]. Section IV-D1 experimentally shows that PhishDet's detection accuracy declined by 9.35% during one year since Dataset A was collected before September 2019 and Dataset B was after November 2020. However, after retraining the model with Dataset B, PhishDet reclaimed the accuracy and proved the effectiveness of retraining in the long run. Although PhishDet requires retraining, it will be less costly because PhishDet uses a representation learning approach without expert involvement. However, retraining requires labelled data and acquiring labelled data in the phishing domain is one of the challenges when maintaining the performance of anti-phishing solutions in the long run [42]. That challenge applies to the proposed solution, and standard data collection approaches need to be established in future to support PhishDet kind of solutions to retain in the anti-phishing domain for a more extended periods.

VI. CONCLUSION AND FUTURE WORK

This paper presents a differentiated phishing detection approach called PhishDet that exercises representation learning techniques simultaneously for URL and HTML content of a web page. It combines two separate models named URLDet and HTMLDet that were modelled using LRCN and GCN techniques to process URLs and HTML contents. PhishDet performs well at present. However, the solution should be retained occasionally to be more effective in future attacks, and the retaining process may not be costly due to the advantages of the representation learning technique.

PhishDet is susceptible to adversarial attacks since the current version of PhishDet is not trained against them. Therefore, a GAN is planned to integrate with PhishDet to minimise adversarial attack effects in future work. Further, an automated framework to retrain the model through a labelled data collection process may add some value to PhishDet to overcome labelled data issues in the long run. Moreover, PhishDet is planned to integrate with an active continuous learning environment to have a dynamic approach against the prevalent identity theft called phishing.

ACKNOWLEDGMENT

The authors acknowledge the support received from the LK Domain Registry in publishing this paper and the invaluable remarks from Dr. Chamath Keppitiyagama of the University of Colombo School of Computing, Sri Lanka.

REFERENCES

[1] K. L. Chiew, K. S. C. Yong, and C. L. Tan, "A survey of phishing attacks: Their types, vectors and technical approaches," *Expert Syst. Appl.*, vol. 106, pp. 1–20, Sep. 2018.

[2] Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha, and M. Guizani, "Systematization of knowledge (SoK): A systematic review of software-based web phishing detection," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2797–2819, 4th Quart., 2017.

[3] W. D. Yu, S. Nargundkar, and N. Tiruthani, "A phishing vulnerability analysis of web based systems," in *Proc. IEEE Symp. Comput. Commun.*, Jul. 2008, pp. 326–331.

[4] A. Sfakianakis, C. Douligeris, L. Marinos, M. Lourenço, and A. O. Raghimi, "ENISA threat landscape report 2018: 15 top cyberthreats and trends," Eur. Union Agency Netw. Inf. Secur. (ENISA), Greece, Tech. Rep., 2019, doi: 10.2824/622757.

[5] A. K. Jain and B. B. Gupta, "Phishing detection: Analysis of visual similarity based approaches," *Secur. Commun. Netw.*, vol. 2017, pp. 1–20, Jan. 2017.

[6] *Phishing Activity Trends Report: 4th Quarter 2020*, Anti-Phishing Working Group, Lexington, MA, USA, Feb. 14, 2021.

[7] Y. Li, Z. Yang, X. Chen, H. Yuan, and W. Liu, "A stacking model using URL and HTML features for phishing webpage detection," *Future Gener. Comput. Syst.*, vol. 94, pp. 27–39, May 2019.

[8] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[9] C. Opara, B. Wei, and Y. Chen, "HTMLPhish: Enabling phishing web page detection by applying deep learning techniques on HTML analysis," 2019, *arXiv:1909.01135*.

[10] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. Gonzalez, "Classifying phishing URLs using recurrent neural networks," in *Proc. APWG Symp. Electron. Crime Res. (eCrime)*, Apr. 2017, pp. 1–8.

[11] W. Chen, W. Zhang, and Y. Su, "Phishing detection research based on LSTM recurrent neural network," in *Proc. Int. Conf. Pioneering Comput. Scientists, Eng. Educators*. Singapore: Springer, 2018, pp. 638–645.

[12] C. Opara and Y. Chen, "Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics, 2020, *arXiv:2011.04412*.

[13] J. Feng, L. Zou, O. Ye, and J. Han, "Web2Vec: Phishing webpage detection method based on multidimensional features driven by deep learning," *IEEE Access*, vol. 8, pp. 221214–221224, 2020.

[14] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, Dec. 2015.

[15] W. Wang, F. Zhang, X. Luo, and S. Zhang, "PDRCNN: Precise phishing detection with recurrent convolutional neural networks," *Secur. Commun. Netw.*, vol. 2019, pp. 1–15, Oct. 2019.

[16] M. Sameen, K. Han, and S. O. Hwang, "PhishHaven—An efficient real-time AI phishing URLs detection system," *IEEE Access*, vol. 8, pp. 83425–83443, 2020.

[17] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing attacks: A recent comprehensive study and a new anatomy," *Frontiers Comput. Sci.*, vol. 3, Mar. 2021, Art. no. 563060.

[18] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in *Proc. NDSS*, 2010, pp. 1–14.

[19] E.-S. M. El-Alfy, "Detection of phishing websites based on probabilistic neural networks and K-medoids clustering," *Comput. J.*, vol. 60, no. 12, pp. 1745–1759, Apr. 2017.

[20] M. Chatterjee and A. S. Namin, "Deep reinforcement learning for detecting malicious websites," 2019, *arXiv:1905.09207*.

[21] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, 2018.

[22] S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge, "Anti-phishing phil: The design and evaluation of a game that teaches people not to fall for phish," in *Proc. 3rd Symp. Usable Privacy Secur.*, 2007, pp. 88–99.

[23] M. Baslyman and S. Chiasson, "'Smells phishy?': An educational game about online phishing scams," in *Proc. APWG Symp. Electron. Crime Res. (eCrime)*, Jun. 2016, pp. 1–11.

[24] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "PhishNet: Predictive blacklisting to detect phishing attacks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–5.

[25] Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list," in *Proc. 4th ACM Workshop Digit. Identity Manage.*, 2008, pp. 51–60.

[26] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091–2121, 4th Quart, 2013.

[27] A. K. Jain and B. B. Gupta, "A novel approach to protect against phishing attacks at client side using auto-updated white-list," *EURASIP J. Inf. Secur.*, vol. 2016, no. 1, pp. 1–11, May 2016.

[28] Y. Zhang, I. J. Hong, and F. L. Cranor, "Cantina: A content-based approach to detecting phishing web sites," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 639–648.

[29] Y. Joshi, S. Saklikar, D. Das, and S. Saha, "PhishGuard: A browser plug-in for protection from phishing," in *Proc. 2nd Int. Conf. Internet Multimedia Services Archit. Appl.*, Dec. 2008, pp. 1–6.

[30] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. Mitchell, "Client-side defense against web-based identity theft," [Online]. Available: http://crypto.stanford.edu/SpoofGuard/webspoof.pdf

[31] B. B. Gupta, A. Tewari, A. K. Jain, and D. P. Agrawal, "Fighting against phishing attacks: State of the art and future challenges," *Neural Comput. Appl.*, vol. 28, no. 12, pp. 3629–3654, Mar. 2016.

[32] A. P. E. Rosiello, E. Kirda, C. Kruegel, and F. Ferrandi, "A layout-similarity-based approach for detecting phishing pages," in *Proc. 3rd Int. Conf. Secur. Privacy Commun. Netw. Workshops (SecureComm)*, Sep. 2007, pp. 454–463.

[33] M. Dunlop, S. Groat, and D. Shelly, "GoldPhish: Using images for content-based phishing analysis," in *Proc. 5th Int. Conf. Internet Monitor. Protection*, May 2010, pp. 123–128.

[34] S. Afroz and R. Greenstadt, "PhishZoo: Detecting phishing websites by looking at them," in *Proc. IEEE 5th Int. Conf. Semantic Comput.*, Sep. 2011, pp. 368–375.

[35] S. C. Jeeva and E. B. Rajsingh, "Intelligent phishing URL detection using association rule mining," *Hum.-Centric Comput. Inf. Sci.*, vol. 6, no. 1, pp. 1–19, Jul. 2016.

[36] A. K. Jain and B. B. Gupta, "PHISH-SAFE: URL features-based phishing detection system using machine learning," in *Advances in Intelligent Systems and Computing*. Singapore: Springer, 2018, pp. 467–474.

[37] M. Aburrous, M. A. Hossain, K. Dahal, and F. Thabtah, "Intelligent phishing detection system for e-banking using fuzzy data mining," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 7913–7921, Dec. 2010.

[38] A. E. Aassal, S. Baki, A. Das, and R. M. Verma, "An in-depth benchmarking and evaluation of phishing detection research for security needs," *IEEE Access*, vol. 8, pp. 22170–22192, 2020.

[39] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Comput. Appl.*, vol. 25, no. 2, pp. 443–458, Nov. 2013.

[40] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Syst. Appl.*, vol. 117, pp. 345–357, Mar. 2019.

[41] A. C. Bahnsen, I. Torroledo, L. D. Camacho, and S. Villegas, "DeepPhish: Simulating malicious AI," in *Proc. APWG Symp. Electron. Crime Res.*, 2018, pp. 1–8.

[42] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL detection using machine learning: A survey," 2017, *arXiv:1701.07179*.

[43] A. AlEroud and G. Karabatis, "Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks," in *Proc. 6th Int. Workshop Secur. Privacy Anal.*, Mar. 2020, pp. 53–60.

[44] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE Access*, vol. 7, pp. 15196–15209, 2019.

[45] R. M. Mohammad, F. Thabtah, and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," in *Proc. Int. Conf. Internet Technol. Secured Trans.*, Dec. 2012, pp. 492–497.

[46] L. A. Tuan Nguyen, B. L. To, H. K. Nguyen, and M. H. Nguyen, "An efficient approach for phishing detection using single-layer neural network," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Oct. 2014, pp. 435–440.

[47] A. Subasi, E. Molah, F. Almkallawi, and T. J. Chaudhery, "Intelligent phishing website detection using random forest classifier," in *Proc. Int. Conf. Electr. Comput. Technol. Appl. (ICECTA)*, Nov. 2017, pp. 1–5.

[48] M. E. Pratiwi, T. A. Lorosae, and F. W. Wibowo, "Phishing site detection analysis using artificial neural network," *J. Phys., Conf. Ser.*, vol. 1140, Dec. 2018, Art. no. 012048.
[49] G. Sonowal and K. S. Kuppusamy, "PhiDMA—A phishing detection model with multi-filter approach," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 1, pp. 99–112, Jan. 2020.
[50] T. T. T. Pham, V. N. Hoang, and T. N. Ha, "Exploring efficiency of character-level convolution neuron network and long short term memory on malicious URL detection," in *Proc. 8th Int. Conf. Netw., Commun. Comput. (ICNCC)*, Dec. 2018, pp. 82–86.
[51] R. M. Verma, V. Zeng, and H. Faridi, "Data quality for security challenges," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2605–2607.
[52] S. Ariyadasa, S. Fernando, and S. Fernando, "Detecting phishing attacks using a combined model of LSTM and CNN," *Int. J. Adv. Appl. Sci.*, vol. 7, no. 7, pp. 56–67, Jul. 2020.
[53] V. Zeng, S. Baki, A. E. Aassal, R. Verma, L. F. T. De Moraes, and A. Das, "Diverse datasets and a customizable benchmarking framework for phishing," in *Proc. 6th Int. Workshop Secur. Privacy Anal.*, Mar. 2020, pp. 35–41.
[54] X. Dong, J. A. Clark, and J. Jacob, "Modelling user-phishing interaction," in *Proc. Conf. Hum. Syst. Interact.*, May 2008, pp. 627–632.
[55] M. A. Adebowale, K. T. Lwin, E. Sánchez, and M. A. Hossain, "Intelligent web-phishing detection and protection scheme using integrated features of images, frames and text," *Expert Syst. Appl.*, vol. 115, pp. 300–313, Jan. 2019.
[56] C.-Y. Huang, S.-P. Ma, W.-L. Yeh, C.-Y. Lin, and C.-T. Liu, "Mitigate web phishing using site signatures," in *Proc. IEEE Region 10 Conf.*, Nov. 2010, pp. 803–808.
[57] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2625–2634.
[58] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional ARMA filters," 2019, *arXiv:1901.01343*.
[59] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[60] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2009.
[61] K. L. Chiew, E. H. Chang, C. L. Tan, J. Abdullah, and K. C. Yong, "Building standard offline anti-phishing dataset for benchmarking," *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 7–14, 2018.
[62] S. Ariyadasa, S. Fernando, and S. Fernando, "Phishing websites dataset," Nov. 17, 2021. Distributed by Mendeley Data, doi: 10.17632/n96ncsr5g4.1.
[63] H. Le, Q. Pham, D. Sahoo, and C. H. Steven Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," 2018, *arXiv:1802.03162*.
[64] Y. Lin, R. Liu, D. M. Divakaran, J. Y. Ng, Q. Z. Chan, Y. Lu, Y. Si, F. Zhang, and J. S. Dong, "Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages," in *Proc. 30th USENIX Secur. Symp.*, Aug. 2021, pp. 3793–3810.
[65] T. Thakur and R. Verma, "Catching classical and hijack-based phishing attacks," in *Proc. Int. Conf. Inf. Syst. Secur.* Cham, Switzerland: Springer, 2014, pp. 318–337.
[66] A. Butnaru, A. Mylonas, and N. Pitropakis, "Towards lightweight URL-based phishing detection," *Future Internet*, vol. 13, no. 6, p. 154, Jun. 2021.

SUBHASH ARIYADASA received the bachelor's degree in information and communication technology from the University of Colombo School of Computing, Sri Lanka, in 2013. He is currently pursuing the Ph.D. degree with the University of Moratuwa, Sri Lanka.

From 2013 to 2015, he worked in the software industry and contributed to several foreign projects. Since 2015, he has been a Lecturer with the Department of Computer Science and Informatics, Uva Wellassa University, Sri Lanka. His research interests include cybersecurity, deep learning, reinforcement learning, and data mining technologies.

SHANTHA FERNANDO received the B.Sc. degree (Hons). in engineering and the Master of Philosophy degree from the University of Moratuwa, in 1993 and 2000, respectively, and the Ph.D. degree from the Delft University of Technology, The Netherlands, in 2010.

He is a Professor with the Department of Computer Science and Engineering, University of Moratuwa; and serves as the Director of Operations Technical Secretariat (OTS) with the University of Moratuwa. He is the Former Director of the Centre for IT Services (CITeS) and the Engineering Research Unit, University of Moratuwa. He is one of the founders and the Chief Advisor of TechCERT, affiliated to LK-Domain Registry. His research interests include in computer and information security, information systems, and e-learning. He became the first Chartered Engineer in Sri Lanka in the field of IT in the Institution of Engineers Sri Lanka (IESL) and has been a Founding Member of IT and Computer Engineering Sectional Committee of IESL. He served in the Council of the Computer Society of Sri Lanka. He also served as a member for the Governing Board of Lanka Education and Research Network (LEARN).

SUBHA FERNANDO (Member, IEEE) received the B.Sc. degree (special) (Hons.) in statistics and computer science from the University of Kelaniya, Sri Lanka, in 2004, and the Master of Engineering and Doctor of Engineering degrees from the Nagaoka University of Technology, Japan, in 2010 and 2013, respectively.

She is an Artificial Intelligence Research Scientist and a Senior Lecturer with the University of Moratuwa and a Consultant of Artificial Intelligence of CodeGen International (Pvt.) Ltd. She is the Coordinator and the Co-Founder of the QBITS—Artificial Intelligence Laboratory, University of Moratuwa. She is the Former Head of the Department of Computational Mathematics; the Former Director of the Information Technology Research Unit, University of Moratuwa; and the Past President of the Sri Lankan Association for Artificial Intelligence. She has been a resource person for many research workshops conducted by research bodies, including academia, and industry. Her research interests include deep learning, machine learning, multi-agent systems, and data mining technologies.